VM/ESA Storage Management with Tuning Guidelines

Document Number GG24-3934-01

August 1994

International Technical Support Organization
Poughkeepsie Center

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xv.

Second Edition (August 1994)

This edition, GG24-3934-01, is a revision of GG24-3934-00, and applies to Virtual Machine/Enterprise Systems Architecture* (VM/ESA) Release 2.2, program number 5684-112.

Changes and additions to the text and illustrations are indicated by a vertical line to the left of the change.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. H52 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400
USA

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Abstract

This document describes the VM/ESA storage hierarchy (central storage, expanded storage, and auxiliary storage), how VM/ESA manages these resources, and how the user can control and influence VM/ESA resource management. Several tuning guidelines and rules-of-thumb (ROT) are included in each chapter.

This document also includes an introduction to queueing theory as a foundation for a better understanding of system performance.

This document is intended for technical professionals in general, and for system programmers and system administrators in particular. A good knowledge of VM/ESA and a general knowledge of system performance are assumed.

VM (254 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help technical professionals to understand how different storage resources may affect the performance of VM/ESA.

The information in this publication is not intended as the specification of any programming interfaces that are provided by VM/ESA. See the PUBLICATIONS section of the IBM Programming Announcement for VM/ESA Release 2.2 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT 06904 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

ACF/VTAM	AIX/ESA	DFSMS	DFSMSdss
ECKD	ES/3090	ES/4381	ESA/370
ESA/390	ES/9000	ES/9021	ES/9221
ESCON	GDDM	Hyperbatch	IBM
MVS/ESA	OfficeVision	OS/2	PR/SM
PS/2	RAMAC	SAA	System /370
System /390	S/390	SQL/DS	VM/ESA
VM/SP	VM/XA	VM/SP HPO	VSE/ESA
VTAM			

Preface

This book helps you understand how the storage resources (central storage, expanded storage, and auxiliary storage) relate to each other and how to get performance benefits out of the available resources. It gives you a detailed view of the performance aspects of storage resources in VM/ESA Release 2.2 and how they interact with each other.

This book provides guidelines to help you decide what to do when the system is running short of storage resources; for example, when the installation cannot meet the end user's response time as committed in the service-level agreement.

Usually, the system runs short of resources because the work load (number of transactions, for example) increases over a period of time without a corresponding increase in the storage resources. The increase in transactions might have been caused by a higher number of users or by a change in the work profile of the average user because applications are increasingly using data bases, for example.

This document intends to help you answer the following kind of questions:

- Which storage resources are constraining the system?
- Which storage resources should be increased?
- How much more storage is necessary, and in what combination?
- Which users should be favored with which resources?
- Which effects have a certain combination of those resources on systems performance?
- Should the dynamic expanded storage routine (arbiter) be bypassed?

| Where the addition of new or enhanced functions or rewording for improved user
| understanding caused changes to the text and illustrations, modifications are
| flagged with a vertical bar on the left, as is this paragraph. Minor editorial
| corrections are not flagged.

How This Document is Organized

The document is organized as follows:

- Part 1

  Describes VM/ESA Storage Management and consists of the following chapters:

  - Chapter 1: Storage Hierarchy
  - Chapter 2: Virtual Storage
  - Chapter 3: Central Storage
  - Chapter 4: Expanded Storage
  - Chapter 5: Direct Access Storage Devices (DASD)
  - Chapter 6: Data-in-Memory Facilities
  - Chapter 7: Storage Administration

- Part 2

  Introduces elementary queueing theory models and consists of the following chapters:

  - Chapter 8: Probability Theory Refresher
  - Chapter 9: Introduction to Queueing Theory
  - Chapter 10: Simulation

Related Publications

The following publications contains more information about the topics covered in this document:

- *IBM ESA/390 Principles of Operation*, SA22-7201
- *VM/ESA: Planning and Administration*, SC24-5521
- *VM/ESA: CP Command and Utility Reference*, SC24-5519
- *VM/ESA: Introduction and Features Summary*, SC24-5651
- *VM/ESA: CP Programming Services*, SC24-5520
- *VM/ESA: Performance*, SC24-5642
- *VM/ESA Release 1.1 Performance Report*, GG66-3236
- *VM/ESA Release 2 Performance Report*, GG66-3245
- *VM/ESA Release 2.1 Performance Report*, GC24-5673-00
- *VM/ESA Release 2.2 Performance Report*, GC24-5673-01
- *VMPRF User's Guide and Reference*, SC23-0460
- *RTM VM/ESA Program Description/Operations Manual*, SH26-7000
- *Device Support Facilities User's Guide and Reference*, GC35-0033
- *Introduction to Nonsynchronous Direct Access Storage Subsystems*, GC26-4519
- *IBM 3380 Direct Access Storage Introduction*, GC26-4491
- *Using the IBM 3380 Direct Access Storage in a VM Environment* , GC26-4493
- *IBM 3390 Direct Access Storage Introduction*, GC26-4573
- *Using the IBM 3390 Direct Access Storage in a VM Environment*, GC26-4575
- *IBM 3990 Storage Control Introduction*, GA32-0098
- *IBM 3990 Storage Control Planning, Installation, and Storage Administration Guide*, GA32-0100
- *IBM 9340 Direct Access Storage Subsystems Introduction*, GC26-4694
- *Using the IBM 9340 Direct Access Storage Subsystems in a VM/ESA Environment*, GC26-4644
- *IBM 9340 Direct Access Storage Subsystems Reference*, GC26-4647
- *IBM RAMAC Array Subsystem Introduction*, GC26-7004
- *IBM RAMAC Array DASD Introduction*, GC26-7012

International Technical Support Organization Publications

The following ITSO publications contain related information:

- *VM/ESA Release 1.1 Overview and Usage Experiences*, GG24-3744
- *VM/ESA Release 2 Overview*, GG24-3860
- *VM/ESA Release 2 Usage and Experience*, GG24-3932
- *VM/ESA Release 2.1 Overview*, GG24-4024
- *VM/ESA Release 2.1 Usage and Experience*, GG24-4032
- *VM/ESA Release 2.2 Overview and Usage Experiences*, GG24-4219
- *ECKD and Nonsynchronous DASD I/O*, GG24-3571
- *IBM Storage Subsystem Enhancements*, GG24-3886

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Technical Publications

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain books on a variety of products.

Other References

The following books are good references for probability and queueing theory:

- Goldberg, S., *Probability - An Introduction*, Dover Publications Inc., New York, 1960
- Mangulis, V., *handbook of Series for Scientists and Engineers*, Academic Press, New York, 1965
- Feller, W., *An Introduction to Probability Theory and its Applications, Volume I*, John Wiley and Sons, New York, 1968
- Rozanov, Y., *Probability Theory - A Concise Course*, Dover Publications Inc., New York, 1969
- Gross, D. and Harris, C., *Fundamentals of Queueing Theory*, John Wiley and Sons, New York, 1974

- Kleinrock, L., *Queueing Systems, Volume I: Theory*, John Wiley and Sons, New York, 1975
- Hillier, F. and Yu, O., *Queueing Tables and Graphs*, John Wiley and Sons, New York, 1981
- Allen, A., *Probability, Statistics, and Queueing Theory*, Academic Press, Inc., New York, 1990
- Medhi, J., *Stochastic Models in Queueing Theory*, Academic Press, Inc., New York, 1991

Acknowledgements

This publication is the result of a residency conducted at the ITSO Poughkeepsie in October, 1992; it was updated in July 1994.

The authors of this document are:

Francisco Grossi	IBM ITSO Poughkeepsie
Lee Ley Cheng	IBM Malaysia
Douglas Newbould	IBM U.K.
João Páscoa	IBM Brazil
Bernd Rueckert	IBM Germany
Theeraphong Thitayanun	IBM Thailand

Thanks to the following people for the invaluable advice and guidance provided in reviewing this document:

Bill Bitner	IBM Endicott
Wayne Smith	IBM Endicott
Ann Gleason	IBM Endicott
Scott Vetter	IBM ITSO Poughkeepsie
Craig Welch	IBM ITSO San Jose
Chris Saul	IBM ITSO San Jose
Bruce Mcnutt	IBM San Jose (USA)
Guy de Ceulaer	IBM Belgium
Kris Buelens	IBM Belgium

Part 1. VM/ESA Storage Management

This part contains a description of VM/ESA Storage Management and the general aspects of storage administration. This part consists of the following chapters:

Chapter 1: Storage Hierarchy

Describes the various levels of the storage hierarchy, such as registers, high speed buffer, central storage, expanded storage, DASD, optical devices, and tapes.

Chapter 2: Virtual Storage

Describes how VM/ESA implements and manages virtual storage for the Control Program (CP) and for virtual machines. Data spaces, virtual disks, auxiliary storage, and the page subsystem are among the subjects described in this chapter.

Chapter 3: Central Storage

Describes how VM/ESA manages central storage, including topics such as storage layout and its main areas (V=R area, CP nucleus, free storage, trace table), and some topics related to the paging subsystem.

Chapter 4: Expanded Storage

Describes how VM/ESA manages expanded storage, including topics such as allocation, deallocation, migration, and dedication.

Chapter 5: Direct Access Storage Management (DASD)

Describes the main characteristics of DASD and its management, and provides a detailed description of DASD response time.

It also describes which RTM and VMPRF reports are important in managing DASD performance, and helps you correlate DASD parameters with the variables reported.

IBM technical professionals have access to tools, such as Cache Analysis Aid (CAA) and the DASD Cache Analysis Tool (DCAT), that can model and predict DASD response times for specific workloads. However, we believe that a good understanding of the subject is an important foundation for understanding DASD performance.

Chapter 6: Data-in-Memory Facilities

Describes some of the data-in-memory facilities available in VM/ESA, such as data caching, minidisk cache, virtual disk, and minidisk mapping.

Chapter 7: Storage Administration

Provides a description of the VM/ESA scheduler and the commands you can use to understand and tune your storage resources. It describes each command and provides some guidelines about its usage and effects on the behavior of the scheduler.

Two tables in the end of this chapter provide guidelines and threshold values that should cause you to investigate if they are exceeded.

Chapter 1. Storage Hierarchy

The technological evolution of large computer systems created several levels of physical storage with different characteristics such as density, speed, and price.

As expected, the fastest storage devices are also the most expensive. Usually, the faster a storage device is, the more electrical power it consumes; high electric power consumption generates heat.

In order to balance speed, power requirements, and heat dissipation, large computers combine a hierarchy of storage as shown in Figure 1.

If you scale access time up to human terms, data in central storage can be accessed in one second, data in expanded storage can be accessed in 17 minutes, data in cached DASD takes 12 hours, and data on DASD takes 3 days.

VM/ESA manages the storage hierarchy in a way that brings the most important data closer to the processor. The paging algorithms keep frequently used data and code in the higher levels of the storage hierarchy. The less frequently used data is moved to lower levels as it becomes less active. Later, data is brought back to central storage, on demand or by anticipation.

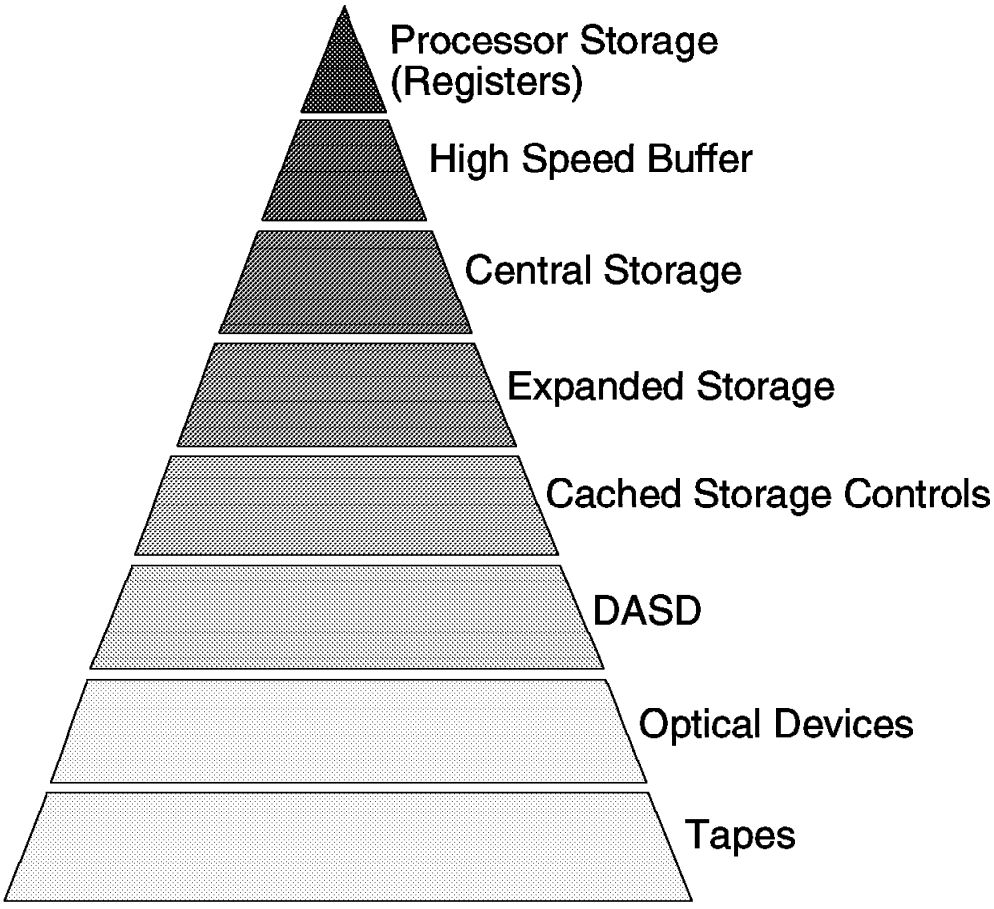

Figure 1. Storage Hierarchy

Registers

To perform its functions, the processor may itself use a certain amount of internal storage, called registers. Some of these registers are:

- General registers
- Floating-point registers
- Control registers
- Access registers

General Registers

General registers are identified by the numbers 0-15 and contain 32 bits each. Instructions may designate information in one or more of the 16 general registers. They may be used as base-address registers and index registers in address arithmetic, as well as accumulators in general arithmetic and logical operations.

When used as accumulators, general registers are designated by a 4-bit R field in an instruction. Some instructions permit addressing of multiple general registers by having several R fields.

In addition to being used as accumulators, 15 of the 16 general registers are also used as base-address and index registers in address generation. In these uses, the registers are designated by a 4-bit B field or X field in an instruction. A value of zero in the B or X field specifies that no base or index is to be applied, and thus, register zero cannot be designated as containing a base or index address.

Floating-Point Registers

The four floating-point registers available for floating-point operations are designated by the numbers 0, 2, 4, and 6, and are specified as a 4-bit R field in floating-point instructions. Each floating-point register is 64 bits long and can contain a short (32-bit) or a long (64-bit) floating-point operand.

Control Registers

There are 16 control registers, each having 32 bits. The bit positions in the registers are assigned to particular facilities in the system, such as interrupt control masks, and are used either to specify that an operation can take place or to provide special information required by the facility.

The information specified in control registers includes:

- External interruption masks
- PSW key mask
- I/O interruption subclass mask
- Monitor masks
- PER masks

The control registers are also used to specify some VM-related information such as the IUCV external interruption mask.

The control registers are identified by the numbers 0-15 and are designated by 4-bit R fields in the LOAD CONTROL and STORE CONTROL instructions.

Access Registers

The access registers, introduced with the ESA/370 architecture, contain an indirect specification of a segment-table designation, which is used by the dynamic address translation (DAT) mechanism to translate references to a corresponding address space. They are identified by the numbers 0-15 and contain 32 bits each.

When the processor is in access-register mode, controlled by bits 16-17 in the PSW, an instruction B field, used to specify a storage-operand reference, also designates an access register, and the corresponding segment-table designation specified by the access register is used by DAT for the reference being made. For some instructions, such as MOVE LONG (MVCL), both R fields can designate access registers.

Access registers 1-15 can each designate any address space, including the current instruction space, called primary address space. Access register zero always designates the primary address space. Therefore, in access-register mode, the 16 access registers can designate, at any one time, a maximum of 16 address spaces.

High Speed Buffer

The High Speed Buffer (HSB) is built with fast bipolar transistors. The HSB is expensive, consumes high power, and generates significant heat per bit.

Each processor in a multiprocessor complex has its independent HSB; some models implement a 2-level HSB. The size of the HSB ranges from 16 KB to a few megabytes in the ES/9000 series. The size of the HSB greatly influences instruction processing speed.

Data is brought from central storage to the HSB before it can be accessed by the processor. An automatic paging system implemented in hardware maintains the most accessed portions of central storage in the HSB and pages the least recently used (LRU) portions out to central storage.

HSBs are implemented using two techniques:

- Store-through technique, used by old implementations, always updates the corresponding central storage locations when the HSB is updated.
- Store-in technique, used by the IBM ES/9000 family, updates central storage only when the LRU algorithm moves the corresponding locations out of the HSB.

A castout is the process of transferring data from the HSB to central storage to make room for new data. A line of data stays in HSB until a castout is required by the System Control Element (SCE).

When data is not found in the HSB, a transfer of data from central storage (castin) is performed synchronously; that is, the processor waits until the operation is completed. In other words, the speed of the operation is determined by the central storage access time.

The HSB functions almost transparently to the software; VM/ESA code, however, tries to optimize HSB usage by defining the main control blocks as multiples of the cache line size.

Central Storage

Built with Complementary Metal Oxide Semiconductor (CMOS) technology, central storage is less expensive than bipolar storage, but not as fast. Central storage consumes less power, generates less heat, and is cheaper than the HSB.

Central storage is the primary storage resource managed by VM/ESA. Without a HSB working at high hit ratios, however, central storage would not be able to sustain a high rate of instruction processing.

Central storage contains control program (CP) resident code, tables, control blocks, and buffers, and is used to implement virtual storage.

Before code and data can be accessed by the processor, they must be made resident in central storage. Code and data cannot be processed by the processor when they are in the virtual storage portions that reside in auxiliary storage.

Large central storages provide high performance levels to VM/ESA because the page activity is reduced. VM/ESA supports central storages as large as 2 GB, which is the maximum size supported by the ESA/390 architecture.

Large central storages are expensive; therefore, the best price-performance is usually achieved with a reasonable paging rate between central storage and auxiliary storage (expanded storage and DASD).

Central storage is accessed by the processors and by the channels.

Processors

In a multiprocessor system, several processors may access central storage concurrently. Interference may be caused by hardware and software serialization when more than one processor accesses central storage.

For example, you may observe a 2-processor system internal throughput rate (ITR) about 5% lower than the sum of the ITRs of two separate systems; that is:

$$ITR_2 \quad 1.9 \cdot ITR_1$$

However, single-image systems have many advantages, such as simplified operation and a better work load balancing, that compensate for the processor's interference.

Channel Subsystem

The channel subsystem cannot access expanded storage directly. When a virtual machine issues an I/O operation that is intercepted by VM/ESA, CP initiates the CCW translation process.

As part of this process, CP looks for I/O data buffers in the virtual storage of the virtual machine. If the buffers are already in central storage, CP marks the page frames, for the duration of the I/O, as nonpageable; that is, the buffers are page fixed. If the buffers are in auxiliary storage (either expanded storage or DASD), CP pages them into the central storage.

Expanded Storage

In the ES/9000 series, expanded storage is built with the same 4-megabit chips used for central storage. However, the implementation of expanded storage is less complex than the implementation of central storage because, among other factors, its address resolution is 4 KB and storage keys are not required.

Some processors, such as the ES/9021 models, are implemented with separate central storage and expanded storage. Others, such as the ES/9121 models, may have their total storage partitioned between central storage and expanded storage, but have a limit on the maximum central storage size. The total storage portion exceeding this limit must be defined as expanded storage. The current limit for the central storage on the ES/9121 series is 512 MB.

Data in expanded storage cannot be processed directly. VM/ESA must bring 4-KB blocks of expanded storage to central storage through special machine instructions before data can be used by the processors.

Expanded storage is an optional resource for VM/ESA, and it can be used as:

- Minidisk caching
- Paging auxiliary storage
- Dedicated to guests

Minidisk caching (MDC) is a VM/ESA function that requires expanded storage in VM/ESA Release 2.1 and prior VM/ESA releases; that is, if expanded storage is not available for CP use, minidisk caching is not performed in those VM/ESA releases. In VM/ESA Release 2.2, expanded storage is not required for MDC.

The use of expanded storage for paging is not required but, when expanded storage is available, the VM/ESA paging algorithms are highly efficient.

Some guest functions also require portions of expanded storage to be attached to their virtual machines. This is the case of the MVS Hyperbatch function, for instance. CMS itself does not use expanded storage but gets the benefits through CP paging algorithms and MDC.

When total storage can be configured as central and expanded storage (as in the ES/9121), you must make a decision on how to partition it.

When paging is a bottleneck, the rule is to have as much central storage as possible; no paging is better than efficient paging.

When the paging rate is low, you can configure part of the total storage as expanded storage and reserve the configured expanded storage for guest use, for example.

In VM/ESA Release 2.1 and prior releases, you may want to configure part of total storage as expanded storage for MDC. This should be done when paging rates are moderate; in this case, all configured expanded storage not attached to guest should be reserved for MDC. It is possible, on systems with poor I/O performance, to achieve better overall throughput by trading higher paging rates (less central storage) with faster data access provided with increased cache storage (more expanded storage).

In VM/ESA Release 2.2 there is no advantage to configure part of total storage as expanded storage for MDC.

DASD Cache

Cache storage controls keeps the most accessed DASD data accessible without requiring device-level services.

Data in cache is still accessed through channels and standard I/O operations, but the response time of a cache hit is several times faster than a typical disk I/O requiring mechanical movement. A typical DASD I/O access can take around 20 milliseconds, while a cache hit can be as fast as 2 milliseconds.

The IBM 9340 Subsystem provides an efficient cache storage that is transparent to the software. It is ideal for small and intermediate systems that can benefit from cache performance for applications with high read-to-write ratios.

The IBM 3990 Models 3 and 6 provide extended functions that enhances cache storage performance for applications with high write-to-read ratios. With the dual copy function, these devices are excellent solutions to protect the availability of critical data bases.

DASD Storage

Auxiliary DASD storage is required by VM/ESA. Although not as efficient as expanded storage for paging, DASD can sustain reasonable paging rates.

VM/ESA uses DASD as auxiliary storage, relying on sophisticated paging algorithms that maximize the efficiency of the flow of pages between DASD and central storage.

DASD are nonvolatile storage devices. They are used by VM/ESA to keep saved segments, spool, warm-start data, and other information that must be maintained across power-off conditions. DASD are not removable, however.

See Chapter 5, “Direct Access Storage Devices (DASD)” on page 41 for more information.

IBM 3995 Optical Library Dataserver

The IBM 3995 Optical Library Dataserver is a family of optical storage library products that provide low-cost, direct access to online information. Due to its slower access speed, the optical library is ideally suited for applications with high volumes of data but that infrequently reference the data.

Applications for the IBM 3995 can be cost-justified replacements for current microfilm/microfiche and similar applications, and as a migration-level 2 (ML2) storage for DFSMS/VM.

Tape Storage

Magnetic tape storage is cheap and slow. It fulfills the requirement of being removable and transportable. It is also the primary low-cost medium for backups.

Chapter 2. Virtual Storage

Modern operating systems, as well as today's hardware computer architectures, focus on virtual storage rather than real storage.

Virtual storage is an economical solution for the demand for large amounts of computer storage. Hardware and software are combined in order to simulate large virtual storage sizes in a relatively small amount of real storage.

Virtual storage is composed of address spaces and data spaces. Address spaces may contain code (executable processor instructions), as well as data. A data space can be used only for data storage and manipulation. Programs cannot execute in this area, although a data space can hold programs stored as data.

Data can be stored in a data space starting at address zero, because a data space does not contain nucleus-reserved areas. However, all data must be managed by the application program.

Address Types

Three basic types of addresses are recognized when addressing storage:

- Absolute address
- Real address
- Virtual address

The addresses are distinguished on the basis of the transformations that are applied to them during a storage access. Address translation converts virtual to real addresses, and prefixing converts real to absolute addresses. In addition to these three basic address types, other types are defined, but are treated as one of the three basic types, depending on the instructions and the current addressing mode.

Absolute Address

An absolute address is the address assigned to a central-storage location. It is used to access storage without any transformation. Available central storage is usually assigned contiguous absolute addresses starting at zero.

Storage consisting of byte locations sequenced according to their absolute addresses is referred to as absolute storage.

Real Address

A real address identifies a location in central storage. When a real address is used for an access to central storage, it is converted, by means of prefixing, to an absolute address. The particular transformation is defined by the value in the prefix register for the processor.

At any instant there is one real-address to absolute-address mapping for each processor in the configuration.

Storage consisting of byte locations sequenced according to their real addresses is referred to as real storage.

Virtual Address

With the appropriate support of a control program, the Dynamic Address Translation (DAT) facility may be used to provide a user with a system in which storage appears to be larger than the central storage that is available in the configuration. This apparent storage is referred to as virtual storage, and the addresses used to designate locations in the virtual storage are referred to as virtual addresses.

Virtual storage is a programming abstraction, implemented in the real system by means of DAT, central storage, expanded storage, DASD volumes, and a control program to manage the necessary translation tables and controls.

When a virtual address is used to access central storage, it is translated by means of DAT to a real address, which is further converted by prefixing to an absolute address.

Depending on the translation mode, a virtual address, as discussed in the next sections, may be a:

- Primary virtual address
- Secondary virtual address
- Home virtual address
- Access-register-specified virtual address

Address Spaces

An address space is simply a sequence of virtual addresses associated with a virtual storage, together with the specific transformation parameters that allow each virtual address to be associated with a byte location in central storage.

DAT uses two levels of tables, segment tables and page tables, as transformation parameters. DAT uses, at different times, the segment table designations in different control registers or specified by the access registers. The choice is determined by the translation mode specified in bits 16-17 of the current PSW. Five translation modes are available:

- Real mode
- Primary-space mode
- Secondary-space mode
- Home-space mode
- Access-register mode

When the PSW specifies real mode (DAT off), the processor does not translate addresses.

When the PSW specifies primary-space mode or secondary-space mode, the processor can translate virtual addresses belonging to two address spaces; the primary address space and the secondary address space.

When the PSW specifies access-register (AR) mode, the processor can translate virtual addresses of up to 16 address spaces; the primary address space and up to 15 AR-specified address spaces.

When the PSW specifies home-space mode, the processor can translate virtual addresses of the home address space.

These different modes refer to the corresponding address spaces, where addresses are translated by means of a primary, secondary, home, or AR-specified segment-table designation, respectively.

The primary and secondary segment-table designations are in control registers 1 and 7, respectively. The home segment-table designation is in control register 13. The AR-specified segment-table designations are in control registers 1 and 7 and in table entries designated by the corresponding access register.

After selection of the appropriate segment-table designation, the translation process is the same for all four types of virtual addresses.

VM/ESA uses primary address spaces and AR-specified address spaces; secondary address spaces and home address spaces are not used.

Data Spaces

Instruction addresses are addresses used to fetch instructions from storage. They are treated as real addresses in real mode; as primary virtual addresses in primary-space mode; secondary-space mode, or AR-mode; and as home virtual addresses in home-space mode.

The storage-operand addresses for most instructions are called logical addresses. Logical addresses are treated as real addresses in real mode, as primary virtual addresses in primary-space mode, as secondary virtual addresses in secondary-space mode, as home virtual addresses in home-space mode, and as AR-specified virtual addresses in AR mode.

Note: Some instructions have storage-operand addresses that do not follow the rules for logical addresses. The instruction definition in *IBM ESA/390 Principles of Operation* provides a precise definition of the type of address for these instructions.

Therefore, no matter which translation mode — primary, secondary, or AR-specified mode — the instructions are always fetched from the primary address space. However, the operands may refer to primary, secondary, home, or AR-specified address spaces. The same address space may be, at the same time, the primary, the secondary, and the home address space.

In AR mode, any address space other than the primary address space can be used only to obtain storage operands of instructions. These address spaces can contain only data, and therefore, are often called data spaces.

Only primary-space mode and AR mode are valid under VM/ESA; these two translation modes can be set by the SET ADDRESS SPACE CONTROL (SAC) semi-privileged instruction as follows:

SAC 0 Primary-space mode

SAC 512 Access-register mode

When in primary-space mode, logical addresses refer to the primary address space. When in AR mode, logical addresses refer to the primary address space when the base register is zero or when the contents of the access register corresponding to the base register (same register number) is zero; otherwise, logical addresses refer to the data space identified by the contents (Address Space Entry Token, ALET) of the access register.

VM Data Spaces

Data spaces was implemented in VM/ESA Release 1.1. This implementation requires SIE extensions, called VM Data Spaces (sometimes also called ESA/390 data spaces), which is part of the ESA/390 architecture, and is currently available only on the ES/9000 family.

ESA/370 data spaces, as used by MVS and VSE are different from VM data spaces. The way MVS manages data spaces would not be appropriate for a VM implementation, because the MVS data space control structures are created and managed by the MVS control program. Therefore, this structure is local to one particular operating system and cannot be shared with another image of any other operating system.

A problem of a data space implementation in VM is the use of DAT, which is the mechanism used by the ESA architecture to translate any virtual address — primary, secondary, home, or AR-specified — to a real address. CMS relies on CP to manage its virtual storage and is not concerned with DAT. Changing this implementation would add considerable complexity to CMS and probably would cause incompatibilities to existing applications.

The implemented solution extended the SIE instruction and defined a new architecture, called ESA/Extended Architecture (ESA/XC), that works with DAT off but is still capable of creating and sharing data spaces. ESA/XC can exist only in a virtual machine and is, therefore, called a virtual architecture.

All the control structures necessary to create and manage data spaces reside in CP, thus allowing all virtual machines managed by CP to share the data spaces owned by any other virtual machine.

Although the concept of data spaces does not exist under the S/370 or 370/XA architectures, 370- and XA-mode virtual machines can still invoke CP services to copy data from an address space into its primary address space.

This support can be very useful for data base management, large graphic applications, and other applications requiring large areas in storage. Applications can now:

- Have more than 2047 MB of virtual machine storage for data.

  An application running in an XC virtual machine can address more storage by creating multiple data spaces of up to 2 GB each . It enables applications to address virtually unlimited amounts of data.

- Improve their storage data integrity by segregating different types of data into different address spaces.

  Applications can use data spaces to isolate data by its particular usage and then share that data only among related users. This is an alternative to using a common area that may contain data for various purposes.

- Improve its performance by using data-in-memory facilities, allowing access to data at storage speed rather than at I/O device speed.

  Application programs requiring access to large amounts of data may directly read data into data spaces or, alternatively, use the MAPMDISK macro service to establish a mapping between minidisks and data spaces, thus using data spaces as a huge buffer for DASD blocks, as shown in Figure 2.

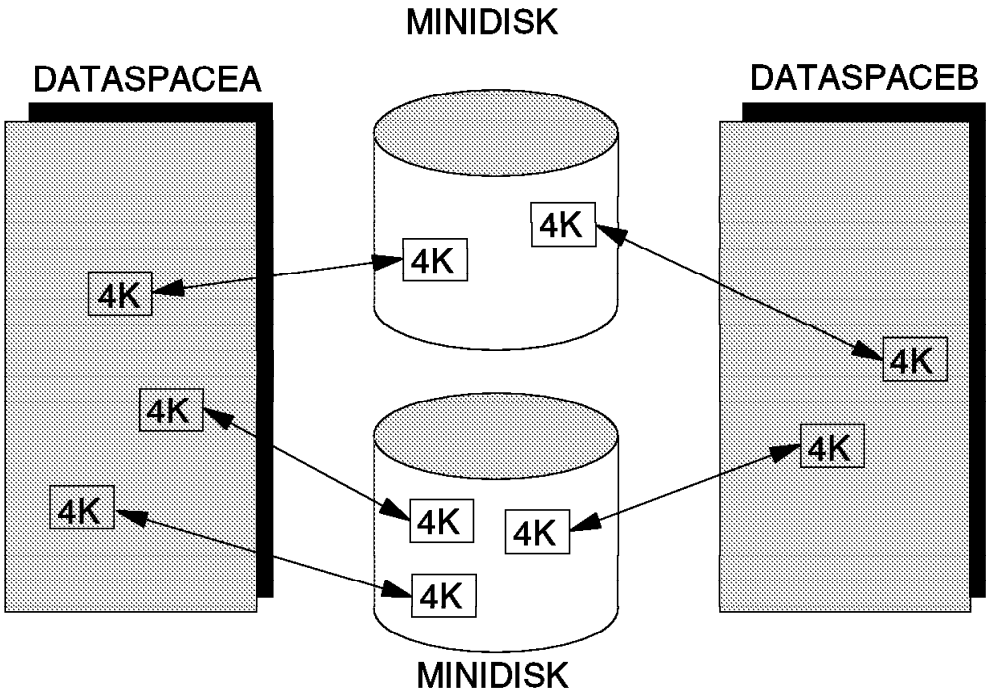

Figure 2. Minidisk Mapping

By temporarily placing the minidisk data in a data space, the application can use ordinary processor instructions to access data in the data space, rather than using I/O instructions to get data from DASD. CP pages minidisk blocks into storage when they are first referenced and pages data out to the minidisk blocks during the steal process. Since paging operations are used instead of virtual I/O operations, you may see an increase in paging activity when you start using VM data spaces.

- Act as a service provider, using shared data spaces, for example, to replace a communication link.

  Applications can even share data located in its primary address space with programs executing in other virtual machines.

Exploitation

The VM components and IBM program products that exploits VM data spaces are:

- SQL/DS Version 3 Release 3 with Data Space Feature
- VS FORTRAN Version 2 Release 5
- CMS Shared File System (SFS)

SQL/DS uses the MAPMDISK function to map the directory information and DBspaces in VM data spaces. Internal DBspaces are created and managed directly into data spaces. Additionally, SQL/DS uses the REFPAGE and PFAULT system service macros to increase performance.

VS FORTRAN uses VM data spaces to define extended COMMON blocks of up to 2 GB each, thus extending its addressing capabilities.

Shared File System (SFS) uses the MAPMDISK function to map directory control directories in VM data spaces.

It is worth mentioning that the SFS exploits the sharing aspect of data spaces, where SQL uses private data spaces for data in memory, CP I/O, and asynchronous page faults. Therefore, different planning requirements are required to handle SQL's greater storage and paging configuration compared with SFS's data sharing and potentially lower storage and paging requirements.

SQL/DS

SQL/DS Version 3 Release 3 with the Data Space Feature maps its disk files to VM data spaces using the MAPMDISK function. It maintains different types of data in three types of minidisks: directory information, DBspaces, and temporary data used as work areas, called internal DBspaces.

During SQL/DS initialization, the directory information is mapped to a data space and remains mapped until SQL/DS terminates. As many data spaces as necessary are created to map all DBspace minidisks, provided that enough storage has been authorized in the SQL server user directory. The database page number is associated with the corresponding mapped block, allowing SQL/DS in most cases to calculate the virtual address of a DBspace page without reference to the directory. This mapping also remains effective until SQL/DS terminates.

The DBspaces are not mapped all at once. A segment is mapped when the first reference is made to the segment. SQL/DS maintains a segment bit map indicating whether that segment has already been mapped.

Whenever a page is updated, the directory is referenced to determine whether shadowing is required. If so, a new database block is allocated and the corresponding data space page is mapped to this new block.

Internal DBspaces are work areas that are not recoverable between failures. SQL/DS allows you to define internal DBspaces as either:

- Mapped internal DBspaces, mapped to data spaces using the MAPMDISK function.
- Unmapped internal DBspaces, created in data spaces and paged out to the system paging areas.

The MAPMDISK SAVE macro ensures that all updated pages are actually written to disk at checkpoint time. The save process is asynchronous; SQL/DS receives an external interrupt signal when the specified pages are written to disk. In the meantime, SQL/DS is free to process other work.

SQL/DS also exploits the asynchronous page fault services provided by the VM data space implementation. Whereas a page fault previously interrupted the entire SQL/DS virtual machine, it now interrupts only the affected database task. This service allows SQL/DS to proceed in processing another agent. Furthermore, multiple page faults may occur in many data spaces, and all are handled asynchronously by VM/ESA.

Each task is further optimized by using the page reference services to reduce the number of page faults. SQL/DS uses the REFPAGE macro to inform the VM/ESA paging subsystem about the upcoming virtual storage reference pattern,

allowing CP to optimize scheduling of these pages from DASD and reducing the number of page faults.

Note that mapping is done by the server machine; the user still uses APPC/VM to send the request to the SQL/DS server and receive information from the server.

VS FORTRAN

The VS FORTRAN Version 2 Release 5 exploitation of data spaces uses the extended addressability required by the definition of large arrays.

VS FORTRAN programs in general tend to process data using arrays. Those programs related to Numerically Intensive Computing (NIC) applications use very large arrays which are normally defined inside COMMON blocks to allow them to be shared among the various subroutines.

The FORTRAN language implemented by VS FORTRAN Version 2 Release 5 allows the definition of three types of COMMON blocks:

- Static COMMON blocks (SC)

  Static COMMON blocks are defined within the compiled program, and therefore, are always in the primary address space. Additionally, they must reside below the 16-MB line.

  This is the most restrictive type of COMMON block and the only type available in old implementations of the FORTRAN language.

- Dynamic COMMON blocks (DC)

  Dynamic COMMON blocks are allocated at run time in the primary address space. They can reside above the 16-MB line, but the sum of all COMMON blocks are restricted to the maximum size of the primary address space (2047 MB) less the space used by the program code and other data.

  Although providing more room for array addressability and not so restrictive as the static blocks, the dynamic COMMON blocks do not really solve the problem for programs using very large arrays.

- Extended COMMON blocks (EC)

  This new type of COMMON block was introduced in VS FORTRAN Version 2 Release 5. Extended COMMON blocks are also allocated at runtime, but the data can reside in data spaces. There are two options:

  - ECPACK

    All blocks are allocated in just one data space; the sum of all COMMON blocks is limited to 2 GB.

  - NOECPACK

    Each COMMON block is allocated in a separate data space, extending the size limit to 2 GB for each block.

  Extended COMMON blocks provide a good solution for programs that require significant addressability for arrays.

The COMMON block type is defined at compile time; if nothing is specified, the COMMON block is assumed to be static. Therefore, a working VS FORTRAN program does not have to be modified to exploit data spaces; just add the extended COMMON option and re-compile the program.

Shared File System

SFS can exploit the data spaces facility provided by ESA/XC architecture for directory control directories. The contents of the directory itself and all file data within the directory are mapped to data spaces, providing users with a substantial performance improvement.

Authorized users who access the directory in read-only (R/O) mode do not have to communicate with the server. Instead, CMS gets the desired data directly from the data space, which provides the following performance benefits:

- All the overhead associated with normal communication (such as APPC/VM) with the server machine is eliminated.
- CMS retrieves data from shared virtual storage, which is far more efficient than having the server read from DASD for each user.
- For XC-mode users, CMS does not use its own virtual machine to maintain control blocks associated with an accessed directory. Instead, it gets the data directly from the shared copy in the data space. Because users refer to a single copy of control blocks, real storage requirements are reduced.

For a directory to reside in VM data spaces, the following must apply:

- VM/ESA Release 1.1 (or later) is running in an ES/9000 processor.
- The server virtual machine is running in ESA/XC mode.
- The directory has the DIRCONTROL attribute and is accessed in R/O mode with the ACCESS command.

  The ACCESS command must be used to access the directory. As CSL routines can refer to SFS files directly specifying filename, filetype, and dirid, some applications may not have to access the directory.

  However, if the directory is accessed before the start of the application, data spaces can be used even if the application references the file by specifying its dirid instead of the filemode.
- The directory is eligible for a data space through the administrator DATASPACE ASSIGN command.
- Enough space is available in the data spaces.

  When a file is changed in a DIRCONTROL directory, current accessors still see the old version mapped to the corresponding data space. On the other hand, new accessors must see the changes; therefore, SFS creates a new data space and maps the new version of the directory.

  When updates are frequent, SFS may have to create so many data spaces for different versions of the directory that it would soon exhaust the limit allowed in the SFS server virtual machine.

  Thereafter, new R/O accessors do not use the directory in a data space until a sufficiently large data space becomes available. This is also bad for performance as data space pages are shared with fewer users.

  The QUERY ACCESSORS command with the DATASPACE option shows whether a directory currently resides in a data space and how many levels of the same directory are currently accessed.

The SFS server virtual machine issues a warning message notifying you if there are any cases where an eligible directory is not able to use data spaces.

The following situations force the use of APPC/VM communications between users and servers:

- When aliases are used to refer to a base file.

  Although aliases cannot exist in DIRCONTROL directories, it is possible to create an alias in a FILECONTROL directory that refers to a base file in a DIRCONTROL directory.

- When an INPLACE file is read.
- When users are accessing a remote filepool.

Figure 3 illustrates the use of data space by server and user virtual machines.

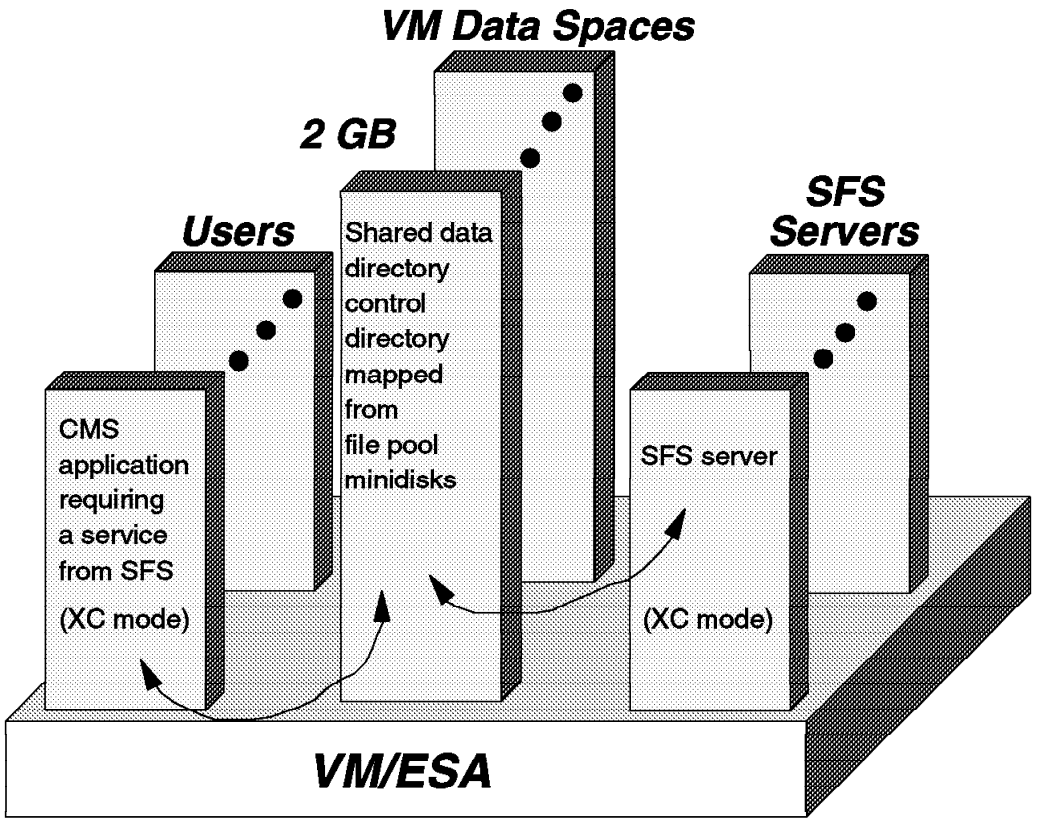

Figure 3. SFS Use of ESA/XC Services

The following criteria should be used to optimize data space usage by DIRCONTROL directories:

- Select read-only directories that have a large number of local users for data space eligibility, such as tools disks and program libraries. However, since SFS does not support auxiliary directories, programs that require them, such as ASSEMBLE XF and OS/COBOL, cannot be placed in SFS directories.
- The directories should be updated infrequently.
- The updates should be confined to brief periods of time, preferably when concurrent access is low.

Virtual Storage Levels

Every virtual machine has its own primary address space for programs, data, and control information. Additionally, virtual machines may define other address spaces (data spaces) for data. Although not strictly correct, the primary address space is also called virtual storage for historical reasons.

VM/ESA is able to simulate virtual storages of up to 2047 MB. This is 1 MB less than the maximum virtual storage size implemented by the ESA/390 architecture (2048 MB or 2 GB). Data spaces can be defined with the 2-GB maximum size.

A guest operating system may simulate additional levels of virtual storage but the virtual storage simulated by VM/ESA always appears as the real storage to the guest operating system.

CP allows a guest operating system to use any page and segment sizes implemented in the architecture mode in which the guest runs. S/370 guests may use 2-KB or 4-KB pages and 64-KB or 1-MB segments in any combination. XA and ESA guests must use 4-KB pages and 1-MB segments.

When discussing guests and guest storage, the following definitions apply:

First-level storage The central storage in the real machine.

Second-level storage The virtual storage simulated by CP, defined in the CP user directory or through a CP DEFINE command. To the guest operating system, it appears to be real storage, and therefore it is also known as virtual-real storage.

Third-level storage The virtual storage simulated by a guest operating system, such as VSE or MVS, for its applications. It is the second-level storage to the guest operating system, and therefore it is also known as virtual-virtual storage. The operating system that runs in the virtual machine may define several address spaces. When a page of third-level storage is logically present in second-level storage, it may actually reside in first-level storage or in CP auxiliary storage.

First-level tables Segment tables, page tables, auxiliary storage tables, page status tables, and other information maintained by CP to map the second-level storage into real storage.

Second-level tables Analogous to the CP first-level tables, these tables are maintained by the guest operating system to map the third-level storage to the second-level storage.

The storage levels are illustrated in Figure 4. Note that when the VM/ESA guest is also a VM operating system, additional levels of storage are defined.

Because DAT (without SIE) can translate only one level of virtual address to a real address, VM/SP and VM/SP HPO have to build shadow tables, which are a combination of both guest tables and CP tables, but map the third-level storage into first-level storage. Shadow table maintenance causes considerable

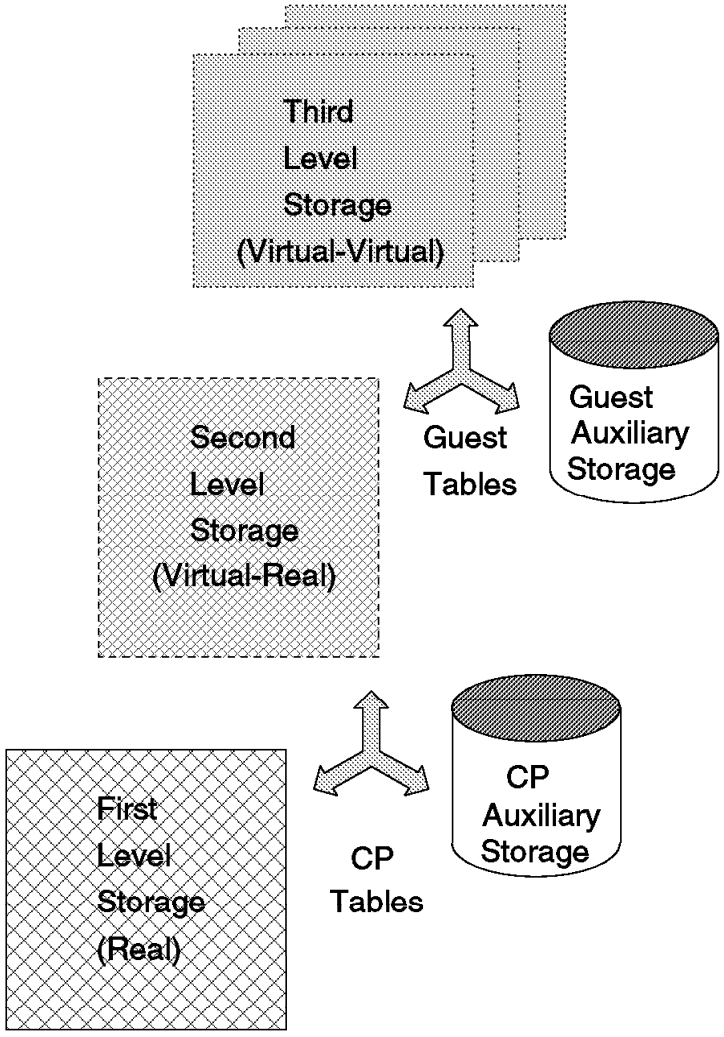

Figure 4. Storage Levels for V=V Virtual Machines

overhead; bypassing shadow tables is one of the major tuning considerations for operating systems (running in a V=V virtual machine) that create multiple address spaces.

Because the SIE instruction performs address translation for the virtual machine using all table levels, VM/ESA does not have to maintain any shadow tables to map third-level storage to first-level storage. Shadow table maintenance overhead has been eliminated by the interpretive-execution facility.

The SIE instruction translates virtual storage levels in steps. It translates a third-level virtual address to a second-level address using the second-level tables. When a segment or page exception occurs, the SIE instruction presents an interruption directly to the virtual machine operating system. The guest then pages in the required page and updates its tables.

In the next step, SIE translates the second-level virtual address to the first-level real address, presenting any segment or page translation exception to CP, which pages in the required page.

The multiple-level storage simulation may result in double paging, double spooling, double CCW translation, and other duplicate functions. Double paging is worse than it may seem at first because the guest and CP both use LRU algorithms for paging. Assume CP selects a page for page-out; some time later, the guest paging routines also select the same page for page-out, and CP must page it in to make it available for the guest page-out.

For important virtual machines, you can reduce this duplication in two ways:

- Defining the virtual machine as preferred

  CP does not page the second-level virtual storage for V=R and V=F preferred virtual machines. Only the third-level virtual storage is paged by the guest operating system directly into the V=R area. Other functions, such as the CP CCW translation, may also be avoided.

- Using guest handshaking functions

  This is selected by the VSE/ESA supervisor options MODE=VM or MODE=VMESA. Only VM/ESA executes paging functions, paging the second level of virtual storage into the real storage because the third and second levels of virtual storage coincide. The VSE/ESA supervisor does not perform paging functions or CCW translations.

  Make sure that the SET PAGEX ON command has been issued. This command should be issued automatically by the guest operating system at initialization time. The PAGEX handshaking function allows the virtual machine to dispatch a task while waiting for CP to satisfy a page-in request for another task. The operating systems that currently issue the SET PAGEX ON command are VSE/SP, VSE/ESA, and AIX/ESA.

  This mode may be combined with the preferred virtual machine option but, then, there would be no paging at all and the guest would require a large amount of central storage.

If none of the above alternatives are possible, you should consider one of the following options to avoid double paging:

- Define a V=V virtual machine with reserved or locked pages, thus minimizing CP paging activity on behalf of the virtual machine and transferring most of the paging load to the guest (see “LOCK and UNLOCK” on page 130 and “SET RESERVED” on page 127).
- Define a V=V virtual machine with a virtual storage (second-level storage) size close to the size of the third-level storage, thus transferring most of the paging load to CP. This option is mostly useful when the guest operating system uses one address space only. However, all the tasks of the guest are kept waiting during page faults, unless the guest supports the PAGEX option.

Double spooling may be avoided by dedicating the printers to the guest operating systems (eliminates CP spooling). This is practical only when there are enough printers to dedicate at least one to each guest with heavy printing requirements.

The second best alternative is having double spooling, as happens when VSE/POWER is controlling the printers in a virtual machine. Eliminating guest spooling (VSE printing directly to a virtual printer without VSE/POWER, for example) would increase the number of virtual I/O operations and would by far offset the benefits gained by avoiding the double spooling.

Auxiliary Storage

VM/ESA simulation of virtual storage relies on an efficient paging system between central storage and auxiliary storage. VM/ESA paging algorithms move 4-KB pages from auxiliary storage to central storage (page-in) when they are required. When the central storage is over-committed, VM/ESA moves pages from central storage to the auxiliary storage (page-out) to free some space.

Auxiliary storage can be located on DASD or expanded storage. Expanded storage, if available, is used by VM/ESA as the preferred paging space. DASD space is always required by virtual storage simulation; although not as efficient as expanded storage for paging, DASD can sustain reasonable paging rates.

The total virtual storage size is one of the factors that determine the requirements for central storage, expanded storage, and DASD. The proper balance between the amount of virtual storage and auxiliary DASD storage is a key factor in VM performance.

Adding all virtual storages (primary address spaces and data spaces) required by all simultaneously logged-on users may be a first estimate to determine DASD storage space. However, keep in mind that:

- CP also uses virtual storage for pageable modules, buffers (account, monitor, spool), and virtual disks, for example.
- Shared segments that are loaded above the user's virtual storage size increase the user's virtual storage requirements. However, only one copy of the shared segments uses paging space.
- Virtual pages that were not used do not require auxiliary storage space.
- CMS frequently releases the virtual storage pages currently not in use, decreasing the page space required.
- Block paging efficiency depends on finding sufficient contiguous blocks on DASD to form large blocks; that is, block paging requires more DASD space allocated for paging than the space required to accommodate the user working sets and CP requirements.

Therefore, a better approach is to start with two or three times the sum of all working set sizes as a first estimate, and monitor paging activity, paging contention, storage contention, and block paging to adjust the DASD paging areas until you achieve an optimum size and distribution. Here are some guidelines:

- Do not allocate other active areas in the same volume.

  The DASD volumes should be dedicated for paging; they should not be used for data files or spooling unless this extra activity is low (around 5% in terms of I/O rate).

  For page-outs, VM/ESA selects first the areas allocated as PAGE. When the PAGE space is exhausted, the excess paging activity overflows to the spool (SPOL) areas. If all areas are full (PAGE and SPOL), the system is not able to continue resulting in a CP abend.

  It is possible to use the DASD space allocated as SPOL for both paging and spooling. If the spooling activity is much less than the paging activity and does not affect response times, you may allocate all areas as SPOL. However, a spool-full condition would also result in a CP abend in this case.

When the spooling activity is high, however, you should confine spooling activity to separate volumes. Using DASD volumes just for paging, with only one area allocated as PAGE in each volume, prevents the spooling and data I/O activities from interfering with the paging activity.

- Do not allocate more than one paging area in any volume.

  Regardless of the alternative selected (as discussed in the previous item), do not allocate more than one PAGE or SPOL area in the same volume.

- Check the volume layout to optimize the average seek length.

  Check the VMPRF seek report and group the most active areas close to the paging area, with the paging area in the center of the volume.

- Spread the paging areas over the storage controls and channels.

  The paging flow between central storage and the paging DASD areas should be facilitated by multiple channel paths connected to the DASD storage controls. For highest paging efficiency, the DASD space allocated for paging should be spread among several volumes and paths.

- Make the paging areas in all volumes approximately the same size; otherwise, smaller areas will be exhausted first, thus reducing the number of paging volumes.

- Look at the I/O rate (not paging rate) and paging contentions to determine whether more paging areas are necessary.

- Define sufficient paging space to have a large block paging factor.

  The allocation of extra paging space allows VM/ESA to find contiguous DASD slots to page out several pages with only one I/O operation, improving block paging efficiency.

- Avoid mixing different device types for paging areas; if the paging rate is moderate to high, faster devices are selected more often with a consequent higher percentage of slots in use and smaller blocking factors.

Figure 5 shows an example of a VMPRF PRF088 report with paging and spooling activities. You can see that a high PCT_PAGE_IN_USE (devices X'0A22' and X'0A23') normally are responsible for lower block page sizes.

```
PRF088  Run 09/08/1993 10:44:53        DASD_SYSTEM_AREAS                                                    Page   12
                                 DASD System Areas by Type -- Paging and Spooling Activity
From 09/08/1993 09:52:16                                                                          GG243934
To   09/08/1993 10:42:16                                                                          CPU 9121     SN  20415
For   3000 Secs 00:50:00             VM/ESA STORAGE MANAGEMENT MEASUREMENTS                            VM/ESA   21.01 SLU 9303
_____________________________________________________________________________________________________________________

<-----Device----->       <---------Slots--------> <------------------Rate----------------->

                                   Pct   Pct                              Total            Serv MLOAD Block PctTimes
Num- Volume       Control      Avail-  Page Spool  Page  Page Spool Spool      Wthout  SSCH  Queue  Time  Resp  Page  Used by
 ber Serial Type   Unit    Type    able InUse InUse  Read Write  Read Write Total S+RSCH +RSCH Length /Page  Time  Size Allocatn

0A20 3934RS 3380-D 3880-23 Page   37050  29.6    0   2.9   6.0    0    0   9.0   7.3   1.6  0.580   7.0   6.4    13     25.5
0A22 3934V1 3380-D 3880-23 Page   28800  51.0    0   4.7   8.8    0    0  13.5  11.7   1.8  0.160   5.4   6.3     4     40.3
0A23 3934V2 3380-D 3880-23 Page   33000  53.0    0   5.5  10.1    0    0  15.6  13.7   2.0  0.660   5.0   5.0     3     45.5
0A24 3934V3 3380-D 3880-23 Page   33000  46.0    0   4.7   8.9    0    0  13.6  11.9   1.8  0.660   5.0   5.2    13     40.3
0A26 3934V5 3380-D 3880-23 Page   62850  26.7    0   5.3  10.0    0    0  15.3  13.4   2.1  0.400   4.8   4.9    12     46.1

096D SET1C1 3390-2 3990-3  Spool  36000     0   7.7   0.4     0   0.1   0.7   1.1     0   1.2      0  11.2  10.8     0    14.2
096F SET2C1 3390-2 3990-3  Spool  36000     0   7.7   0.4   0.0   0.1   0.4   0.9     0   1.0      0  11.2  11.1     0    22.3
0DAD SET3C1 3390-3 3990-3  Spool  36000     0   6.2     0   0.0   0.1   0.9   1.1     0   1.4      0  10.8  10.8     0    25.9
```

Figure 5. VMPRF PRF088 Report

The rightmost five variables in Figure 5 (average queue length, service time per page, response time (MLOAD), block page size, and percentage of time used), provide valuable information about the performance of the paging and spooling activities.

VM/ESA selects a volume with an MLOAD less than the average system MLOAD in such a way that, in the long run, the allocation is evenly spread among the paging volumes, as you can see by the variable that indicates the percentage of time each volume was used by allocation.

Figure 6 shows an example of a VMPRF PRF004 report that can help you estimate the effectiveness of the paging subsystem.

```
PRF004  Run 09/09/1993 08:59:46        PAGING_BY_TIME                                                    Page   11
                                  Paging Overview by Time
From 09/09/1993 07:41:02                                                                  GG243934
To   09/09/1993 08:51:02                                                                  CPU 9121     SN  20415
For   4200 Secs 01:10:00              VM/ESA STORAGE MANAGEMENT MEASUREMENTS                    VM/ESA   21.01 SLU 9303
_____________________________________________________________________________________________________________________

          <--------------Expanded Storage--------------> <----------Real Storage-----------> <------------DASD Rate----------->
                                                                                                          <Single Reads>
                         Pct              Est.          DPA     Non Resident  Mean  Est.
From  To    Paging  PGIN Fast-  PGOUT  Total    ES   Migr Pagable Pagable   Shared Avail  Page                    Gu-
Time  Time  Blocks  Rate  Path   Rate   Rate  Life   Rate   Pages   Pages    Pages  List  Life Read Writ  Tot Shrd  est Syst  Tot

   ........................
08:00 08:01  61440   277  93.1    393    669   156    100   13129    2739     620   289    28   62   81  143    4    2    1    3
08:01 08:02  61801   258  91.9    380    638   163    104   13125    2743     617   183    30   76   55  130    6    2    2    4
08:02 08:03  61801   220  92.5    350    570   177    110   13122    2746     629   194    33   97   44  141    9    8    1   10
08:03 08:04  61801   242  92.0    389    631   159    138   13119    2749     655   338    30  107   56  162    5    4    2    6
   ........................
08:46 08:47  61816   223  92.4    306    529   202     68   13066    2802     638   338    37   56   51  107    6    8    2   10
08:47 08:48  62010   230  93.0    321    551   193     77   13059    2809     640   117    35   49   53  101    5    6    2    8
08:48 08:49  62010   183  93.0    260    443   238     64   13022    2846     654    85    42   54   50  104    4    7    2    9
08:49 08:50  62010   230  93.5    277    508   224     41   13064    2804     662   230    43   43   28   70    4    6    2    8
08:50 08:51  62010   258  92.8    359    617   173     77   13059    2809     650   150    30   61   77  138    5    6    2    7

07:41 08:51  61735   240  92.4    369    609   167    101   13244    2624     635   713    29   68   82  149    7    7    2    8
```

Figure 6. VMPRF PRF004 Report

The EST._PAGE_LIFE (in real storage and expanded storage) variables should be at least the average command cycle time (average think time plus average response time, reported by VMPRF report PRF033). Two times the average command cycle time is an optimum value.

Note that the same rules regarding minimum and optimum lifetimes in expanded storage apply to paging and minidisk cache. If you use expanded storage for paging, the lifetimes should be approximately the same for both paging and minidisk cache. If you do not achieve the desired page lifetimes, the amount of storage needed is:

$$\text{Needed storage} = \text{current storage} \times \frac{\text{desired lifetime}}{\text{actual lifetime}}$$

You should also look at the VMPRF PRF019 report and check the amount of time spent waiting for paging. If paging wait is higher than 10%, you need more expanded storage for paging. Otherwise, you need more expanded storage for minidisk cache which will probably be reflected in the amount of time spent on instruction simulation (DIAGNOSE I/O).

The RTM SYSDASD screen is also quite useful to help you isolate the paging load for actual system paging and other minidisk activities, such as MDC and minidisk mapping.

Chapter 3. Central Storage

The amount of central storage that you need may consist of the following:

- Preferred virtual machines requirements
  - Sum of the storage sizes of V=R and V=F virtual machines
  - V=R recovery area
- RIO370 Area
- CP requirements
  - Resident CP nucleus
  - Pageable CP nucleus working set
  - Other CP areas fixed in central storage
- Dynamic paging area (DPA)
  - Sum of the working sets of the V=V virtual machines
  - Free storage required by CP to manage V=V and V=F virtual machines

After estimating the central storage requirements, you should allow for future growth.

This chapter provides you with the basic information to understand and plan for central storage requirements.

Central Storage Layout

At system initialization, CP dynamically senses how much central storage is available and builds the corresponding storage management tables. A page frame table consisting of 16-byte entries, one entry for each 4-KB real frame, is built on top of the real storage. The size of the frame table is, therefore, 1/256 of the central storage size.

However, if you want to use less central storage than is available, you can either specify the STORE parameter during the IPL process or code the SYSSTORE macro in the HCPSYS system file.

In any case, CP creates the page frame table with entries describing only the central storage size that is made available to the system; in other words, no storage is wasted on unnecessary control blocks.

All other parameters previously found in SYSSTORE, such as RIO370, TRACE, V=R, and V=R_FREE, can be specified through the STORAGE statement of the system configuration file. This method should be preferred for coding these parameters, thus eliminating the need for CP nucleus generation in case of changes in the central storage layout. Therefore, you should not code the SYSSTORE macro in HCPSYS.

Figure 7 shows the layout of the central storage in VM/ESA when all the optional areas are used. If you create the CP nucleus as a relocatable module, you can load it at any storage location (see “Resident CP Nucleus” on page 28).

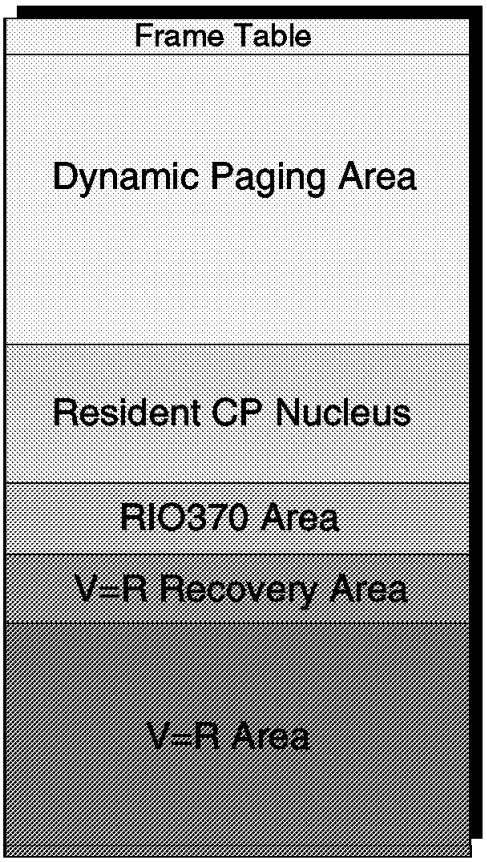

Figure 7. VM/ESA Central Storage Layout

V=R Area

The optional V=R area is reserved for preferred virtual machines. Preferred virtual machines are the machines elected to run in the V=R area. They must have an OPTION V=R or V=F statement coded in their directory entries.

Storage for preferred virtual machines is not paged by CP and their dedicated I/O devices are eligible for the SIE Assist function. VM/ESA makes use of the PR/SM hardware function to run more than one preferred virtual machine. The preferred virtual machines may reach near-to-native performance levels when other virtual machines do not drain real resources required by them and when complex VM/ESA services are not used.

Only one V=R virtual machine may be active. Its storage is allocated starting at real storage page frame 0. The PR/SM feature is not required to run one V=R preferred virtual machine. By default, VM/ESA dedicates real processors to the V=R virtual machine virtual processors. You may avoid this default through the UNDEDICATE command or the OPTION NODEDICATE directory option.

Up to five V=F virtual machines may be active concurrently with the V=R machine. Their storages are allocated from the top of the V=R area, below the V=R recovery area. V=F machines require the PR/SM hardware function.

The storage requirements of the preferred virtual machines should be estimated with their internal paging rates in mind. A large V=R area may lead to high paging rates of the V=V virtual machines by CP. A small V=R area, on the other hand, may lead to high paging rates inside the preferred virtual machines.

The guideline is to balance the paging rates between the preferred virtual machines and CP, as a starting point. Then, you may favor one preferred virtual machine by allocating more storage to it, and thus reduce its internal page rate. Or, you may favor the V=V virtual machines, which are paged by CP, by allocating more storage to the DPA.

V=R Recovery Area

VM/ESA builds the V=R virtual machine control blocks into the V=R recovery area. These control blocks are used to define the virtual machine's virtual processors, virtual I/O configuration, and current activities, and are used to restore the V=R machine operating environment in case of a failure of the VM/ESA control program.

If this area is not defined or if its size is not big enough, the V=R virtual machine is not restarted after a VM/ESA failure. Usually, 1 MB is enough for the preferred guest recovery function (the minimum is one page), but it depends mostly on the number of devices used by the V=R guest. If the VRFREE area is over-estimated, the excess is not used for any other purpose, and is wasted.

VM/ESA displays a message at the operator console and at the virtual machine console when the V=R recovery area overflows to the normal free storage.

RIO370 Area

The RIO370 area is used to increase the performance of I/O operations issued through DIAGNOSE X'98' channel programs in S/370 architecture format. The DIAGNOSE X'98' function reduces the CP overhead associated with the translation of the CCWs related to these I/O operations. VTAM, PVM, and TCP/IP are examples of programs that use the DIAGNOSE X'98' function. The OPTION DIAG98 directory statement authorizes a V=V virtual machine to request RIO370 or DPA page frames for DIAGNOSE X'98' channel programs.

The RIO370 area is not required by 31-bit addressing mode virtual machines. VTAM Version 3 Release 3.0 or above (5684-095), running under GCS in an XA virtual machine, may issue DIAGNOSE X'98' instructions without requiring an RIO370 area. Page frames required by 31-bit addressing mode virtual machines are taken from the DPA.

The RIO370 area is allocated below the CP nucleus and above the V=R area, or starting at real page frame 1 if there is no V=R area. You may use the QUERY FRAMES command to adjust the size of the RIO370 area. Usually, VTAM requires less than 1 MB, but you may have problems in starting VTAM if the RIO370 area is too small. In that case, just remove the OPTION DIAG98 from the directory entry until the system can be IPLed with a larger RIO370 area.

The RIO370 area may conflict with the definition of preferred virtual machines because the RIO370 area must be allocated below the 16-MB line in order to execute S/370 format CCWs. This reduces the size of the V=R area to less than 16 MB. In this case, you must choose between using DIAG98 or extending the V=R area to alleviate preferred guest internal paging.

Resident CP Nucleus

Part of the CP nucleus is resident in central storage at all times; the other part is paged in when necessary.

The resident part of the CP nucleus contains the most frequently used routines and areas, as well as key paging routines and tables that cannot be paged; the HCPCPE module marks the end of the resident nucleus in the list of CP modules. You can determine the size of resident nucleus using the CP LOCATE HCPCPE command or the CP QUERY FRAMES command.

The pageable CP nucleus contains less-frequently-used routines such as the logon routines. Most of the pageable CP routines are reentrant and, therefore, they do not have to be paged-out; CP refreshes them (page-in) only when necessary. A pageable routine that is frequently used can be moved from the pageable part to the resident part of the CP nucleus by moving the required routine before the HCPCPE entry in the CP load list.

The resident CP nucleus requires approximately 1.6 MB plus the page frame table and the real I/O device blocks. A page frame table entry is 16 bytes long; therefore, 256 MB of central storage are mapped by a 1 MB page frame table, for example. The real I/O device blocks require little resident nucleus storage; for example, 300 devices require less than 150 KB.

The total size of the CP nucleus ranges from 3 MB to 4 MB. A detailed method of calculating the total size of the CP nucleus can be found in *Planning and Administration*. It can be verified by using the CP QUERY FRAMES command.

A good approach is to always generate CP nucleus as a relocatable module and load it high enough in central storage; just remember to allow space enough for the page frame table. Then, you can change the V=R area, for example, just changing the system configuration file.

Dynamic Paging Area

The amount of virtual storage is the main factor in determining the DPA requirements. As DPA becomes over-committed, you should either increase central storage or increase the paging capability. The dynamic paging area contains page frames related to:

- Virtual storages of the V=V virtual machines
- System virtual address space
- Buffers and control blocks

The storage keys used for fetch and write protection of the virtual storage of the virtual machines are controlled by the virtual machine operating systems.

CP monitors the storage key settings and saves them into the page status tables of the virtual machines in order to preserve the settings during the paging process. When the virtual page is paged into a real page frame, CP sets the real storage key according to the user's virtual page key.

The storage key facility (SKF) hardware function reduces the CP overhead associated to the storage key instructions simulation. The SKF is a standard function available in all processors supported by VM/ESA except the IBM ES/4381 Models 9xE. Therefore, VM/ESA is expected to have a slight additional processor overhead when running on IBM ES/4381 processors.

System Virtual Address Space

The VM/ESA control program (CP) implements virtual storage for its own use, as in the example shown in Figure 8.

The size of the system virtual address space is 2 GB. The low addresses of the system virtual address space, up to the end of the resident CP nucleus (HCPCPE), are not pageable. These virtual addresses map directly to real storage addresses.

Higher virtual addresses, starting with the pageable CP nucleus, are pageable in the same way as the virtual addresses of virtual machines. These pages are brought to central storage on demand and are locked in central storage when in use. Each time a pageable CP routine is called, a lock count is incremented by one; each time execution of a pageable CP routine is completed, the lock count is decremented by one. When the lock count reaches zero, the corresponding page frames are unlocked and compete with virtual machines pages under the control of the normal page replacement algorithm.

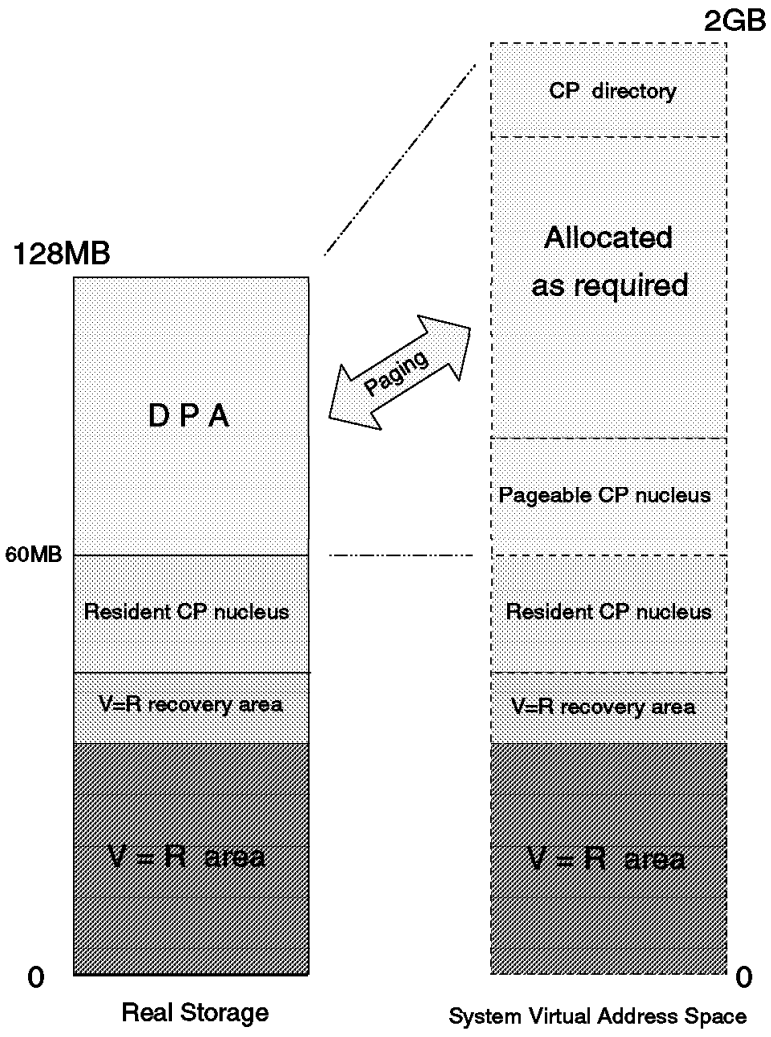

Figure 8. System Virtual Address Space

During system initialization, all pageable CP routines are read into central storage and forced to be paged out to the system paging areas. As the pageable CP routines are reentrant, hence never modified, the initial copy in the paging areas is always current and can be paged in whenever necessary. In this way, it is always possible to write a new CP nucleus while the existing nucleus is also online, even if you are not using a CP nucleus module.

The virtual addresses at the top of the system virtual address space are preallocated to the CP user directory. When the user directory is brought online, these virtual addresses are then used to hold the entire directory. This operation provides a highly efficient read-only access to the directory by commands such as LOGON, LINK, and SPOOL. The DASD copy of the CP directory must still be accessed because of changes made in the directory.

This virtual directory is built only for the first contiguous area allocated as DRCT. References to directory pages in any remaining areas require the allocation of available pages from the system virtual address space and the corresponding page-in operations. Therefore, allocating more than one DRCT area is not recommended.

The rest of the system virtual address space is allocated as needed. The system virtual address space pages include:

- Pageable CP modules.
- CP User Directory.
- Large CP control blocks that require one full page or more, such as virtual machine definition blocks (VMDBK).
- Free storage.
- Spool file buffers.
- System data file buffers, such as monitor buffers.
- Buffers for migrating expanded storage to DASD.
- Prefix pages for the real processors prefixed storage areas (PSAs).
- CP Internal Trace Tables.
- DIAGNOSE X'98' frames requested by 31-bit mode V=V virtual machines.

Free Storage

Free storage contains most CP control blocks and work areas. Most control blocks are allocated in the free storage. Some others reside in the CP nucleus. Some large (full-page) control blocks are taken directly from the DPA.

Smaller control blocks are sub-allocated from the free storage pages that are allocated from the DPA. These free storage control blocks contain, for example:

- CCWs for paging I/O operations
- Translated CCWs for data I/O operations
- I/O request blocks
- I/O error blocks
- Terminal I/O message buffers
- Accounting records

Trace Table

The CP internal trace table is a wrap-around sequence of entries normally used to debug CP failures. The CP internal trace table is taken from the DPA; the minimum size is three pages.

The trace table size defaults to one page per processor for each 64 pages of central storage, excluding the V=R area. This represents 6% of the DPA storage on a 4-way processor; that is, a 4-way processor with 256 MB of central storage would have 16 MB allocated to the trace table by default.

You should override the system default and define the trace table with 16 pages per processor, by specifying TRACE=16 in the system configuration file or in HCPSYS.

Saved Segments

A saved segment is a range of virtual storage pages defined to hold reentrant programs. Defining frequently used programs as saved segments saves central storage because many users can share the same real storage frames. As a consequence, the use of saved segments reduces the contention for central storage and the paging activity.

Saved segments are a key facility in central storage management. Without saved segments, 200 CMS users would require 600 MB of pageable virtual storage just to hold 200 copies of the CMS nucleus. This occurs when users IPL 190 instead of IPL CMS.

Using the CMS named saved system (NSS), VM/ESA pages only one copy of the reentrant part of the CMS nucleus for all users. Besides, when several virtual machines use the same code, references to the shared pages are made quite often, decreasing the probability of stealing. Therefore, the shared pages have a higher probability of remaining in central storage.

Most multiple-user programs provide a saved segment installation option. The read-only reentrant part of the program code is saved in a shared segment and invoked by the nonshared part of the program. Examples include GDDM, APL2, VM/AS, VSAM, VTAM, and OV/VM, among many others.

Several VM/ESA components, such as CMS, GCS, CSL, and CMSINST, are designed to take advantage of this facility. CMS also allows you to have minidisk directories in shared segments (the Y- and S-disk directories are saved in the CMS nucleus); however, if the disk is modified, the shared segment is no longer used.

Figure 9 shows the implementation of shared segments by VM/ESA. Basically, each virtual machine has a virtual storage mapped by a segment table. Each segment table entry represents 1 MB of virtual storage and points to a page table. All segment table entries that represent the shared segment point to the same page table. Since the page tables map the virtual addresses to real addresses, only one copy of the shared segment and page tables is maintained.

Saved segments are also used for nonshared areas, but in this case each virtual machine has its own private copy of the nonshared segments.

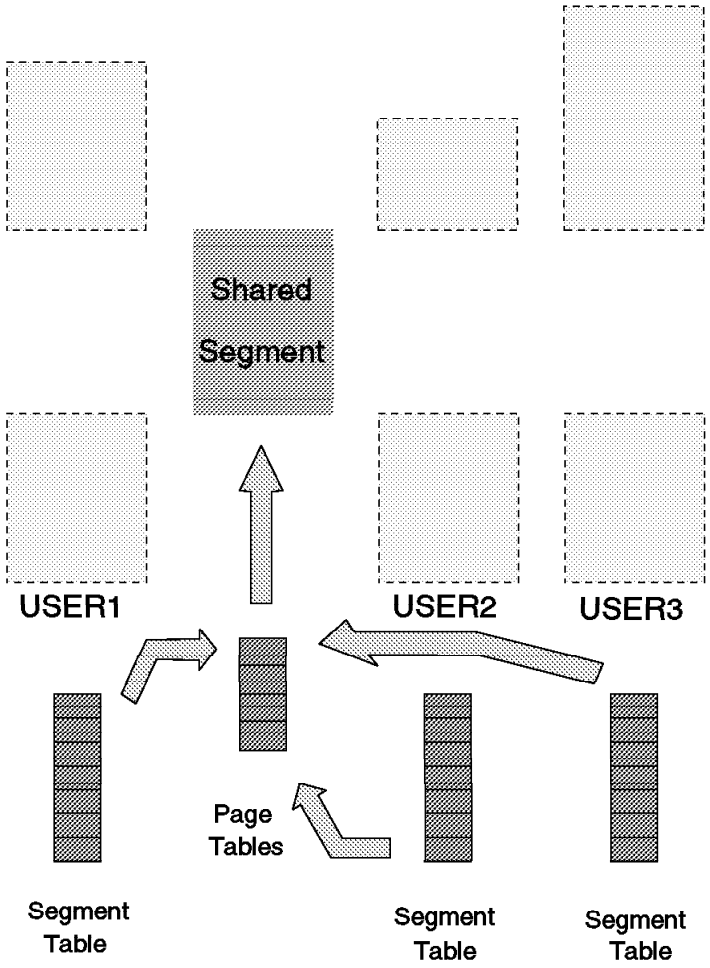

Figure 9. Saved Segments Virtual Storage Structure

Working Sets

The working set of a virtual machine is a projection of the number of pages that should be resident in central storage in order to process the virtual machine's transactions efficiently; that is, with a minimum number of page faults.

The working set size (WSS) is based on the virtual machine's run history and is calculated each time the virtual machine is dropped from the dispatch list.

If VM/ESA is able to keep a user's working set resident in central storage, then this user should experience only a small paging wait delay. If central storage is not available to hold the sum of all users' WSSs, then the paging rate increases and the users may experience some paging wait time.

Included in a user's working set are the control blocks CP builds to preserve the virtual machine environment. These blocks are used to describe the virtual machine configuration, to save its status, and to contain the segment and page tables. Typically, about 30 KB are required by each virtual machine for these control blocks. The INDICATE USER command shows the current WSS for one user.

V=V virtual machines with up to 32 MB of virtual storage have their segment tables in their VMDBK control block. V=V virtual machines with more than 32 MB require a separate real storage page to contain the segment table. V=V virtual machines with more than 1 GB of virtual storage require two consecutive real storage page frames to contain the segment table.

When a virtual machine attaches a saved segment, CP may have to extend the segment table to describe the storage used by the saved segment. VM/ESA keeps this extended virtual storage until the virtual machine logs off. In order to keep most of the segment table in the VMDBK, try defining the most used segments below 32 MB; if that is not possible, defining a segment at 33 MB or 999 MB would use the same amount of real storage for the segment table.

VM/ESA assumes that most programs have a locality of reference characteristic, so that the virtual storage references in the next interval are likely to occur in the same pages as in the previous interval. In other words, the configuration of the referenced pages usually changes slowly.

In fact, the total amount of virtual storage a program uses is not nearly as significant a performance factor as is the way in which virtual storage is referenced. That is, the pattern and frequency of references to pages in a program have more effect on the number of page faults that occur than the total size of the program.

A well-structured program keeps its references to the pages of the current executing subroutine, and paging activity occurs only when the program progresses from one subroutine to the next.

Paging System

In a well-dimensioned and tuned system, paging activity should not impact the response times. When response times are not good, paging waits are often the largest component of the delays.

Excessive commitment of central storage may lead to a situation where the system spends most of the time performing paging activities, and the users spend most of their time in page wait status, experiencing unacceptable response times; this is called thrashing.

Tuning Guidelines

A reasonable guideline is to keep the average paging wait time below 10% of the transaction response time. A similar guideline is to keep less than 10% of the active users in page wait status. When paging wait time is high, you may:

- Increase central storage size and thus reduce the paging rate
  - Reduce the V=R area to increase the DPA, if possible
- Increase the paging capability
  - Add expanded storage
  - Add paging DASD devices and paths
- Discriminate against high paging V=V virtual machines
  - Use the SET SRM LDUBUF command
  - Use the SET SRM STORBUF command
  - Use the SET SRM MAXWSS command

Block Paging

Block paging is a technique implemented by VM/ESA to maximize paging subsystem efficiency. Figure 10 illustrates the main concepts associated with the block page technique.

The goals of the block paging function are:

- Move several pages with only one I/O operation to DASD.
- Keep together as blocks the most frequently used pages of the user's working set.
- Reduce the probability of page faults by bringing to central storage blocks of related pages.

Page fault interruptions are resolved by page-in operations. VM/ESA must allocate a real storage page frame before starting a page-in operation. Real storage page frame requests are satisfied from the available list, which is a repository of available page frames in the DPA.

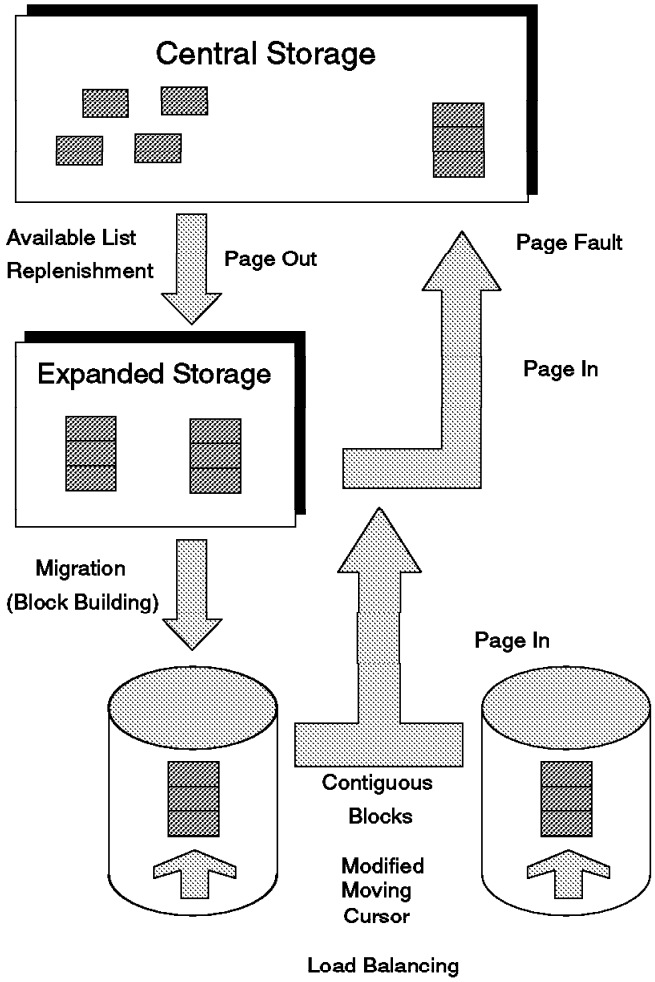

Figure 10. Block Paging

CP keeps a list of available pages for immediate use; pages are placed in the list when a user logs off, for example. When the number of pages in this available list drops below a threshold, CP replenishes it by paging-out some least active pages.

Expanded storage, when available, is the preferred paging storage resource. Pages are moved to expanded storage by the available list replenishment routines and brought back to central storage on demand. A page can reside either in central storage or in expanded storage, but not in both. However, if the page was not changed, it might reside in both expanded storage and DASD at the same time.

When expanded storage is not available or when it is full, pages are written to DASD. VM/ESA does not use a DASD paging hierarchy based on device type. The device selection routine selects the device with the best response time at the moment (MLOAD variable shown in Figure 5 on page 22), balancing paging activity among all paging devices.

CP uses the Resume Subchannel (RSCH) instruction together with suspend bits in the ORB and paging channel program in order to continuously chain individual paging requests to active paging channel programs.

After a DASD volume has been selected, VM/ESA uses a modified moving cursor technique to allocate slots inside the PAGE extents. When all the extents allocated as PAGE are full, VM/ESA starts using the extents allocated as SPOL. The operator is notified when 90% of the PAGE or SPOL space is allocated. Once the space is exhausted, VM/ESA terminates abnormally.

The allocation scheme locates, ahead of time, groups of contiguous available slots for both paging and spooling space. For paging, an appropriate size is any group larger than two; anything less is considered a crumb and is ignored. When a request for slots arrives, the groups of slots are searched, and the one with the best fit is selected. Only those slots from the group that are needed to fill the request are used.

The block size ranges from two to a high limit of 64 pages per block. For each DASD type, the maximum block formed is further limited in such a way that the path is not tied up for more than 60 milliseconds. Therefore, the maximum block size is the smaller of 64 or the following value:

$$max_{devtype} = \frac{pages/track}{revolution\ time} \times 60$$

The fraction evaluates the number of milliseconds necessary to transfer one page and, when multiplied by 60, yields the number of pages that can be transferred in 60 milliseconds.

The maximum block size for some IBM DASD types are:

$$max_{3380} = \frac{10}{16.6} \times 60 \quad 36$$

$$max_{3390} = \frac{12}{14.1} \times 60 \quad 51$$

$$max_{9345} = \frac{10}{11.2} \times 60 \quad 53$$

Page blocks are formed according to the following rules:

- Blocks are not created for any CP-owned virtual address space.
- Blocks can span more than one device.
- For saved segments, blocks are formed only for pages that reside in the same 1-MB virtual segment.
- For other address spaces, blocks are formed across the entire address space, with a maximum of 64 pages per block.

Allocating more DASD space than required by the total virtual storage size allows VM/ESA to find large contiguous sequences of slots and maintain good paging efficiency; otherwise, CP has to split a block in smaller blocks, thus increasing the processor overhead and decreasing the paging I/O efficiency.

Check the block paging efficiency by analyzing the BKRD and BKMG variables in the SYSTEM report provided by RTM VM/ESA, and the BLOCK PAGE SIZE variable in the PRF088 report (DASD SYSTEM AREAS) provided by VMPRF.

A DASD slot is allocated to a page the first time the page must be paged out. A nonshared page that is currently assigned a slot on a paging device is reassigned a new slot each time it must be paged out, and the slot previously allocated is made available for reassignment. A shared page is paged out only once and, therefore, always resides in the same slot.

A block that has been read from DASD is transferred in its entirety to expanded storage. Even unreferenced pages are moved to expanded storage to allow more time for the page to be referenced again.

The expanded storage migration routine does not migrate unreferenced blocks to DASD because the DASD copy is still valid. They are simply released without being rewritten to DASD.

Referenced pages are written to DASD even if they have not been changed. In this way, the referenced pages, which represent the virtual machine's new working set, are placed in one or more contiguous blocks and can be quickly read when required. Unreferenced pages are dropped off the new working set. As the blocks are restructured, the unreferenced pages are left in individual slots on DASD.

When a page fault occurs, VM/ESA checks whether the page is in expanded storage or on DASD. Page in from expanded storage is done on demand, one page at a time, while page in from DASD is done in blocks unless the page is in an individual slot (not part of a block).

In order to resolve the page fault as quickly as possible, the read operation is performed in two parts:

- The first part starts at the slot that caused the page fault and continues to the end of the block. As soon as the first page is read in, the page entry is validated (so the virtual machine can be dispatched), and the channel program continues to the end of the block.
- The second part starts at the first slot of the block and continues to the slot that immediately precedes the page that caused the page fault, and is validated asynchronously.

Chapter 4. Expanded Storage

Expanded storage is currently available on ES/3090, ES/4381-9xE, and ES/9000 processors. On some processors the total storage can be configured to be either central storage or expanded storage.

There is no requirement to change any system generation parameters to inform CP that expanded storage is available. During the IPL of CP, the availability of expanded storage is detected and is automatically brought on line. Once on line, it cannot be varied off; however, if you have to reduce the amount of expanded storage available to CP, the excess can be attached to any virtual machine in the system.

Expanded storage can be used as either a fast electronic paging device or as a data cache to reduce I/O activity to disk devices. In any case, it reduces the time that a virtual machine has to wait to finish its transaction.

Expanded storage can greatly reduce the I/O activity on a VM/ESA system by keeping pages of virtual storage or data in an electronic media. Frames of 4 KB of data are transferred between central storage and expanded storage at much faster rates than can be sustained by I/O devices.

Expanded storage is dynamically configured into a paging area and a data cache area by the CP arbiter routine. The arbiter monitors the system activity and computes the ratio which optimizes system throughput.

Configuration

Expanded storage can be partitioned for use by CP and any number of guests. Since expanded storage can be specified in the virtual machine directory entry or through CP commands, the way expanded storage is allocated to the various guests depend on the order they log on to the system.

CP and any V=R, V=F, and V=V guests can have expanded storage dedicated. CP uses whatever part of expanded storage is not allocated specifically to a guest until such a request is made, either implicitly through a guest log on, or explicitly through a CP command.

Expanded storage is attached in increments of 1 MB; Figure 11 illustrates how expanded storage may be partitioned.

A request to retain expanded storage for CP use (CP RETAIN command) can be satisfied by multiple discontiguous areas of expanded storage if necessary. If a request to retain expanded storage for CP use is for a larger amount than is currently available (because it is in use by one or more virtual machines), CP retains whatever expanded storage is available and sets a pending request for the remainder. The pending request is satisfied when a sufficient amount of expanded storage is detached from one or more virtual machines.

After the CP partition has been defined, CP shares the partition between paging and minidisk caching automatically and dynamically. However, if desired, the CP RETAIN command can be used to set lower and upper boundaries on the size of the area within the partition used for minidisk caching.

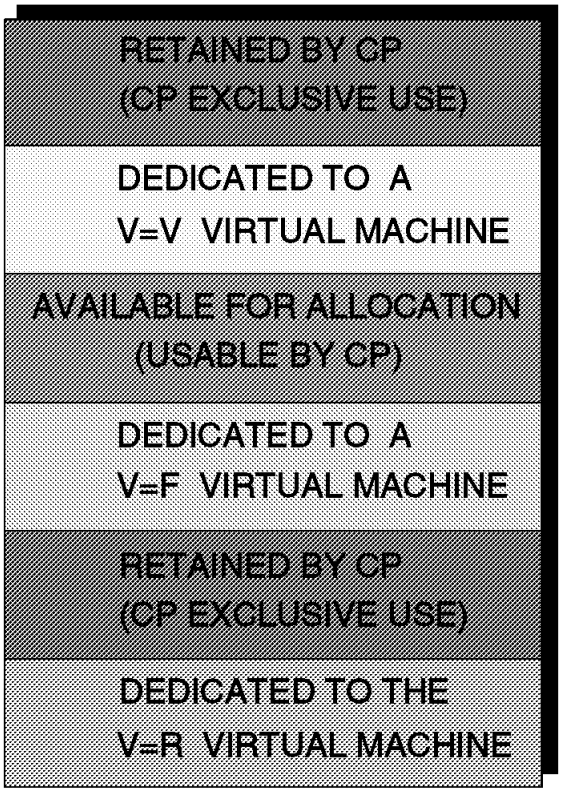

Figure 11. Expanded Storage Allocation

Allocation and Deallocation

CP expanded storage allocation routines use a bit map and a lookaside buffer. The bit map describes each expanded storage block and the lookaside buffer is a simple push-down stack with the block numbers of formerly-released blocks. The use of this lookaside buffer can considerably expedite the allocation of expanded storage blocks.

To allocate an expanded storage block, CP examines the lookaside buffer. If an entry is available, it is removed from the buffer, and the block number is returned to the caller routine. Blocks in the lookaside buffer are already marked as allocated in the bit map, and can quickly be given to the allocation routine. If no entries are available in the lookaside buffer, CP searches the bit map looking for a zero bit, indicating a block that has been previously deallocated.

On a page fault for a page in expanded storage, CP moves this page back into central storage, and places its block number into the next table entry of the allocation buffer. Although not assigned to a user, the corresponding block allocation bit in the bit map is not altered.

When all entries of the lookaside buffer contain a block number, CP already has sufficient expanded storage blocks for allocation. In this case, a released block is just marked as deallocated by resetting to zero its corresponding bit in the bit map.

Page Migration

During the allocation process, CP checks the total number of available blocks. If this number is below the low threshold value, CP starts the expanded storage migration task.

This task replenishes the available blocks by moving expanded storage blocks to DASD until the number of available blocks has reached the high threshold value. These two threshold values are dynamically adjusted to achieve balanced system operation.

The page migration routine is also called to move CP allocated expanded storage blocks (user virtual pages and minidisk cache blocks) when attaching expanded storage to a guest.

There are no direct paths between the channel subsystem and expanded storage. To migrate the contents of expanded storage blocks to DASD, the migration routine temporarily transfers the blocks to central storage. One page at a time is transferred because the movement of a page between expanded storage and central storage is synchronous. From central storage, the migration routine moves the blocks to DASD using block paging, as shown in Figure 12.

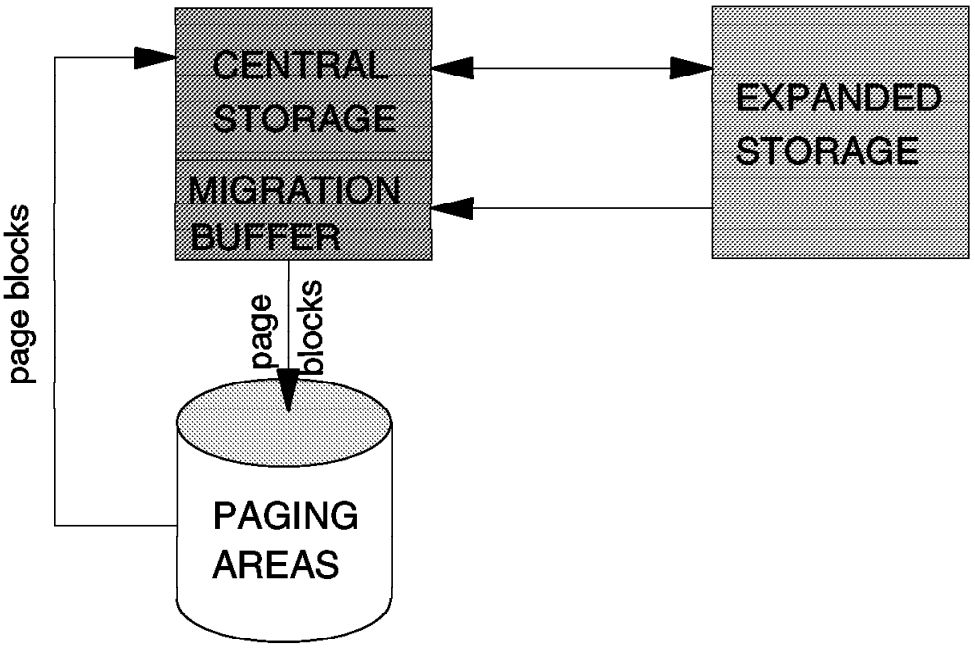

Figure 12. Page Migration

For each expanded storage block, CP keeps a time stamp that indicates the time the corresponding page frame was moved to expanded storage. During expanded storage migration, CP migrates those blocks that are older than the average age of all expanded storage blocks.

Blocks are normally taken from user private space, but shared segments and shared address spaces (including the system virtual address space) may also be taken if not enough blocks are found during the first scan.

Partitioning

The CP RETAIN XSTORE command is used to control how much expanded storage is reserved exclusively for CP use. Expanded storage that is retained for CP use cannot be attached to a virtual machine

If the CP RETAIN command is not used, CP shares the expanded storage available for use between paging and minidisk caching automatically and dynamically, through a CP routine known as the arbiter. The arbiter decides how many expanded storage blocks to allocate for paging and minidisk caching (MDC). It dynamically monitors the ratio of the number of users in I/O wait and the number of users in page wait, and adjusts the size of the MDC to minimize these enforced waits. By default, the arbiter can adjust the size of the MDC between zero blocks and the maximum number of blocks available to CP.

Sometimes the freedom exercised by the arbiter must be controlled more closely. To prevent a user from dominating the MDC area, CP allocates a fair share for each user based on the number of insertions that users are placing into the MDC.

Some virtual machines, such as SFS servers, require more than their fair share of the minidisk cache. This fair share can be overridden by the NOMDCFS directory option, which removes the limit of insertions allowed to the virtual machine.

Dedicating Expanded Storage to Guests

With the CP ATTACH XSTORE command, part of the expanded storage can be dedicated to a VM/ESA guest for its exclusive use. If there is not sufficient expanded storage, CP attaches whatever is available.

In the CP ATTACH XSTORE command in Figure 13, 512 MB of expanded storage are dedicated to MVSESA and 256 MB of expanded storage are still available to be attached to another virtual machine or to be used by CP.

The CP DETACH XSTORE command can be used to detach expanded storage from a guest. However, you must be sure that the guest is not using that expanded storage (vary the expanded storage offline in the guest) before you detach it; otherwise, you may disrupt work in progress in the guest virtual machine.

```
attach xstore to mvsesa 512m

XSTORE attached to MVSESA size= 512M

query xstore

XSTORE= 768M online= 768M
XSTORE= 256M userid= SYSTEM usage= 7% retained= 0M pending= 0M
XSTORE MDC min=0M, max=256M, usage=0%
XSTORE= 512M userid= MVSESA
XSTORE= 256M userid=  (none)  max. attach= 256M
```

Figure 13. CP QUERY XSTORE Command

Chapter 5. Direct Access Storage Devices (DASD)

Several types of DASD have been developed by IBM for use in different system configurations. Some are designed primarily for small and intermediate systems; others are designed primarily for large systems.

The types of devices differ from one another in several ways. Among these are storage capacity, speed, physical appearance, and environmental aspects, such as floor space occupied, power consumed, heat generated, and so on.

Most types of devices are offered in several models, which also differ from one another in various ways, such as capacity, speed, and number of devices per unit, among others. Table 1, Table 2, and Table 3 list the main characteristics of the CKD and FBA devices types supported by VM/ESA Release 2.1.

Important characteristics are the number of concurrent data paths that one string of devices can have, the mode of operation of the storage control (DLS or DLSE), and the capability to reconnect to a different path than the one used to start the I/O operation (Dynamic Path Reconnection, or DPR for short).

Important differences between CKD and FBA devices are related to volume capacity and data rate.

With CKD devices, the number of bytes/volume is seldom, if ever, experienced by the user. To achieve the maximum, all records must be written using full-track size, which is generally not desirable from the application point of view.

In contrast, with FBA devices the maximum data storage capacity is always available to the user. The capacity stated for an FBA device is usable space regardless of the block size written by the application. With CKD devices, the capacity stated is usable space only if full-track blocking is used.

Because data in FBA devices is stored in 512-byte blocks, the gaps between each FBA block are much smaller than those of CKD devices. However, a typical FBA user record contains more gaps and the data is not transferred in a continuous stream as it is for CKD devices; therefore, the instantaneous data rate cannot be sustained for multiple blocks of user data.

Hence, you should use the sustained data rate when calculating the data transfer time for FBA records. This can also happen with CKD devices when you read more than one record; the number of bytes corresponding to the gaps must be considered when calculating the data transfer time for CKD devices.

DASD I/O Management

CP manages I/O to the real devices and simulates I/O to the virtual devices when these virtual devices are not handled by the hardware I/O interpretation facility (SIE assist). The Real I/O Supervisor performs the following I/O services for the virtual machines and for CP:

- Queueing and executing I/O requests
- First-level interrupt handler
- Interface to error recovery routines

Table 1. Main Characteristics of CKD Devices (Part 1)

Characteristic	3330 (see note)			3350	3375	3380		
	1	2	11	2	1	Std	D	E
Megabytes/volume	100	100	200	317	410	630	630	1260
Cylinders/volume	404	404	808	555	959	885	885	1770
Bytes/track	13030	13030	13030	19069	35616	47476	46476	46476
Tracks/cylinder	19	19	19	30	12	15	15	15
4-KB blocks/cylinder	57	57	57	120	96	150	150	150
Data rate (KB/s)	806	806	806	1198	3000	3000	3000	3000
Minimum seek time	10	10	10	10	4	3	3	3
Average seek time	30	30	30	25	19	16	15	17
Maximum seek time	55	55	55	50	38	30	28	31
Latency (1/2 revolution)	8.3	8.3	8.3	8.4	10.1	8.3	8.3	8.3
Data paths	1	1	1	1/2	1/2	2	2	2
DPR	No	No	No	No	No	Yes	Yes	Yes

Note: IBM 3330s are supported in 3350-emulation mode only.

Table 2. Main Characteristics of CKD Devices (Part 2)

Characteristic	3380		3390				9345	
	J	K	1	2	3	9	1	2
Megabytes/volume	630	1890	946	1892	2838	8514	1004	1503
Cylinders/volume	885	2655	1113	2226	3339	10017	1440	2156
Bytes/track	47476	47476	56664	56664	56664	56664	46456	46456
Tracks/cylinder	15	15	15	15	15	15	15	15
4-KB blocks/cylinder	150	150	180	180	180	180	150	150
Data rate (MB/s)	3	3	4.2	4.2	4.2	3.9	4.4	4.4
Minimum seek time	2	2	1.5	1.5	1.5	2.5	1.5	1.5
Average seek time	12	16	9.5	12.5	15	22.5	10	11
Maximum seek time	21	29	18	23	33	38	16	20
Latency (1/2 revolution)	8.3	8.3	7.1	7.1	7.1	22.8	5.6	5.6
Data paths	4	4	4	4	4	4	2/4	2/4
DPR	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

I/O Scheduling

CP initiates real I/O operations on its own behalf, or on behalf of virtual machines, by initializing an operation request block (ORB) that describes the request and issuing a Start Subchannel (SSCH) instruction.

When an I/O request is ready to be started, its associated real device block (RDEV) is inspected. If no I/O is in progress in the associated real device, an SSCH instruction is issued to initiate the operation. The RDEV is marked busy and the I/O request, represented by an I/O Request Block (IORBK), is chained to the RDEV to indicate that it is the active request for the real device.

Table 3. Main Characteristics of FBA Devices

Characteristic	3370		9332		9335	9336	
	1	2	40X	60X	1	10	20
Megabytes/volume	286	365	184	284	412	471	857
Cylinders/volume	750	958	1349	2017	1963	1458	1458
Bytes/track	31744	31744	37888	37888	36352	29696	29696
Tracks/cylinder	12	12	8	8	6	11	20
Blocks/track	62	62	74	74	71	58	58
Blocks/cylinder	744	744	592	592	426	632	1149
Blocks/volume	558000	712752	360036	554800	804714	920115	1672881
Max data rate (MB/s)	1.859	1.859	2.5	2.5	3	3	3
Sustained rate (MB/s)	1.32	1.32	2.05	2.05	2.2	2.2	2.2
Minimum seek time	5	4	3.2	2.6	4.5	1.2	1.2
Average seek time	20	19	19.5	19.5	18	11.2	11.2
Maximum seek time	40	38	25	25	37	28	28
Latency (1/2 revolution)	8.3	8.3	9.6	9.6	8.3	8.3	8.3
Data paths	1	1	1	1	1	2	2
DPR	No	No	No	No	No	Yes	Yes

The channel subsystem processes the ORB stored in the subchannel by the SSCH instruction, selects a path, and passes the request to the proper device.

When the I/O operation is complete, the channel subsystem generates an I/O interrupt; CP issues a Test Subchannel (TSCH) instruction that causes the channel subsystem to return an Interruption Response Block (IRB). If the channel subsystem detects a condition that cannot be reported in an IRB, it reports the result as a machine check, which may be accompanied by one or more Channel Report Words (CRWs) describing the condition in further detail.

If the I/O request was initiated on behalf of a virtual machine, CP reflects the result to the virtual machine by means of a virtual I/O interrupt, translating the result to a form appropriate to the virtual machine architecture.

When the processing of the I/O is complete, the queue of I/O requests is inspected. If no IORBKs are queued, the RDEV is marked available. Otherwise, the first IORBK of the appropriate queue is selected.

I/O Queueing

If the RDEV is marked busy because an I/O operation is in progress on the associated real device, the I/O request, represented by an I/O Request Block (IORBK), is placed in the appropriate queue for the real device.

For fixed-head CKD DASD and for FBA DASD, one request queue, called the immediate queue, is maintained in FIFO sequence. For other CKD devices, three I/O request queues are maintained, as follows:

- Immediate queue

  All requests for nonstart operations, requests within the fixed-head area, and requests for the cylinder for which an I/O is currently active, are placed in the immediate queue in FIFO sequence. This is the first queue from which IORBKs are dequeued.

- High queue

  All requests to start operations to a cylinder number higher than the cylinder for which an I/O operation is currently active are placed in the DASD high queue in ascending real-cylinder-address sequence. This is the second queue from which IORBKs are dequeued.

- Low queue

  All requests to start operations to a cylinder number lower than the cylinder for which an I/O operation is currently active are placed in the DASD low queue in descending real-cylinder-address sequence. This is the third, and last, queue from which IORBKs are dequeued.

Because of this ordering, as I/O operations are executed by CP, the actuator moves from lower-numbered cylinders to higher-numbered cylinders until the end of the DASD high queue; the direction is then reversed until the beginning (first request queued) of the DASD low queue.

When the beginning of the DASD low queue is reached, the selection process is repeated. This technique causes the actuator to move back and forth across the cylinders on a disk volume in a way that minimizes actuator movement.

Other I/O operations, such as Store Subchannel and Modify Subchannel, that obtain only subchannel information, are queued in the immediate queue. Nonstart I/O operations issued by virtual machines, such as Test Pending Interrupt, Store Channel Report Word, Set Address Limit, and others, are simulated using the I/O control blocks for the addressed virtual subchannel; that is, no I/O operation is actually executed.

All FBA IORBKs are queued in the immediate queue in FIFO sequence, because the ordered seek queueing algorithm might lead to a data integrity problem. Assume that one I/O operation writes six blocks starting at block number 7437, followed by a second I/O operation from another task that writes three blocks starting at block number 7440, as shown in Figure 14.

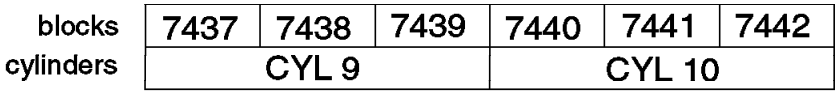

Figure 14. FBA Cylinders and Blocks

If the actuator is moving up, the six blocks would be written first; if the actuator is moving down, the three blocks would be written first. This can happen only when two virtual devices are overlapping because normally there is only one I/O operation at the virtual device. Since this problem could also occur with CKD devices, and overlapping virtual devices can cause data integrity problems on its own, CP could schedule FBA I/O in the same way as CKD I/O scheduling.

IBM 3990 Storage Control Model 3

The IBM 3990 Storage Control Model 3 is a high performance DASD storage control that incorporate cache storage and related functions. This section also applies to the IBM 3990 Storage Control Model 6 when operating in basic mode; the enhanced mode of the IBM 3990 Model 6 is discussed in “IBM 3990 Storage Control Model 6” on page 59.

Caching provides performance benefits by bringing the data closer to the processor, thus reducing the overall service time of I/O operations. This allows higher device utilizations and the use of higher capacity devices while maintaining a good device response time.

The features available on the IBM Storage Control Models 3 and 6 include:

- Cache storage

  Cache is a volatile storage that buffers the more frequently used data in the attached DASD. Data references can be satisfied at channel speeds without mechanical delays, if the cache contains the required data.

  Data can be transferred between the channel and the cache, between the channel and device, and between the device and cache. Data can be transferred from the device to the channel at the same time that data goes to cache. Data can also be transferred between cache and channel, cache and the device, and cache and nonvolatile storage at the same time.

- Nonvolatile Storage

  Nonvolatile Storage (NVS) provides storage to maintain data integrity for fast write and dual copy operations in the event of a power failure. If a power outage occurs before the data is stored on DASD, a battery-backup system maintains power on NVS to prevent data loss for up to 48 hours, assuming a fully-charged battery. When power is restored, the storage control destages any data in NVS to DASD.

- Cache storage transfer rates are 4.5 MB/s through parallel channels, and up to 17 MB/s through ESCON channels.

- Synchronous and nonsynchronous modes of operations

  A 64-KB Automatic Data Transfer (ADT) buffer for each storage path is used for 2-level error correction code (ECC) and for nonsynchronous operations (see “Nonsynchronous I/O Operations” on page 89).

- Modes of operation

  - Device Level Selection (DLS) which enables two simultaneous data transfers to and from the devices. Cache and NVS are logically divided into two equal parts; one part is assigned to each pair of single-path storage directors in a subsystem. Cache storage and cache directory for each DLS subsystem is kept separate and is not shared with the other DLS subsystem.

  - Device Level Selection Enhanced (DLSE) which enables four simultaneous data transfers to and from the devices. Both cache and NVS remain a single entity and are shared by both multipath storage directors; this is the preferred mode of operation for the IBM 3990.

The IBM 3990 Model 3 can operate in either DLS or DLSE mode and can attach 2-path, 4-path, or a mixture of both kinds of strings (3380 and 3390 devices). The IBM 3990 Model 6 operates in DLSE mode only and with 3390 devices only.

Querying Cache and NVS

Figure 15 shows how to display the cache size information with the CP QUERY CACHE command.

```
cp query cache 960

0960 CACHE available for subsystem
00262144K Bytes configured
00261296K Bytes available
00000000K Bytes offline
00000000K Bytes pinned
```

Figure 15. Displaying Cache Size Information

The command response indicates that out of the 262144 KB (256 MB) of cache configured, 261296 KB are available for allocation. The difference of 848 KB between bytes configured and bytes available is used for the cache directory, which contains control information about tracks staged into the cache.

Figure 16 shows how to display the NVS size information with the CP QUERY NVS command.

```
cp query nvs 960

0960 Non-volatile storage is available.
00004096K Bytes are configured.
00000000K Bytes are pinned.
```

Figure 16. Displaying NVS Size Information

Status

The status of the storage control can be displayed with the CP QUERY DASD DETAILS command as shown in Figure 17. Do not rely on an initial (default) status because it depends on the level of the Licensed Internal Code (LIC).

```
cp query dasd details 960

0960  CUTYPE = 3990-EC, DEVTYPE = 3390-06, VOLSER = D14AA1
     CACHE DETAILS:  CACHE NVS CFW DFW PINNED CONCOPY
         -SUBSYSTEM   Y   N   Y   -   N      N
           -DEVICE   Y   -   -   N   N      N
     DEVICE DETAILS: CCA = 60, DDC = 20
     DUPLEX DETAILS: Simplex
```

Figure 17. IBM 3990 Status

The CUTYPE and DEVTYPE returned by the command is taken from the response of the Read Device Characteristics CCW. In the example shown in Figure 17, 3990-EC indicates an IBM 3990 capable of nonsynchronous operations; 3390-06 indicates an IBM 3390 Model 2 working in native mode.

Cache Functions

The following functions are provided by the IBM 3990:

- Basic cache
- Cache fast write
- DASD fast write
- Dual copy
- Concurrent copy

This section describes the fundamental caching concepts and provides a description of each of the IBM 3990 cache functions.

Least-Recently-Used Algorithm

The IBM 3990 works on the premise that data that was last referenced is likely to be referenced again, while data that has not been referenced for some time is not likely to be referenced soon.

Additionally, the IBM 3990 works under the assumption that a user is likely to reference data that is close to data being referenced; this is called locality of reference.

A Least-Recently-Used (LRU) algorithm replaces the least recently used data in cache when space is needed for new data in cache. The LRU algorithm optimizes cache performance and ensures that the most appropriate data is stored in the cache at any one time.

The IBM 3990 performs promotion, demotion, staging, and destaging operations as it manages cache through the LRU algorithm. These operations are defined as follows:

Promotion The process of validating a track image in the cache; promotion may or may not involve staging the track from DASD to cache.

Demotion The process of removing the image of one or more records from the cache either by being selected for replacement (overlay) by another set of DASD records or by being marked invalid.

Staging The process of writing data from a DASD to the cache.

Destaging The process of asynchronously writing new or updated data from cache or NVS to DASD.

Whenever a new track is promoted to cache, track slot segments are allocated. If all the track slot segments are in use, the IBM 3990 selects the most appropriate slot to be demoted or destaged.

All track slots are contained in an LRU list in cache storage, as shown in Figure 18. The LRU list is ordered from the most-recently-used track slot at the top to the least-recently-used in the bottom.

Each time new data is promoted, the track slot is selected from the bottom of the LRU list. The least-recently-used track is demoted unless it contains modified data, in which case it is destaged; the old data is not destaged if a valid copy exists on DASD.

A newly promoted track slot or track slot that is the target of a read hit is marked most recently used by moving it to the top of the LRU list.

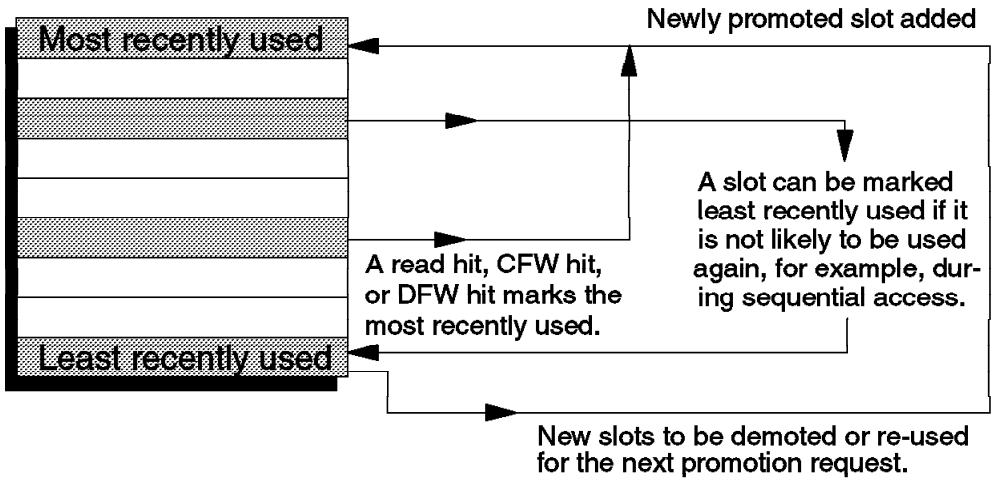

Figure 18. LRU List Example

Caching Modes

One of several caching modes can be specified on an I/O operation basis in the Define Extent CCW; therefore, the caching mode is determined by the execution of a channel program and is valid only while that channel program is in still executing.

Normal Caching Mode: This mode is the default mode; that is, it is in effect unless a Define Extent CCW changes the caching mode. When a read miss occurs, the requested record and the remaining records on the track are promoted to the cache. When space is needed in cache for additional tracks, the IBM 3990 destages the least recently used tracks to DASD.

Sequential Access Mode: This mode is set by the Define Extent CCW when the access method determines that tracks should be processed sequentially. The storage control improves the overall cache performance by promoting tracks in advance, before a read miss occurs. The storage control also minimizes the number of tracks slots in use by demoting track slots shortly after they have been processed.

Bypass Cache: This mode is set by the Define Extent CCW and causes the data in the cache to be bypassed; the I/O request is sent directly to DASD. Tracks are neither read from cache nor promoted to cache; the LRU list is not updated.

If a basic cache write operation occurs in this mode and the track is also in the cache, the corresponding cache slot is marked invalid. If the track slot was modified by a previous cache fast write hit or DASD fast write hit, the track is destaged and the slot is marked invalid. The performance of each I/O operation is almost equivalent to DASD being attached to noncached storage controls; the difference is the required directory search in a write operation.

Inhibit Cache Load: This mode is set by the Define Extent CCW and uses existing copies of the track if they are in the cache, but does not load any new tracks into the cache. This mode avoids the overhead that can be caused by track promotion when there is no benefit; that is, whenever these slots are not likely to be accessed again soon.

Dual Data Transfer and Device Allegiance

During stages and destages, the internal path and device are physically busy. The host considers the device free, because the device-end status has already been received but the device may still be physically busy while it has logically completed the I/O operation.

Dual data transfer allows the staging and destaging of up to four simultaneous and independent operations on the lower ports and at the same time, up to four independent operations on the upper ports; up to eight operations are permitted in the IBM 3990

Logical device allegiance permits the storage control to complete operations without selecting the physical device if the request does not have to access the physical device. Host access to cache data can be concurrent with stage and destage activities to the same volume.

Dual data transfer and logical device allegiance allows up to eight simultaneous operations, of which four are stage and destage operations and four are data transfers between the host and the storage control.

Basic Cache Read Operations

A cache read operation is the transfer of data from cache to channel, or DASD to cache and channel. Depending on the data already in cache, it is possible to have a read hit, a read miss, or a front-end miss.

Read Hit: Occurs when the requested record is already in cache. The channel program operates only with the cache and the DASD volume is not accessed. Because no DASD access is required, seek time, latency, and RPS miss delays are eliminated (see “DASD Performance” on page 66). The storage control:

1. Searches the cache for proper track
2. Transfers the requested data to channel
3. Presents CE and DE to channel
4. Flags staged tracks as most recently used

Read Miss: Occurs when the requested data is not in cache. The requested data is simultaneously transferred to cache and the channel, and the rest of the track is staged into the cache (starting with the record accessed). The storage control:

1. Reads the record from DASD
2. Transfers data simultaneously to cache and channel
3. Presents CE and DE status to channel
4. Stages the rest of the track (normal caching mode)
5. Stages the current and next tracks (sequential access mode)
6. Flags staged tracks as most recently used

When normal caching mode is in effect, the rest of the track is staged into the cache. When sequential access mode is in effect, the current and next tracks are staged into the cache. Normal caching mode is the default; sequential access mode is specified in the Define Extent CCW.

Front-End Miss: Tracks are normally staged from the first record following the addressed sector to the end of track. Therefore, many cache slots actually contain partial track images. When the requested track is in cache, but the addressed record is not, only records at the front end of the track (from record 0 up to the existing record) are transferred from DASD to cache.

Basic Cache Write Operations

The primary objective of a basic cache write operation is to emulate a DASD write and to ensure that the DASD copy of the data is always valid. All write commands operate directly with DASD unless DASD fast write or cache fast write is active.

Performance gains are realized by applications with a high read-to-write ratio; high activity read-only minidisks such as the 190-S and the 19E-Y minidisks are natural candidates for basic caching. Other public minidisks with similar characteristics should also be considered as cache candidates. However, applications with low read-to-write ratios still benefit from cache speeds when using DASD fast write, as discussed later.

The probability of a read hit, evaluated as number of read operations that find the data in cache divided by the total number of read operations, is called *read hit ratio*. You can define the write hit ratio, DASD fast write hit ratio, and overall hit ratio in a similar manner.

Depending on the data already in cache, it is possible to have a write hit or a write miss.

Write Hit: Occurs when a write request updates a record that is already in cache. Without fast write active, the storage control:

1. Searches the cache for proper track
2. Transfers data to cache and to DASD concurrently
3. Presents CE and DE to channel

Therefore, there are no performance improvements for basic cache write operations.

Write Miss: Occurs when a write request updates a record that is not in cache. Except for the initial cache directory search, a write miss on cached storage controls works exactly like a write operation on noncached storage controls:

1. Data is transferred from the channel to DASD
2. CE and DE are presented to the channel

Cache Fast Write

Data recorded in cache fast write mode remains only in cache storage; it is written to DASD only if the LRU algorithm needs more cache space. Therefore, cache fast write (CFW) is appropriate for files that are not critical to the application; that is, cache fast write is used by applications that generate data that may not be required at the completion of the task, or data that may be easily reconstructed, if necessary.

An application can use cache fast write when CFW is enabled in the IBM 3990 and the CFW flag and CFW id are set in the Define Extent CCW. If CFW and DFW are active on the same volume at the same time, CFW overrides DFW for the duration of the CFW channel program.

Temporary files are good candidates for cache fast write mode. Typically, work files created by sort programs and compilers are the best candidates for the cache fast write function.

VM/ESA tolerates cache fast write for guest use; currently, only MVS DFSORT exploits cache fast write.

DASD Fast Write

DASD fast write (DFW) provides a significant write performance enhancement over basic cache write operations because the device-end status is presented when the cache data transfer is complete. Access to a physical DASD is not required for write hits and predictable writes, which are treated as write hits. DASD fast write gives an improvement to the write operations similar to that provided by basic cache to read operations.

On a DASD fast write hit, the copy of the record in cache is updated and the data is stored in NVS; the track is flagged as most recently used. Channel-end and device-end status are returned together at the end of cache-speed data transfer; data remains in cache and in NVS until the storage control destages the data to DASD through the LRU algorithm.

A DASD fast write miss functions in a manner similar to a basic cache read miss; the data to be written is transferred to cache and DASD, and the track is staged into the cache from the point of access.

Because the IBM 3990 stores the write data simultaneously in cache storage and in NVS, data integrity is equivalent to DASD writes. If a power failure should occur, the storage control retains the data in NVS for up to 48 hours, assuming a fully charged battery. When power is restored, the storage control destages the data from NVS to DASD.

The storage control destages DASD fast write data to DASD when DASD fast write, NVS, subsystem cache, or device cache are deactivated. A destage command may be issued during CP shutdown process.

Files with high write-to-read ratios obtain the most benefit from DASD fast write. In order to take advantage of the DASD fast write function, an application should have a cache write hit ratio above 90%. Examples of such applications include:

- Log files such as Shared File System (SFS) logs, Coordinated Resource Recovery (CRR) logs, and RACF SMF logs.
- Checkpoint files, such as the CP SPOOL checkpoint area.
- Monitor data.

The following circumstances favor cache write hits:

- A previous write operation with a cache write miss writes the data to DASD and then reads the remainder of the track into the cache on the same revolution. A later write operation to a record located in the remainder of this track finds a cache write hit condition.
- A previous read operation with a cache read miss also brings the record and the remainder of the track to cache, favoring a later write hit to the same or subsequent records on the same track.
- The access method may set sequential access mode. When the IBM 3990 cache algorithm is aware of sequential access mode, the storage control stages the accessed track and the next tracks into the cache. Later, write operations may find cache write hit conditions.
- When an application updates records, it precedes write operations with read operations. Therefore, the record is located on a track that is already in cache storage when the write operation occurs.

Dual Copy

The dual copy function (DC) allows you to maintain logically identical copies of a DASD volume on two devices in the same subsystem. The IBM 3990 internally synchronizes the volumes by writing all modified data on both volumes.

The dual copy function provides a high efficiency back-up copy of important data. It also alleviates the load on the central processor and channels that would be incurred to provide the back-up copy without using the dual copy function.

To use dual copy, two devices are designated as a dual copy logical volume (also called a duplex pair). One device is designated as the *primary* and the other is designated as the *secondary*. The primary device must be online; the secondary device may be offline or online, but must not be attached to any virtual machine or SYSTEM, or be a CP-owned device.

During read operations, the IBM 3990 reads data from the primary volume. If the read operation fails because of a data check, the data is read from the secondary volume. If other permanent device hardware errors occur, the duplex pair is suspended and all operations are directed to the operational volume. The changed cylinder numbers are logged in NVS so the IBM 3990 can restore dual copy operations upon request when the hardware is repaired.

Basic cache at the device level must be inactive for both volumes before a duplex pair is established. After establishing the duplex pair, you should at least re-activate basic cache; DFW should also be activated if you want to enable the fast dual copy function (see details in “Dual Copy Enabling” on page 57).

When only dual copy is active, with basic cache inactive, the device-end status is returned only after data is written to the primary volume and then to the secondary volume; this is done serially.

When basic cache is also active, the device-end status is returned after data is written to the primary volume. Data is subsequently written to the secondary volume from the cache. The completion of the write operation on the secondary is not signalled to the system unless an error occurs.

When both basic cache and DASD fast write are also active, a write hit causes data to be transferred simultaneously to the cache and NVS. Channel-end and device-end status are presented to the channel, allowing the application to continue processing. The changed data is scheduled for immediate destage to the primary volume, keeping the logical device busy; the changed data is asynchronously destaged to the secondary through the LRU algorithm. A write miss causes data to be written directly to the primary volume; data is destaged to the secondary volume as before. This process is an old implementation of the fast dual copy function (FDC), and may still exist in some IBM 3990 with old licensed internal code (LIC).

The current LIC implements an enhanced fast dual copy that processes write hits in a manner similar to normal DFW processing; that is, data is transferred into the cache and NVS and channel-end and device-end status are presented to the channel. The logical device is free as soon as device-end is presented; data is asynchronously destaged to both primary and secondary volumes.

Enhanced fast dual copy is recommended because it combines availability and performance; in some cases, the performance is close to that of DFW without dual copy.

Concurrent Copy

Concurrent copy allows backing up and copying databases while they are actively being updated. VM/ESA provides guest support for the IBM 3990 concurrent copy extended function; that is, VM/ESA tolerates the use of concurrent copy by any number of guests sharing the IBM 3990.

Before the backup process starts, update activity is quiesced and a snapshot of the database is captured. After the snapshot, normal update activity to the database can continue. Whenever a track is updated prior to its being copied, the original image is first saved into a cache sidefile in the storage subsystem and then the update continues normally. The saved track image is then transferred to the host backup program, which later merges it into the destination file. The result is an identical copy of the database at the time that the snapshot was taken.

The following concepts and definitions are used in the concurrent copy facility:

- Register Concurrent Copy Session

  The process of informing the storage control subsystem of the token value to be used as a concurrent copy session identifier when establishing a concurrent copy session with individual devices.

- Establish Concurrent Copy Session

  The process of defining the DASD and track extents to be associated with a concurrent copy session for the addressed device.

- Discard Concurrent Copy Track

  The process of removing a concurrent copy track from the set of tracks associated with a concurrent copy session.

- Withdraw Concurrent Copy Extent

  The process of de-establishing a device from concurrent copy session if no tracks remain to be copied for the specified concurrent copy session.

- Terminate Concurrent Copy Session

  The process of de-registering a concurrent copy session. All subsystems resources related to the session are released.

- Cache sidefile

  A dynamically allocated area of the IBM 3990 cache that is used to temporarily save DASD track images for the concurrent copy function. These track images are original track data for those tracks that have been updated prior to being copy by the host backup program. If there is no space left in the cache when a sidefile has to be copied to the cache, the concurrent copy session is terminated by the subsystem.

A new directory statement (STDEVOPT DASDSYS) enables you to select which guests can use the concurrent copy facility. The CP QUERY CONCOPY command can be used to display concurrent copy session information. The CP CONCOPY TERMINATE command can be used to end concurrent copy sessions on behalf of virtual machines that started a concurrent copy session but, because of malfunction, are unable to end it.

Concurrent copy is used only by the DFSMSdss component of the DFSMS/MVS product.

VM/ESA Support

VM/ESA exploits basic cache, DASD fast write, and dual copy. Cache fast write and concurrent copy are tolerated; that is, VM/ESA itself does not use cache fast write and concurrent copy, but allows a guest to control and use these functions.

When specified and controlled by VM/ESA, the IBM 3990 functions, except cache fast write and concurrent copy, are automatically used by CP; that is, the guest operating system does not have to explicitly invoke the functions. This automatic use of the IBM 3990 functions is targeted to CMS minidisks, but can also be used by guest operating systems that support the caching functions.

Table 4 summarizes the IBM 3990 support by VM operating systems. VM/SP Releases 5 and 6, and VM/ESA 370 Feature (Releases 1.0 and 1.5) do not support IBM 3990.

Table 4. Summary of VM Support of IBM 3990 Cache Functions

VM Operating System	Basic Cache	DASD Fast Write	Cache Fast Write	Dual Copy	Concur-rent Copy
VM/SP HPO Releases 5 and 6	Ö	Ö	Ö		
VM/XA SP Release 2.1	Ö				
VM/ESA Release 1.0 (ESA)	Ö				
VM/ESA Release 1.1	Ö	Ö	Ö	Ö	
VM/ESA Release 2.0	Ö	Ö	Ö	Ö	Ö
VM/ESA Release 2.1	Ö	Ö	Ö	Ö	Ö
VM/ESA Release 2.2	Ö	Ö	Ö	Ö	Ö

Certain operations may cause presentation of channel-end and device-end status before the operation completes. These operations are potentially long-running commands, such as the DUPLEX, SET CACHE, SET CACHEFW, SET DASDFW, and SET NVS. The CP command responses associated with such operations are called delayed responses; you may use the CP QUERY PENDING COMMANDS command to check which commands are still pending.

Table 5 lists the CP commands related to the IBM 3990; most of them are privilege class B commands. Refer to *VM/ESA CP Command and Utility Reference* for a complete description of these commands.

Using caching for paging volumes may cause a performance degradation because the IBM 3990 would do considerable staging and destaging of data that is referenced only once, or not at all.

Because VM/ESA does not issue its paging I/Os inhibiting cache, caching should be disabled for volumes containing paging areas. This should not be a problem because the paging volumes normally contain only page areas and low activity minidisks.

You should also limit the cache usage to those volumes that have high cache hit ratios (read hit ratio above 80% and write hit ratio above 90%). Since the cache has a limited storage, several stages and destages of data may cause the overall performance to be even worse than without caching.

Table 5. CP Commands Related to IBM 3990

CP Command	Description
ATTACH	Attaches real disks to virtual machines specifying the control level option.
COMMIT	Transfers cache fast write data to the addressed device.
CONCOPY TERMINATE	Ends concurrent copy on behalf of a guest that started CONCOPY.
DESTAGE	Transfers modified data (cache fast write or DASD fast write) to DASD.
DISCARD PINNED	Discards pinned data that is stored in cache or NVS.
DUPLEX (*)	Establishes duplex pairs for the dual copy function.
QUERY CACHE	Displays the number of cache bytes available and the device cache status.
QUERY CACHEFW	Displays the cache fast write function status for device.
QUERY CONCOPY	Displays concurrent copy session information.
QUERY DASD DETAILS	Displays the caching and NVS status for device.
QUERY VIRTUAL DASD DETAILS	Displays the caching status and the control level.
QUERY DASDFW	Displays the DASD fast write function status for device.
QUERY DUPLEX	Displays the dual copy function status of duplex pairs or simplex volumes.
QUERY FENCES	Displays information about IBM 3990 internal errors.
QUERY NVS	Displays the NVS storage status and the number of bytes configured.
QUERY PENDING COMMANDS	Displays commands already started for which the associated asynchronous function has not yet completed.
QUERY PINNED	Displays the tracks and bytes of cache and NVS containing pinned data.
QUERY RSAW	Generates and displays the remote access service word (RSAW) for storage clusters.
SET CACHE (*)	Activates or deactivates the basic cache function by device or by subsystem.
SET CACHEFW (*)	Activates or deactivates the cache fast write function in the subsystem.
SET DASDFW (*)	Activates or deactivates the DASD fast write function for device.
SET NVS (*)	Makes NVS available or unavailable for the subsystem.
SHUTDOWN	Terminates all outstanding concurrent copy sessions. It can also perform a DESTAGE command as part of its operation.

Note: The commands with asterisk (*) may result in delayed responses. Use the QUERY PENDING COMMANDS command to see which commands are still pending.

You can control caching at the subsystem level, device level, or minidisk level; therefore, if a device contains several minidisks, it is possible to enable cache selectively for the minidisks you select.

Basic Cache Enabling

The basic cache function and the cache fast write function are initially active. Basic cache is effectively active only when the all the following conditions are met:

- MINIOPT statements do not inhibit cache usage
- CACHE is ON at the SUBSYSTEM level
- CACHE is ON at the DEVICE level

Check the response of the CP QUERY DASD DETAILS command to verify whether basic cache is effectively active.

DASD Fast Write Enabling

In order to have DASD fast write enabled and available for use, five conditions must be met:

- MINIOPT statements do not inhibit cache usage
- CACHE is ON at the SUBSYSTEM level
- CACHE is ON at the DEVICE level
- NVS is ON at the SUBSYSTEM level
- DFW is ON at the DEVICE level

The following sequence of CP commands can be used to activate DASD fast write on device X'960', assuming that the function is not inhibited by a MINIOPT directory statement:

```
SET CACHE SUBSYSTEM ON 960
SET CACHE ON 960
SET NVS ON 960
SET DASDFW ON 960
```

Therefore, the response from the QUERY DASD DETAILS command that correctly indicates that the DASD fast write function is active should be similar to Figure 19, where the required fields are highlighted; if any highlighted field is N, DFW is not active.

```
cp query dasd details 960

0960  CUTYPE = 3990-EC, DEVTYPE = 3390-06, VOLSER = D14AA1
     CACHE DETAILS:  CACHE NVS CFW DFW PINNED CONCOPY
         -SUBSYSTEM   Y    Y   Y   -    N      N
           -DEVICE    Y    -   -   Y    N      N
     DEVICE DETAILS: CCA = 60, DDC = 20
     DUPLEX DETAILS: Simplex
```

Figure 19. DASD Fast Write Enabling Conditions

The DASD fast write function and the cache fast write function can be active on the same volume at the same time. When cache fast write is active for a channel program, it overrides DASD fast write for the duration of that channel program.

Dual Copy Enabling

The dual copy function is activated through the DUPLEX command which can be used to establish (ON parameter) or discontinue (OFF parameter) a duplex pair. This function is asynchronous and may take some time to complete, particularly when you specify that the primary volume should be copied to the secondary volume (COPY parameter, which is the default).

The PACE parameter allows you to specify a rate at which the copy should be performed (default is number of tracks/cylinder). On rare occasions, the response may be lost; therefore, check the completion of the function.

In old levels of LIC, basic cache and DFW must be inactive for both volumes before a duplex pair is established, but you can turn them back on after the pair is established. The recommended sequence of commands to enable dual copy and fast dual copy is:

```
SET CACHE SUBSYSTEM ON 960
SET NVS ON 960
SET CACHE OFF 960 967
SET DASDFW OFF 960 967
DUPLEX 960 ON 967
SET CACHE ON 960
SET DASDFW ON 960  (fast dual copy only)
```

After the duplex pair is established, you cannot issue any command against the secondary volume; however, the conditions set for the primary volume are also valid for the secondary volume. For example, SET CACHE ON 960 also establishes CACHE ON for 967.

The correct status should be checked using the command shown in Figure 20.

```
cp query dasd details 960 967

0960  CUTYPE = 3990-EC, DEVTYPE = 3390-06, VOLSER = D14AA1
     CACHE DETAILS:  CACHE NVS CFW DFW PINNED CONCOPY
         -SUBSYSTEM   Y    Y   Y   -    N      N
           -DEVICE    Y    -   -   Y    N      N
     DEVICE DETAILS: CCA = 60, DDC = 20
 DUPLEX DETAILS: Primary, OTHER DEVICE = 67, Active duplex

0967  CUTYPE = 3990-EC, DEVTYPE = 3390-06, VOLSER = D14AA1
     CACHE DETAILS:  CACHE NVS CFW DFW PINNED CONCOPY
         -SUBSYSTEM   Y    Y   Y   -    N      N
           -DEVICE    Y    -   -   Y    N      N
     DEVICE DETAILS: CCA = 67, DDC = 27
 DUPLEX DETAILS: Secondary, OTHER DEVICE = 60, Active duplex
```

Figure 20. Dual Copy Enabling Conditions

You can terminate dual copy (split the duplex pair) by specifying the OFF parameter in the CP DUPLEX command. You can also suspend dual copy (SUSPEND parameter) by specifying which volume, primary or secondary, should be marked as suspended; the other volume becomes the only operational volume of the duplex pair. The duplex pair can be re-established using the ON parameter on the CP DUPLEX command; you can specify the same suspended volume or another volume in the same subsystem.

Guest Usage of IBM 3990

A guest operating system can use the IBM 3990 cache functions without any further control from the guest, provided that the guest itself supports the IBM 3990 functions. The cache functions can be activated for these guests using CP commands, as discussed in “CP Management” on page 56.

However, VM/ESA allows a guest with IBM 3990 support to invoke and use the IBM 3990 functions directly. The level of control depends on the option specified in the DASDOPT directory statement. The DASDOPT statement must be included immediately after the related DEDICATE, LINK, or full-pack MDISK statement.

The DASDOPT directory statement specifies one of the three levels of guest control described below:

NOCTL This option prohibits guest use of any CCWs that control cache functions.

DEVCTL This option allows guest use of CCWs that control device-level cache functions. Subsystem-level commands cannot be used by the guest.

SYSCTL Guest use of all cache control CCWs supported by VM/ESA, including global subsystem functions, is allowed. This option provides full control over the entire storage subsystem.

An alternative to the DASDOPT directory statement is the ATTACH command and the WITH option; for example:

```
ATTACH  240-247  TO  MVS1  WITH  SYSCTL
```

The following considerations apply when DASDOPT refers to dedicated devices, or when the ATTACH command is used:

- The NOCTL option is not valid for preferred guests. If it is specified for preferred guests (V=R or V=F), the default is used (see below).
- For 3880 storage controls, the default level of control assigned to the devices of preferred guests is SYSCTL, and the default assigned to V=V guests is DEVCTL.

  Because I/O assisted devices are not under CP control, and the IBM 3880 Storage Control does not provide any mechanism to disallow system-level or device-level commands, NOCTL and DEVCTL are not accepted for preferred guests.
- For the IBM 3990, DEVCTL is the default for V=V, V=F, and V=R guests.

  CP primes the IBM 3990 with Set Special Intercept Condition (SSIC) before dispatching a guest with I/O assisted devices in the subsystem. In this case, the IBM 3990 responds with a unit check whenever the guest tries to use unauthorized subsystem commands.

Note the following:

- SYSCTL is the recommended option for MVS systems; otherwise, an error message may occur when MVS issues destage commands during EOD processing.
- Although the IBM 3990 caching functions are completely managed by the licensed internal code of the storage control, a guest must include support for the IBM 3990 cache functions to be able to use them under VM/ESA.

IBM 3990 Storage Control Model 6

The IBM 3990 Model 6 can emulate an IBM 3990 Model 3. When operating in this manner, called *basic mode*, the IBM 3990 Model 6 still provides many enhancements when compared to the IBM 3990 Model 3, but requires no additional software beyond what is required for the IBM 3990 Model 3. One exception is when using some new functions, such as remote copy, that requires additional software support.

The IBM 3990 Model 6 can also operate in *extended mode* which requires additional software support, regardless of the functions used.

When operating in extended mode, the IBM 3990 Model 6 supports the same functions it supports in basic mode. The enhanced mode provides the following additional functions and does not remove any of the functions available in basic mode:

- RPS miss avoidance for parallel channels

  In basic mode, the IBM 3990 Model 6 supports RPS miss avoidance on ESCON channels only. In extended mode, RPS miss avoidance is extended to parallel channels. The IBM 3990 Model 3 with the Extended Platform also supports RPS miss avoidance on both parallel and ESCON channels. See "RPS Miss Avoidance" on page 93 for details.

- Control unit initiated reconfiguration

  In large configurations, quiescing channel paths in preparation for upgrades or service actions is a complex, time consuming, and potentially error prone process. Control unit initiated reconfiguration (C.U.I.R.) automates the process of quiescing and re-enabling channel paths, thus reducing the time required for service actions and dramatically reducing the operations staff effort required, especially in large geographically disparate systems.

The IBM 3990 Model 6 operating mode (basic or enhanced) is selected through the vital product data (VPD) settings. Changing the IBM 3990 Model 6 operating mode is a disruptive action.

Record Cache

The previous caching algorithm used by both IBM 3990 Model 3 and IBM 3990 Model 6 is called track caching; this algorithm is still available in the IBM 3990 Model 6. Record cache is a new algorithm that the IBM 3990 Model 6 uses to determine how much data to read into cache.

Many workloads already receive substantial benefits from using the IBM 3990 cache with the track algorithm; these workloads are called cache-friendly workloads. Record cache is designed to improve performance for workloads that do not perform well with the track caching algorithm, often called cache-unfriendly workloads. Normally, record cache does not greatly improve performance for cache-friendly workloads.

There are two implementations of record cache. One implementation requires software support in host systems that notifies the IBM 3990 Model 6 when to use the record cache algorithm; this implementation is often called record cache I.

The other implementation, called record cache II or adaptive cache, requires no software support. In this implementation, the IBM 3990 Model 6 automatically

determines whether to use track cache or record cache. Unlike cached IBM 9340 Subsystems whose adaptive cache automatically determines whether to use cache, record cache II in the IBM 3990 Model 6 never decides **not** to use cache. When you enable basic cache and DFW for a volume, the IBM 3990 Model 6 always use cache, but dynamically selects the management algorithm.

Write Operations: DASD fast write must be active for the device. All write operations that are eligible for record cache are considered DFW hits provided that adequate NVS space is available. No staging is performed for those hits when record cache is active; therefore, the physical volume and the IBM 3990 Model 3 lower interface are available for other I/O operations that are not cache hits.

Read Operations: Basic cache must be active for the device. Record cache treats a read cache hit in exactly the same way as track cache. For a read miss, the IBM 3990 Model 6 reads the record requested to the channel and into the cache at the same time. The rest of the track is **not** staged into the cache.

Dual Copy Enhancements

The following capabilities, which significantly enhance dual copy, are available for the IBM 3990 Models 3 and 6:

- Multitrack establish duplex pair

  With this enhancement, the IBM 3990 uses its multitrack stage and destage capability to copy tracks in groups of three when establishing a duplex pair. In the original implementation of dual copy, the IBM 3990 copies tracks one at a time.

- Dual copy cache status

  With this enhancement, the IBM 3990 permits DFW to be active while it establishes a duplex pair. In the original implementation of dual copy, DFW cannot be active during the establishment of dual copy (see “Dual Copy Enabling” on page 57).

No changes to existing software that supports the IBM 3990 Models 3 and 6 are required to benefit from these enhancements.

Remote Copy

The IBM 3990 Model 6 supports advanced remote copy facilities. One of these facilities offers DASD synchronous remote copy through the IBM 3990 to IBM 3990 peer-to-peer ESCON connections. The other offers asynchronous remote copy through system-connected IBM 3990 and system software.

The synchronous remote copy is also known as peer-to-peer remote copy, and the asynchronous remote copy is also known as extended remote copy.

Both facilities provide real-time remote copy of DASD data for disaster recovery of applications at a second data center. They differ in their effect on DASD I/O performance, the degree of data currency at the time of a disaster, use of system resources, and operational control of the remote copy facilities.

Both facilities are supported only by MVS/ESA running native (see Table 6 on page 61).

VM/ESA Release 2.2 Commands

Figure 21 shows the possible responses for some VM/ESA Release 2.2 CP commands related to the IBM 3990 Model 6.

```
cp query dasd details db3

0DB3  CUTYPE = 3990-CC, DEVTYPE = 3390-0A, VOLSER = TOTE22
      CACHE DETAILS:  CACHE NVS CFW DFW PINNED CONCOPY
          -SUBSYSTEM   M    T   Y   -    N      N
             -DEVICE   Y    -   -   N    N      N
      DEVICE DETAILS: CCA = B3, DDC = 33
      DUPLEX DETAILS: SIMPLEX

cp query paths db3

DASD 0DB3 online
Path status for device 0DB3: online - 10 14 1B 1F, offline - NONE

query chpid 1b

Path 1B online to devices 0DAC 0DAD 0DAE 0DAF 0DB1 0DB2 0DB3
```

Figure 21. IBM 3990 Cached Model Status

For SUBSYSTEM CACHE and NVS, status can be one of the following:

Y Function active

N Function inactive

T Function terminated

M Function disabled for maintenance

F Function pending inactive, destage has failed (cache only)

VM/ESA Support

Table 6 shows which IBM 3990 Model 6 facilities are supported for VM/HPO, VM/XA SP, and VM/ESA (ESA feature).

Table 6. Summary of VM Support of IBM 3990 Model 6 Functions

VM Operating System	Record Cache I (1)	Record Cache II	Dual Copy Enhan-cements	Parallel RMA	C.U. Initiated Reconfi-guration	Extend-ed Remote Copy	Peer-to-peer Remote Copy
VM/SP HPO Releases 5 and 6	Ö	Ö					
VM/XA SP Release 2.1 (2) (3)	Ö	Ö					
VM/ESA Release 1.0 (ESA) (2) (3)	Ö	Ö					
VM/ESA Release 1.1	Ö	Ö	Ö	Ö (3)			
VM/ESA Release 2.0	Ö	Ö	Ö	Ö (3)			
VM/ESA Release 2.1	Ö	Ö	Ö	Ö (3)			
VM/ESA Release 2.2	Ö	Ö	Ö	Ö	Ö		

Note:
(1) For guest support only
(2) Basic cache only
(3) Requires PTF

IBM 9340 Subsystem

The IBM 9343 Models CC2, CC4 and DC4 provide cache functions for improved performance. Model CC2 has 32 MB of cache, and Models CC4 and DC4 both have 64 MB of cache.

The cache logic in the IBM 9340 subsystem operates transparently to the host system, allowing complete compatibility between the cached subsystems and their noncached counterparts. The cache is completely controlled by the internal cache management algorithm in the IBM 9340; the software can influence the caching algorithm but cannot enable or disable the cache functions.

Cache Functions

The IBM 9340 subsystem provides cache performance without the extended functions of the IBM 3990; only basic caching is available.

The cache is divided into 48-KB track slots and is entirely managed by the Licensed Internal Code (LIC) of the IBM 9340. Any operating system that supports the IBM 9340 subsystem can benefit from the caching functions.

Track slots within the IBM 9343 cache are managed using LRU algorithm. The IBM 9340 LIC manages the cache using a technique called adaptive cache management. Adaptive caching keeps ineffective data out of cache, making more cache storage available for the higher read-hit-ratio data. The main characteristics of this technique are:

- Dynamically identifies I/O operations that are transferring large amounts of data, and disables them from using the cache.
- Attempts to recognize I/O patterns that are not benefiting from use if the cache and temporarily disables their access to the cache, enhancing use of the cache by other I/O operations.

In case of a read miss, the IBM 9343 stages data into cache starting at the record requested and ending at the end of the track. At an appropriate point, the IBM 9343 reconnects to the channel and transfers the record requested, at channel speed.

For a read hit, the IBM 9343 transfers data directly from the cache to the channel, at channel speed.

For write operations, the IBM 9343 uses the cache as a buffer. The IBM 9343 transfers data at channel speed from the channel into the cache; when the DASD is oriented to the first record to be written, the IBM 9343 writes the data from the cache to the device. The data written may or may not be retained in the cache, depending on the nature of the I/O operation.

CP Management

There are no external controls to manipulate the cache of the IBM 9340 subsystem.

However, the Define Extent CCW contains indicators to specify the caching mode: inhibit cache load (ICL), bypass cache, or sequential access mode. The IBM 9343 interprets these indicators to modify its cache management algorithm.

IBM RAMAC Array Family

IBM started a revolution in 1956 with the delivery of the direct access storage device named RAMAC. Now, there is another revolution going on with the RAMAC array family, the high-end storage subsystems that provide high performance and continuous availability with environmentally efficient packaging.

Any operating system that supports 3380, 3390, or 9345, support the IBM RAMAC Array family. The RAMAC Array family uses adaptive cache (record cache II) and, therefore, offers caching to operating systems that do not support caching.

The RAMAC Array family consists of:

- IBM RAMAC Array Subsystem,
- IBM RAMAC Array DASD
- IBM RAMAC Drawer

IBM RAMAC Array Subsystem

The IBM RAMAC Array Subsystem provides up to 90 GB of fault-tolerant storage and all storage control functions in a single physical unit. The RAMAC Subsystem offers storage in three different device geometries:

- 3380 Model K
- 3390 Model 3
- 9345 Model 2

The RAMAC Subsystem offers fault tolerance through redundant hardware and RAID 5 data storage techniques. The combination of the integrated storage control and 3.5-inch disks results in significant environmental (floor space, power, cooling) savings compared with existing 3380 or 3390 strings.

IBM RAMAC Array DASD

The RAMAC Array DASD offers up to 90 GB of fault-tolerant storage in 3390 Model 3 format. The RAMAC Array DASD attaches to IBM 3990 Model 3 or Model 6 Storage Control and takes advantages of all functions offered by the IBM 3990 Storage Controls. The RAMAC Array DASD uses RAID 5 data storage techniques and has redundant hardware to greatly improve data availability.

IBM RAMAC Drawer

The RAMAC Drawer is used in both the RAMAC Array Subsystem and the RAMAC Array DASD. The RAMAC Drawer contains four disk drives, a nonvolatile drawer cache, and multiple microprocessors to direct the independent functions of the drawer.

Every drawer has its own cache used for read caching and nonvolatile write caching. The drawer manages the cache using an LRU algorithm, essentially the same as the subsystem cache of the RAMAC family members. The cache acts as a second level cache for two or three logical volumes in the drawer. The drawer performs volume emulation, RAID 5 mapping, drawer level caching, and other functions.

Each drawer in a RAMAC Array Subsystem or RAMAC Array DASD configuration can be considered an independent parallel processing array capable of delivering significant function with little involvement from the storage control.

Drawer Disk Drive

The disk drive used is an advanced 2-GB 3.5 inch disk drive. It uses advanced technologies, including magneto-sensitive heads for high density and thin-film disks. It has a density close to that of the IBM 3390 Model 9, with a data transfer rate of 5.2 MB/s. Each disk drive has a 512 KB cache used to anticipate requests from the drawer and stage data.

Multilevel Cache

There are three levels of cache in the RAMAC family:

- Subsystem cache
- Drawer cache
- Disk drive cache

One rack contains 16 drawers, and each drawer can contain up to 64 disks, which are emulating 32 to 48 volumes. One key advantage of the multilevel cache is that every disk can always connect to a drawer cache and transfer data; RPS misses can never happen. The drawer cache can be handling a transfer for each of the emulated volumes simultaneously.

Redundant Array of Independent Disks (RAID)

Redundant array of independent disks (RAID) is any disk subsystem architecture that combines two or more physical disk storage devices into a single logical device in order to achieve data redundancy.

RAID types have been categorized into five levels, RAID-1 to RAID-5. The important levels are RAID-1, RAID-3, and RAID-5.

RAID-1 implementation employs data mirroring to achieve redundancy. Two copies of the data are created and maintained on separate volumes; identical data is written to both volumes. If data becomes unavailable from one volume, it can be obtained from the other volume. An example of RAID-1 implementation is IBM 3990 dual copy.

RAID-3 stores data blocks in parallel on multiple volumes with a segment of the data block written on each volume. A dedicated volume is used to store the parity information. In this case, the actuators always move together. All volumes are involved with every read and write operation; therefore, only one I/O operation can be processed in the subsystem at a time.

RAID-5 stores data on multiple volumes, with a portion of each volume used to maintain parity information for three other volumes. The actuators can move independently of one another; therefore, multiple concurrent accesses to the volumes are possible, thus providing high transaction throughput.

Dynamic Sparing

Dynamic sparing, when enabled, allows a drawer to notify the subsystem of an error and prevent any loss of data access by copying the data to a predefined spare volume. The process, similar to the establishing dual copy on IBM 3990 Storage Controls, maintains full access to data while (and after) the copy operation is in progress.

Activation of the dynamic sparing function is selected by the Customer Engineer at the service panel as either an automatic process or on a demand basis.

IBM RAMAC Support

This section describes the processors to which you can connect the IBM RAMAC, channels, software required, and IOCP definition.

Processors

A wide range of old and new processors can be used with the IBM RAMAC. The IBM RAMAC is supported by the following processors:

- IBM ES/9370
- IBM ES/9221
- IBM ES/9121
- IBM ES/3090
- IBM 3090
- IBM 308X

Channels

Table 7 shows the channels, the data rate, and maximum distance of the channel for the IBM RAMAC.

Table 7. IBM RAMAC Channel Support

Channel	Data Rate	Maximum Distance	IBM RAMAC Support
Parallel	3.0 4.5	122 m 400 ft	YES
3044-2 Channel Extender	3.0 4.5	245 m 800 ft	YES
ESCON	10 17	43 km 26.7 miles	YES
9034 ESCON Converter	4.5	1.2 km 3937 ft	YES
9035 ESCON Converter	-	-	NO

Software

Any operating system that supports IBM 3380 Model K, IBM 3390 Model 3, or IBM 9345 Model 2, supports the IBM RAMAC.

To initialize the new volumes on a IBM RAMAC, ICKDSF Release 16 is required.

For error reporting, EREP version 3.5.0 needs PTF UR90280.

IOCP Definition

Defining the IBM RAMAC depends on the type of storage control and device emulation. You define the IOCP macros for the IBM RAMAC family exactly as you would define the corresponding device emulated:

- IBM 3990 Model 2 with IBM 3880 Model K attached
- IBM 3990 Model 2 with IBM 3990 Model 3 attached
- IBM 9343 with IBM 9345 Model 2 attached

DASD Performance

The DASD I/O subsystem represents a capability to store data and a capability to deliver the data quickly; the DASD I/O subsystem is then configured with two purposes:

- Store system, user, and application data
- Provide satisfactory performance, even at peak times

The maximum requirement for either of these two capabilities should determine the facilities installed, but a compromise between performance and storage is normally required from a cost effectiveness point of view.

However, due to differences between the speed of processors and the time required to perform an I/O operation, a typical installation requires more DASD for performance than for data storage. Sometimes this is hard to accept, and because of that, we often see systems that are under-configured for performance; therefore, I/O subsystem performance management needs particular attention.

This section helps you identify the components of a DASD I/O operation, the factors that influence its components, and what can be done to improve the performance of each component.

DASD I/O Response Time

DASD I/O response time is one of the most important factors in system performance because it is a major contributor to the transaction response time a user sees or the time a batch job takes to complete.

The response time (T_r) of any I/O operation consists of a queue time (T_q) and a service time (T_s):

$$T_r = T_s + T_q \quad (1)$$

Service time is the time that the average I/O operation takes to complete. For part of the service time, the components are connected to the path; that is, all components (channel, storage control, internal path, and device) are busy with the I/O operation. The other part is disconnected; that is, only the device is busy with the I/O operation.

Queue time is the average time that one I/O operation has to wait because the device is busy with another I/O operation.

Response time, service time, and its components are reported by VMPRF and RTM, as discussed in “Case Study” on page 78.

Although most of the information discussed here is applicable to all DASD types, some special considerations for nonsynchronous storage controls and FBA DASD are discussed in “Nonsynchronous I/O Operations” on page 89 and in “FBA Considerations” on page 94.

In this discussion, we examine how to tune the response time for each device type and configuration; therefore, changing to faster devices or channels are not tuning recommendations here. However, this analysis is useful for an understanding of the effects of changing these facilities.

Service Time

The service time of an I/O operation is the amount of time that the I/O request takes to complete. The components of the service time are:

- Pending
- Path protocol
- Seek
- Latency
- RPS miss
- Search
- Data transfer

The device is connected during protocol, search, and data transfer, and is disconnected during pending time, seek, latency, and RPS miss (in fact, RPS misses occur because the device is disconnected). During pending time the device is logically busy (from the software point of view).

The service time as a whole may be substantially reduced by using:

- Data-in-memory facilities, which trade some I/O operations for storage accesses and paging I/O operations.
- Cached storage controls, which eliminate seek, latency, RPS miss, and search for cache hits (disconnected part of the service time).

We are assuming that the I/O operation is executed on DASD; the effect of caching on the service time is discussed in "Effect of Caching" on page 83.

Pending: Pending time is a queue time inside the channel subsystem. Although the pending time is a queue time, it is considered as part of the device service time because the I/O operation has already been scheduled and the real device block is flagged as busy.

CP issues the I/O operation (SSCH or RSCH) because the corresponding RDEV block is free, as discussed in "I/O Scheduling" on page 42. However, the channel subsystem may not start the I/O operation for the following reasons:

- The path to the device is busy.
- The device is busy with another host.
- The device is reserved (Reserve CCW) by another host.
- The physical device is staging or destaging data.

The solution for the pending time is quite complex; therefore, we limit this analysis to single hosts (no shared DASD) and noncached storage controls. The paths can be considered the servers in a multiple-server system, as discussed in "M/M/c Model" on page 211, and the pending time can be estimated by:

$$T_{pend} = \frac{T_{path}}{c(1 - U_{ch})} PPB \qquad (2)$$

where
- PPB = probability of path busy
- T_{path} = path service time
- c = number of paths
- U_{ch} = average channel utilization

The average channel utilization can be estimated by adding the utilizations of all channels in the path and dividing by the number of channels.

The probability of a path being busy depends on the utilization of each component of the path, such as channels, storage controls, and internal paths, as discussed in “Path Utilization” on page 74 (see also Chapter 8, “Probability Theory Refresher” on page 151 to review the basic concepts). We can use effective path utilization as a synonym for probability of path busy.

The effective path utilization can be reduced by:

- Adding more paths to the device.
- Balancing the load over all paths in the system by moving some active areas to other devices under other paths.

Using channels with higher speeds would not help much here, because the device normally transfers data at device speed regardless of the speed of the channel. Potential exceptions to this are cached storage controls and nonsynchronous storage controls, as discussed in “Effect of Caching” on page 83 and in “Nonsynchronous I/O Operations” on page 89.

IBM 3380s and 3390s connected with four paths in a DLSE configuration and IBM 9340 subsystems seldom have large pending times. However, some time that is reported as connected time in parallel channel environments is reported as pending time in ESCON environments.

Path Protocol: This is the time necessary for the initial and ending protocols; you can use 1 millisecond as an average.

Seek: Seek time is the time spent by the actuator to move from the current cylinder to the desired cylinder. It depends on two factors:

- How fast the actuator can move across the volume.
- How far the actuator must move between two successive requests.

The first component cannot be tuned; the second component can be tuned by:

- Moving the most active areas closer to each other.
- Reducing the number of active areas.

The VMPRF reports (PRF056, PRF057, and PRF058) can be used to determine the active areas and the best place to move them.

The average seek is the expected value of the seek time, and it is about the time the actuator takes to travel 1/3 of the whole volume considering a random distribution of seeks (see “DASD Seek Model” on page 175). Typical applications, however, have many zero-length or short-length seeks.

In a typical VM environment, you can expect an average seek of about 1/4 to 1/3 of the average seek for the device.

Latency: Latency is the time that it takes the device to rotate until the desired record is under the actuator. It is constant for each type of DASD, and therefore, cannot be tuned.

Since the current position of the actuator could be in any place with the same probability, we assume half a revolution time for latency.

RPS Miss: Rotational Position Sensing (RPS) is a facility that allows the storage control to sense the current position (sector) in the track. The device can then disconnect from the entire path during its orientation until the required record comes under the actuator.

The disconnect during rotation must be initiated by the software in case of CKD DASD. In case of FBA DASD, this is done automatically, as discussed in "FBA Considerations" on page 94.

When the device tries to reconnect (normally two or three sectors before the record) the path may be busy and one revolution is missed. Two, three, or more revolutions may be missed depending on the path utilization. This component can be estimated, as discussed in "RPS-miss Time" on page 178, by:

$$T_{rps} = \frac{PPB}{1 - PPB} T_{rev} \qquad (3)$$

where
- PPB ◦ probability of path busy
- T_{rev} ◦ full revolution time

RPS-miss time increases very quickly as the effective path utilization increases. When path utilization triples, say from 20% to 60%, the RPS miss time increases by a factor of six, as shown in Table 8. The RPS expansion factors multiplied by the DASD revolution time are the RPS-miss times for that DASD type.

Table 8. Effect of Path Busy on RPS-Miss Time

Path Busy (%)	RPS Expansion Factor	RPS-miss Time (ms)		
		3380	3390	9345
10	0.111	1.8	1.6	1.2
20	0.25	4.2	3.5	2.8
30	0.429	7.1	6.0	4.8
40	0.667	11.1	9.4	7.7
50	1.0	16.6	14.1	11.2
60	1.5	24.9	21.2	16.8
70	2.33	38.7	32.9	26.1

The effective path utilization can be reduced by:

- Adding more paths to the device.
- Balancing the load over all paths in the system by moving some active areas to another devices under other paths.

IBM 3380s and 3390s connected with four paths in a DLSE configuration and IBM 9340 subsystems seldom have large RPS-miss times. Having enough paths to a device provides not only performance but consistency in I/O response time (see "Case Study" on page 78). The ability to provide this consistency protects the installation from severe perturbations in performance during peak periods.

The IBM 3990 Storage Control and the IBM 9340 subsystem provide an RPS Miss Avoidance (RMA) function that reduces the RPS-miss time, as discussed in "Nonsynchronous I/O Operations" on page 89.

Search: RPS is able to sense the sector under the actuator, and the storage control requests the reconnection to the channel two or three sectors before the sector addressed by the channel program.

When the device connects to the path, the record may still not be properly positioned to be accessed. The search function ensures that the record processed is the addressed record, and we assume it takes two sectors on average.

Because there are 224 sectors in a 3390 track, and the 3390 revolution time is 14.1 milliseconds (twice the latency), the search time for an IBM 3390 Model 1 is:

$$T_{search} = \frac{2 \times 14.1}{224} = 0.13$$

Data Transfer: Data-transfer time is a function of angular speed, track density, and block size.

Reducing the block size reduces the data-transfer time of one I/O operation. For most cases, the corresponding increase in the number of I/O operations would substantially increase the total I/O time. In fact, for most applications, increasing the block size has the most significant tuning effect on total I/O time, because of the reduction in the total number of I/O operations.

Some applications that process data randomly may be exceptions because the larger block sizes increase the I/O time without reducing the total number of I/O operations. However, even in random accesses, there may exist a locality of reference for the data. Therefore, the data pattern should be carefully analyzed before deciding to reduce the block size. Since data transfer is a small component of the total service time, saving some time on data transfers may not offset a possible increase in the number of I/O operations.

You can estimate the data transfer time dividing the number of bytes (plus gaps) of data by the device speed. You can also find the data transfer time of one record by dividing the full rotation time by the number of records per track; for example, 4 KB in a 3990 is transferred in about 1.175 (14.1/12) milliseconds.

Queue Time

The queue time is the amount of time an I/O operation has to wait (in a software queue) before it gets started because the device is busy with another I/O operation (initiated by the software in the same host).

The best way to estimate the queue time is through the following equation, known as Little's Law (see “Little's Law” on page 184 for details):

$$T_q = \frac{N_q}{IORATE}$$

where N_q ◦ number of requests in queue
IORATE ◦ I/O operations per second

Do not forget to adjust the units because the I/O rate is normally expressed in seconds and the queue time in milliseconds.

For example, if there are 15 I/O operations per second and an average of 0.3 SSCHs in queue for a device, the average queue time in milliseconds is:

$$T_q = \frac{0.3 \cdot 1000}{15} = 20$$

For a simplified model (called M/M/1), Little's Law can be reduced to the following equation, which is very useful for studying the effect of device utilization on the queue time:

$$T_q = \frac{U_{dev}}{1 - U_{dev}} T_s \qquad (4)$$

where U_s = device utilization as seen by this host
T_s = device service time

Equation (4) is a fair estimate of the queue time; however, the results tend to be conservative with moderate to high loads. See “M/M/1 Model” on page 199 for a discussion of this model.

Equation (4) is similar to the equation for RPS-miss time, and therefore, the queue time increases very quickly with device utilization. However, an application reading a sequential data set (a DDR backup, for example), may drive a device to high utilization without creating contention if the device is not used by other applications.

The device utilization should be below 30% to avoid large queue times, as shown in Table 9.

Table 9. Effect of Utilization on Queue Time (M/M/1 Model)

Device Service Time	Device Utilization and Queue Time						
	10%	20%	25%	30%	40%	50%	60%
10	1.1	2.5	3.3	4.3	6.7	10.0	15.0
15	1.7	3.8	5.0	6.4	10.0	15.0	22.5
20	2.2	5.0	6.6	8.6	13.3	20.0	30.0
25	2.8	6.3	8.3	10.7	16.7	25.0	37.5
30	3.3	7.5	10.0	12.9	20.0	30.0	45.0
35	3.9	8.8	11.7	15.0	23.3	35.0	52.5
40	4.4	10.0	13.3	17.1	26.7	40.0	60.0

One way of reducing the queue time is to reduce the device utilization by moving active areas to other devices. The VMPRF reports (PRF056, PRF057, and PRF058) should be used to determine these areas and to which devices they should be moved.

Guest operating systems queue their own I/O requests before sending them to CP; therefore, no I/O queue condition under CP does not necessarily mean that the device has no contentions (see the discussion in “Little's Law or M/M/1?” on page 82).

Quick Queue Time Estimate

Using equations (1) and (4) for the response time and M/M/1 queue time, respectively, we can write:

$$T_r = T_s + T_q = T_s + \frac{U_{dev}}{1 - U_{dev}} T_s$$

$$\therefore \quad T_r = \frac{T_s}{1 - U_{dev}} \tag{5}$$

Rearranging the terms of equation (5), we can write:

$$\begin{aligned} \frac{T_s}{T_r} &= 1 - U_{dev} \\ \frac{T_q}{T_r} &= U_{dev} \end{aligned} \tag{6}$$

Therefore, if the device utilization is 60%, you can deduce immediately that the queue time accounts for 60% of the response time and the service time accounts for 40% of the response time. An easy way to remember the system of equations (6) is by plotting the second equation, as shown in Figure 22.

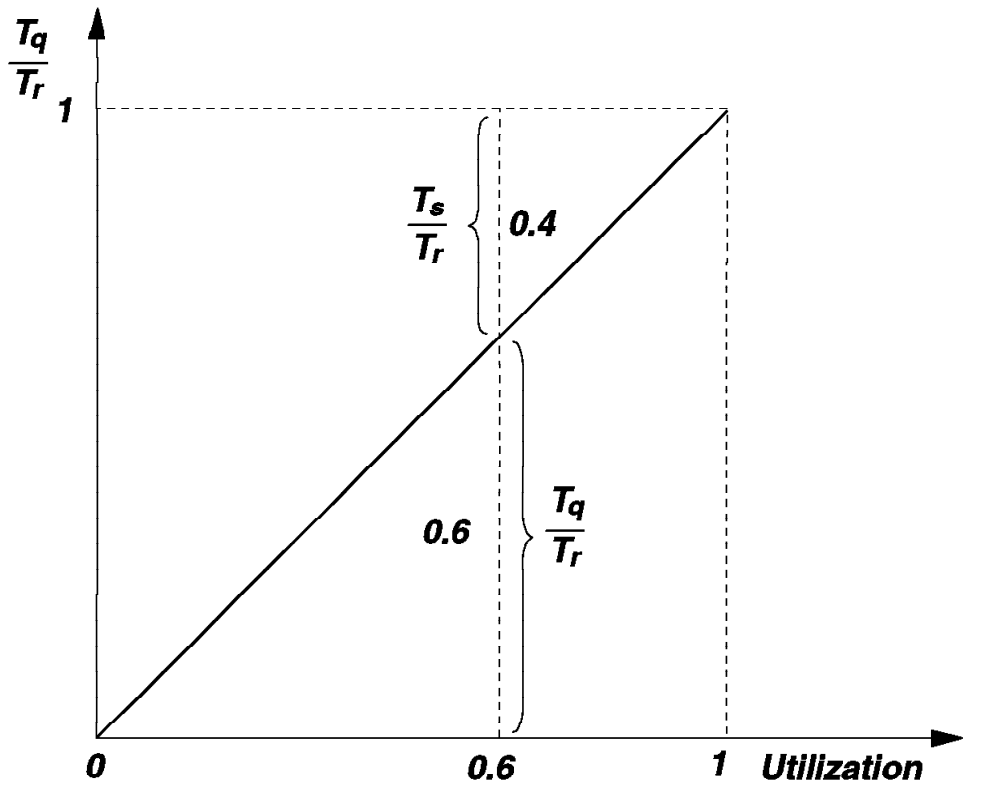

Figure 22. Service Time, Queue Time, and Utilization

Device Utilization

Device utilization is one of the key variables used when analyzing DASD performance, because it is an indication of possible contention.

Both RTM and VMPRF report the device utilization; you may check, as an exercise in understanding the variables, that equation (7) normally holds:

$$U_{dev} = IORATE \cdot T_s \quad (7)$$

where
$IORATE$ = I/O operations per second
T_s = service time

For example, if there are 20 I/O operations per second and the average service time is 15 milliseconds, the device utilization is:

$$U_{dev} = 20 \cdot \frac{15}{1000} \cdot 100 = \frac{20 \cdot 15}{10} = 30\%$$

The device is busy for 300 milliseconds during an interval of 1000 milliseconds. In other words, the device utilization is the probability of device being busy.

Shared DASD

For shared DASD, each host reports the device utilization as seen by each image. The device appears as busy to more than one processor at the same time because the real device block is flagged as busy when the I/O operation is scheduled.

You can estimate the actual device utilization as the probability of the device being flagged as busy by one processor, by the other processor, or by both. Since these events are not mutually exclusive (see Chapter 8, “Probability Theory Refresher” on page 151), the actual utilization is:

$$U_{dev} = U_{proc1} + U_{proc2} - U_{proc1} \cdot U_{proc2} \quad (8)$$

where
U_{proc1} = device utilization reported by system 1
U_{proc2} = device utilization reported by system 2

This actual device utilization can give you an idea of what the contentions are on a particular DASD volume. You can extend this to any number of systems by combining the result for two systems with a third, and so on.

However, the device utilization seen by one host is responsible for the queue time (queue inside the software), and the device utilization seen by the other host is responsible for pending time.

The current IBM 3990 and IBM 9340 storage controls handle requests to the same volume from multiple hosts in a round-robin manner. Other IBM storage controls select from outstanding requests in a way that favors the host with the active request. The effect of this is that a dominant host can virtually block the other host's activity.

Path Utilization

The VMPRF PRF013 report provides information about the utilization of individual channels.

When there are several paths to the device you have to estimate the probability of the entire path being busy (effective path utilization).

However, different capabilities, such as number of data paths, internal data path contention, and DPR makes the estimation of effective path utilization different in each case.

This section describes how to estimate the effective path utilization for the most common storage controls and DASD types.

Probability of Path Busy

Effective path busy is calculated as the probability of the entire path being busy, which depends on the DASD subsystem capabilities, such as concurrent data paths and dynamic reconnect.

DPR-capable Devices: All IBM 3380 models, all IBM 3390 models, and all IBM 9345 models are DPR-capable devices. This means that the probability of path busy is the same when the path is selected and when the device tries to reconnect to the path. Their storage control are multiple-server systems and the effective path busy can be estimated by the Erlang-C formula (see "M/M/c Model" on page 211 for a discussion of this model).

When there are two data paths, the Erlang-C formula is:

$$PPB_2 = \frac{2U_{ch}^2}{1 + U_{ch}} \qquad (9)$$

where PPB_2 = probability of path busy (two data paths)
U_{ch} = average channel utilization

As an example, assume you have 3380s Model D or E with a DLS attachment and an average channel utilization of 40%:

$$PPB_2 = \frac{2 \times 0.4^2}{1 + 0.4} \quad 0.23 = 23\%$$

When there are three data paths, the Erlang-C formula is:

$$PPB_3 = \frac{27U_{ch}^3}{6 + 12U_{ch} + 9U_{ch}^2} \qquad (10)$$

where PPB_3 = probability of path busy (three data paths)
U_{ch} = average channel utilization

As an example, assume you have 3380s Models J or K or 3390s, with a DLSE attachment and with an average channel utilization of 49%, and that one of the four paths is unavailable:

$$PPB_3 = \frac{27 \cdot 0.49^3}{6 + 12 \cdot 0.49 + 9 \cdot 0.49^2} \approx 0.23 \approx 23\%$$

When there are four data paths, the Erlang-C formula is:

$$PPB_4 = \frac{32U_{ch}^4}{3 + 9U_{ch} + 12U_{ch}^2 + 8U_{ch}^3} \quad (11)$$

where PPB_4 ≈ probability of path busy (four data paths)
U_{ch} ≈ average channel utilization

As an example, assume you have 3380s Models J or K or 3390s, with a DLSE attachment and with an average channel utilization of 55%:

$$PPB_4 = \frac{32 \cdot 0.55^4}{3 + 9 \cdot 0.55 + 12 \cdot 0.55^2 + 8 \cdot 0.55^3} \approx 0.23 \approx 23\%$$

The average channel utilization can be estimated by adding the utilizations of the channels and dividing by the number of channels.

Table 10 lists the effective path utilization for different numbers of paths.

Table 10. Effective Path Utilization (M/M/c Model)

Average Channel Utilization	Number of Paths			
	1	2	3	4
10	10.00	1.818	0.370	0.079
15	15.00	3.913	1.139	0.349
20	20.00	6.667	2.466	0.958
25	25.00	10.00	4.412	2.041
30	30.00	13.85	7.003	3.705
35	35.00	18.15	10.24	6.030
40	40.00	22.86	14.12	9.070
42	42.00	24.85	15.84	10.49
45	45.00	27.93	18.61	12.85
50	50.00	33.33	23.68	17.39
55	55.00	39.03	29.32	22.68
60	60.00	45.00	35.47	28.70
70	70.00	57.65	49.23	42.87
80	80.00	71.11	64.72	59.64
90	90.00	85.26	81.71	78.76

IBM 3380 Standard: The IBM 3380 Standard is a DPR-capable device that can sustain two simultaneous data transfers but it is subject to internal path contention. A full string has 16 devices; each internal path is shared by four devices in the following manner:

- Path 0 - devices 0, 1, 8, 9
- Path 1 - devices 2, 3, A, B
- Path 2 - devices 4, 5, C, D
- Path 3 - devices 6, 7, E, F

The path is busy when both channels are busy (PPB_2, as defined in the previous topic) or the internal path is busy. Because the events are not mutually exclusive, the effective path utilization is:

$$U_{path} = U_{ip} + PPB_2 - U_{ip} \times PPB_2$$

$$PPB_2 = \frac{2U_{ch}^2}{1 + U_{ch}}$$

where U_{ip} = internal path utilization

U_{ch} = average channel utilization

The average channel utilization is estimated as before, adding the utilization of the two channels and dividing by two.

The internal path utilization can be estimated considering that:

- The internal path is busy when any of the other devices sharing the path is using it.
- One device uses the internal path during the search and data transfer.
- When one device is using the internal path, no other device can use it (mutually exclusive events).

Therefore, the internal path utilization can be estimated adding the contributions of all devices sharing the internal path except the device under consideration. The device under consideration should not be counted because when the internal path is busy due to that device, the device is not competing for the internal path.

You can use equation (7) discussed in “Device Utilization” on page 73 to estimate the utilization of the internal path by each device:

$$U_{ip} = \frac{IORATE \times T_{conn}}{1000}$$

where IORATE = I/O operations per second

T_{conn} = connect time

IBM 3375: A typical 3375 configuration is shown in Figure 23. The A and D models are head-of-string devices and contain the logic to manage one data transfer each; therefore, two simultaneous data transfers are possible.

Because the 3375 is not DPR-capable, it is necessary to distinguish between two path busy conditions:

- When estimating the pending time, the path is busy when both channels are busy (PPB_2) or the internal path is busy:

  $$U_{path} = U_{ip} + PPB_2 - U_{ip} \times PPB_2$$

  where $$PPB_2 = \frac{2U_{ch}^2}{1 + U_{ch}}$$

  U_{ch} = average channel utilization

  Since the 3375 internal path is shared between two devices (0 and 1; 2 and 3; 4 and 5; 6 and 7), the internal path utilization is the contribution of the other device of the pair, as described previously.

- When estimating the RPS-miss time, the path is busy when just one channel is busy (no DPR) or the internal path is busy:

  $$U_{path} = U_{ip} + U_{ch} - U_{ip} \times U_{ch}$$

  where U_{ch} = average channel utilization

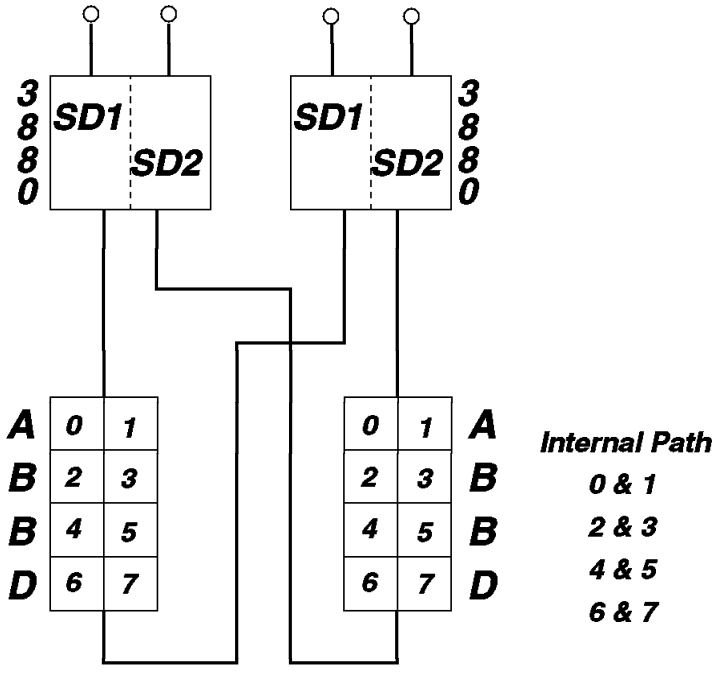

Figure 23. Typical 3375 Configuration (A-B-B-D)

Case Study

This section describes how to calculate the response time components and how to identify these components in the RTM and VMPRF reports. Although this example is for a 3390-1, you can use the same methodology for other DASD types and conditions.

Table 11 lists the components for the following case:

- IBM 3390-1 attached to 3990-2 in DLSE mode
- Device utilization is 25%
- No DASD is shared among real processors
- Effective path utilization is 10% (42% each of four paths)
- Two thirds of all seeks are zero-length seeks

Table 11. DASD I/O Response Time Components - Example

I/O Component	Time (ms)	Form of Calculation	Comments	Page
Pending	0.10	2.28*0.1¸ 4(1-0.42)	M/M/c queue time	67
Seek	3.17	9.5¸3	1/3 of average seek	68
Latency	7.10	Half revolution	Rotational delay	68
RPS miss	1.57	14.1*0.10¸(1-0.10)	Lost revolutions	69
Path protocol	1.00	-	Initial and ending	68
Search	0.13	2*14.1¸224	2 sectors before	70
Data transfer	1.15	4862¸4220	4 KB plus gap	70
Connect time	2.28	1.00+0.13+1.15	Path is busy	67
Disconnect time	11.84	3.17+7.10+1.57	Path is free	67
Service Time	14.22	0.10+2.28+11.84	Device Busy	67
Queue time	4.74	14.22*0.25¸(1-0.25)	M/M/1 queue time	70
Response time	18.96	14.22+4.74	Queue + Service	66

Assume that one of the four paths is lost, either because of hardware problems or because a transaction tied up one path.

Using the Erlang-C formula or Table 10 on page 75, you can find that the new effective path utilization would be about 16% (assuming the same channel utilization over the three remaining paths).

The changed components would be:

```
pending    =  2.28  * 0.16 ÷ 3(1 - 0.42)  =  0.21
RPS miss   =  14.1  * 0.16 ÷  (1 - 0.16)  =  2.69
disconnect =  3.17  + 7.10 + 2.69         =  12.96
service    =  0.21  + 2.28 + 12.96        =  15.45
queue      =  15.45 * 0.25 ÷ (1 - 0.25)   =  5.15
response   =  15.45 + 5.15                =  20.60
increase   = (20.60 - 18.96) ÷ 18.96      =  8.6%
```

The increase in response time would be only 8.6% and the configuration could still provide a good response time for transactions. Therefore, 4-path configurations are good solutions for providing consistent response times.

RTM Variables

Figure 24 shows the RTM DEVICE screen; the variables under the heading "Measurement Facility" come directly from the channel subsystem.

```
<>VM/ESA CPU9121 SERIAL 420415  253M DATE 09/12/93 START 11:00:49 END 11:48:38<>
 <--- DEVICE ---> <----- DEVICE  RDEV  DATA ------> <-- MEASUREMENT FACILITY ->
  DEV TYPE VOLSER IOREQST SEC  %Q %ER R %LK  LNK PA %UT  ACC  FPT  DCT   CN %CN
 0960 3390 D14AA1   15051   5 .00 .00   .00   40  2 2.5    4    0    1    3 1.6
 0961 3390 D14AA2   17340   6 .00 .00   .01   36  2 2.7    4    0    1    3 1.8
 0962 3390 D14AA3   10533   3 .00 .00   .00   24  2 1.6    4    0    1    3 1.1
 0963 3390 D14AA4    4732   1 .00 .00   .01   19  2 .77    4    0    1    3 .50
 0966 3390 D14AA7    6481   2 .00 .00   .00   11  2 1.0    4    0    1    3 .71
 0967 3390 D14AA8    5754   2 .00 .00   .01   15  2 .95    4    0    1    3 .64
 0968 3390 D14AA9    4942   1 .00 .00   .00   10  2 2.7   16    0   13    2 .50
 096C 3390 SET1C0   35471  12 .07 .00   .03  539  2  20   16    0   12    4 5.4
 096D 3390 SET1C1    7497   2 .00 .00   .00  220  2 4.2   16    0   11    4 1.2
 096E 3390 SET2C0   36942  12 .09 .00   .04  529  2  20   16    0   12    4 5.2
 096F 3390 SET2C1    7563   2 .00 .00   .02  216  2 4.6   17    0   12    5 1.3
```

Figure 24. RTM DEVICE Screen

The most important variables are:

ACC Device service time.

FPT Pending time.

DCT Disconnect time.

CN Connect time.

%CN The percentage of time the device is connected to the path; it is an indication of how much this device is using the path.

%Q The percentage of I/O requests queued for the device.

%UT Device utilization.

IOREQST Total number of I/O requests.

SEC I/O rate expressed as I/O requests per second.

You can estimate the queue time and response time using equation (5) described in "Queue Time" on page 70; for device X'096E':

$$T_q = \frac{\%UT}{100 - \%UT} ACC = \frac{20}{100 - 20} \times 16 = 4$$

$$T_r = ACC + T_q = 16 + 4 = 20$$

You may verify that equation (7) discussed in "Device Utilization" on page 73 holds within the limits of the precision of the report. Note that the I/O rate is reported as a whole number, and therefore, some deviations may occur.

$$\%UT = SEC \times \frac{ACC}{10} \qquad (12)$$

You can calculate a better value for SEC by dividing IOREQST by the difference between starting and ending times reported in the top right corner in Figure 24.

VMPRF Variables

Figure 25 shows one of the VMPRF reports that provides the I/O response time components (PRF012).

```
PRF012  Run 09/08/1993 08:17:33        DASD_BY_ACTIVITY                                          Page   25
                                 DASD Activity Ordered by Activity
From 09/08/1993 07:10:16                                                          GG243934
To   09/08/1993 07:12:16                                                          CPU 9121     SN  20415
For    120 Secs 00:02:00              VM/ESA STORAGE MANAGEMENT MEASUREMENTS          VM/ESA   21.01 SLU 9303
_____________________________________________________________________________________________________

<-----Device------>                          <--SSCH+RSCH-->      <--------------Time--------------> <-SSCHs in>
                                                                                                    <--Queue-->

                                 Mini-    On-
Num- Volume        Control          disk   line                Pct
 ber Serial Type    Unit    Owner    Links   Secs    Count   Rate  Busy   Pend   Disc   Conn   Serv   Resp  Mean   Max Err

096E SET2C0 3390-2  3990-3             65    120     3604   30.0  45.2    0.5   11.2    3.4   15.1   36.7   0.6   1.2   0
096C SET1C0 3390-2  3990-3             63    120     3481   29.0  43.1    0.3   11.2    3.4   14.9   22.9   0.2   0.4   0
0961 D14AA2 3390-2  3990-3              9    120      167    1.4   0.9    0.1    3.7    2.7    6.5    6.5     0     0   0
0960 D14AA1 3390-2  3990-3              7    120       87    0.7   0.6    0.1    4.8    3.1    8.0    8.0     0     0   0
0967 D14AA8 3390-2  3990-3              8    120       64    0.5   0.5    0.1    5.8    3.2    9.1    9.1     0     0   0
0963 D14AA4 3390-2  3990-3              4    120       44    0.4   0.2    0.1    3.3    2.8    6.2    6.2     0     0   0
0965 D14AA6 3390-2  3990-3              3    120       37    0.3   0.2    0.1    3.7    2.7    6.5    6.5     0     0   0
```

Figure 25. VMPRF PRF012 Report

The most important variables are:

PEND Pending time

DISC Disconnect time

CONN Connect time

SERV Device service time

RESP Device response time

PCT BUSY Device utilization

COUNT Total number of I/O requests

RATE I/O rate expressed as I/O requests per second

Seek time is not reported; you can estimate it by subtracting the RPS-miss time and latency (easy to estimate) from the disconnect time.

VMPRF uses Little's Law to calculate the queue time, but beware of the precision of the average queue time reported (variable MEAN). The response time in milliseconds for the device X'096E' in Figure 25 is calculated as:

$$T_r = SERV + \frac{MEAN \cdot 1000}{RATE} = 15.1 + \frac{0.6 \cdot 1000}{30} = 35.1$$

To get the same number as reported by VMPRF (36.7), the variable MEAN should have been 0.648 instead of 0.6.

You may verify that equation (7) discussed in “Device Utilization” on page 73 holds within the limits of the precision of the report:

$$PCT\_BUSY = RATE \cdot \frac{SERV}{10} \qquad (13)$$

DASD Tuning Recommendations

You should try to find contentions in the I/O subsystem in order to improve response time. General guidelines to analyze I/O contentions include the following:

- Investigate any number other than zero on the device I/O queue.
- Contention may be caused by a high device utilization; investigate any device with utilization greater than 30%.
- If the device utilization (and requests in the queue) is high, look at the service time and the I/O rate:
  - If the I/O rate is high, move some minidisks to other devices.
  - If the service time is high, determine which I/O components are causing the inflated value:
    - High path utilization causes high pending and RPS-miss times.
    - Bad disk layout causes high seek times.
  - If the service time and I/O rate are low, contention may be caused by shared DASD; this will be indicated by high pending times.
- Looking at a single number in isolation from all others may lead to invalid conclusions. For example, I/O rate is a work-load measure and indicates how much work is being done by the device or subsystem.

  After a change, look at both the response time and the I/O rate; the change is good if the response time decreases for the same I/O rate.

  If response time for a particular device increased, and the I/O rate also increased, further evaluation is necessary. Response time may have been far higher if the change had not been made.

Rules-of-thumb (ROTs) normally provide thresholds that should trigger investigations when exceeded. In that sense they are very useful, and in fact, we do provide a lot of these ROTs in this document.

However, you should use ROTs as guidelines to see whether further investigation is required; but do not rely on these simple mystery numbers when you can calculate better values. In other words, you should understand the underlying processes the ROTs are derived from, in such a way that you are able to determine when you have a problem and how to fix it.

Do not compare a device running a very heavy work load to the same device running a very light work load. Response will likely drop without changes simply due to the lighter work load.

Ensure you have enough samples to validate the data; if a device has performed only 500 I/O operations in a 24-hour period, the sample size is not sufficient to make a comparison to a 24-hour period where it does 100,000 I/O operations.

Look at sustained peaks (somewhere between 15- and 60-minutes intervals) rather than 24-hour averages; look at 8-hour shifts rather than 24-hour periods.

Little's Law or M/M/1?

We have discussed two ways to estimate the queue time for a DASD I/O operation; namely, using Little's Law or the M/M/1 model. If you have enough data to calculate the queue length using both methods, you may obtain different results with each method. This section discusses the advantages and disadvantages of these two methods.

Little's Law

The main advantage of Little's Law when evaluating the queue time is that it does not depend on the model assumed for the I/O subsystem. In other words, Little's Law holds for any distribution of the I/O random variables. If you know the I/O rate (λ) and the average number of requests in the queue ($\overline{N}_q$), you can evaluate the average queue time by:

$$\overline{T}_q = \frac{\overline{N}_q}{\lambda}$$

The main problems when applying Little's Law to the VM/ESA DASD I/O subsystem are:

- The average number of I/O requests in the queue is not reported by all performance tools.
- Part of the I/O request queue may be hidden inside guest operating systems.
- CP maintains different queues for paging/spooling and other I/O requests. When a device contains paging areas and user data, you have to look at the PRF012 report (user data) and PRF088 (paging data), but there is no easy way to combine the results.

You can use Little's Law with good results when evaluating queue times in CMS environments for devices that only contain user data.

M/M/1

The greatest advantage of the M/M/1 queue time equation is its simplicity. You need the utilization of the device (ρ) and the average service time, but these values are normally reported by all VM/ESA performance tools, as provided by the channel subsystem. You can evaluate the queue time by using:

$$\overline{T}_q = \frac{\rho}{1 - \rho}\overline{T}_s$$

Since device utilization and service time are both calculated by the channel subsystem, they are always correct regardless of hidden queues or the way CP queues its I/O requests.

The problem with the M/M/1 equation is its own nature, since it depends on the I/O subsystem following a theoretical model with exponential distributions (see “Birth-Death Processes” on page 188), which does not happen all the time. Higher deviations from the M/M/1 model occurs at moderate to high device utilizations (see “M/G/1 Model” on page 239).

The M/M/1 model can be used with reasonable accuracy when the device utilization is low (20% or less).

Effect of Caching

The cache eliminates seek, latency, RPS-miss, and search times when the requested record is found in cache (cache hit); that is, for a cache hit, the I/O response time components are reduced to:

- Path protocol
- Pending
- Data transfer

Disconnect time is then eliminated for a cache hit; and connect time is reduced because the search component is eliminated and because data is transferred at channel speed. Therefore, the key indicator of effective cache performance is a low disconnect time.

Estimating Cache Service Time

For any subsystem that uses cache, you can estimate the average service time for I/O operations using the following equation:

$$T_s = T_{cache} \times HIT + T_{nocache}(1 - HIT)$$
$$= T_{cache} \times HIT + T_{nocache} - T_{nocache} \times HIT$$
$$= T_{nocache} - HIT \times (T_{nocache} - T_{cache})$$

where: HIT = probability of a cache hit

Therefore, the reduction in service time is the amount of time saved by the cache multiplied by the probability of a cache hit.

If you are examining a basic cache case, the probability of a hit (basic cache effectiveness) can be calculated as the probability of a read operation multiplied by the probability of a cache read hit:

$$HIT = READ \times READ\_HIT \quad (14)$$

Conversions between the probability of a read and the read-to-write ratio are made with the following equations:

$$RATIO = \frac{READ}{1 - READ} \qquad READ = \frac{RATIO}{1 + RATIO}$$

In case of cache extended functions, the probability of a hit (extended functions effectiveness) is the union of two mutually exclusive events, each event composed of two independent events:

- A read operation and a read hit
- A write operation and a write hit

Therefore, the probability of a hit is the sum of the probabilities of these two events (see Chapter 8, “Probability Theory Refresher” on page 151):

$$HIT = READ \times READ\_HIT + WRITE \times WRITE\_HIT \quad (15)$$

The probability of a hit is also known as the cache hit ratio; a hit may be further classified as read hit, write hit, DFW hit, or CFW hit. The probability of a read operation is the percentage of read operations. Be aware that different performance tools use slightly different definitions for these variables.

Basic Caching Report

The VMPRF PRF043 report provides one line for each cached storage control, but only basic cache information is reported, as shown in Figure 26. Other VMPRF reports related to basic cache are PRF016, PRF041, PRF042, and PRF044 which report basic cache information for each DASD volume.

The percentage of read operations, read cache hits, and read misses are provided as variables READ, CACHE_HITS, and READ_MISS, respectively

```
PRF043  Run 09/08/1993 11:50:27       CACHE_CONTROLLER                                    Page   27
                                 CACHE Controller Activity
From 09/08/1993 11:01:27                                                    GG243934
To   09/08/1993 11:03:27                                                    CPU 9121     SN  20415
For    120 Secs 00:02:00            VM/ESA STORAGE MANAGEMENT MEASUREMENTS          VM/ESA   21.01 SLU 9303
_____________________________________________________________________________________________________

          <-Cache Size>  <-SSCH+RSCH->      <-----------Time------------>      <----------Percent---------->

Stg                                                                 Mean                Inhib
Ctlr Control   Con-                    Pct                             SSCH      Cache  Read it/By
ID   Unit      fig  Avail    Count  Rate  Busy  Pend  Disc  Conn  Serv   Resp Queue  Read  Hits  Miss  pass   Seq

0006 3880-23   32MB   31MB      235   2.0   4.1   2.2  11.0   8.0  21.1   21.1     0  88.7  35.7  64.3     0    0
000F 3880-23   32MB   31MB     2009  16.7   7.8   0.5   1.0   3.2   4.6    4.6     0  99.3  94.0   6.0     0    0
0016 3990-3   256MB  255MB     7810  65.1  95.8   0.4  10.9   3.4  14.7   30.8   1.2  95.1  10.1  89.9     0    0
0018 3990-3   256MB  255MB      995   8.3  10.9   0.1  10.0   2.9  13.1   17.1   0.1  99.9     0 100.0     0    0

Sum/Mean                   11049  92.1 118.5   0.4   9.0   3.4  12.9   24.6   1.2  97.5  14.3  85.7     0    0
```

Figure 26. VMPRF PRF043 Report

The DASD average response time components are also reported, so you can quickly relate the disconnect time to cache usage. In fact, storage control 000F is the only one with a low disconnect time, which is a consequence of its high cache hit ratio. The relevant variables are calculated by VMPRF as:

$$\text{READ} = \frac{\text{Cache Reads}}{\text{Cache Reads} + \text{Cache Writes}} \times 100$$

$$\text{CACHE\_HITS} = \frac{\text{Cache Read Hits}}{\text{Cache Reads} + \text{Cache Inhibits} + \text{Cache Bypasses}} \times 100$$

$$\text{READ\_MISS} = \frac{\text{Cache Reads} - \text{Cache Read Hits}}{\text{Cache Reads} + \text{Cache Inhibits} + \text{Cache Bypasses}} \times 100$$

$$\text{INHIBIT/BYPASS} = \frac{\text{Cache Inhibits} + \text{Cache Bypasses}}{\text{Cache Reads} + \text{Cache Inhibits} + \text{Cache Bypasses}} \times 100$$

Note the following:

- Variables CACHE_HITS, READ_MISS, and INHIBIT/BYPASS add up to 100%.
- The percentage of read operations is calculated relative to the total number of cached I/O operations; it does not include the cache inhibit and bypasses.
- The cache hit ratio is relative to the number of read operations and not to the total number of I/O operations, but it includes all read operations, both cached and noncached.
- You should check volumes with cache read hit ratios less than 80%.

The basic cache effectiveness (equation (14)) for storage control X'0006' is:

basic cache effectiveness = 35.7 ´ .887 = 31.7%

Basic Caching Example

As an example, assume that an application program executes 1000 I/O operations in basic cache mode, distributed in the following way:

- 1000 I/O operations
  - 800 reads
    - 700 hits
    - 100 misses
  - 200 writes
- Average service time for a miss is 15 milliseconds
- Average service time for a hit is 2 milliseconds

Since there are no cache inhibits and bypasses, the relevant variables are (multiplication by 100 has been dropped):

$$\text{READ} = \frac{800}{1000} = 0.80$$

$$\text{TOTAL\_HITS} = \frac{700}{1000} = 0.70$$

$$\text{CACHE\_HITS} = \frac{700}{800} = 0.875$$

$$\text{READ\_MISS} = \frac{100}{800} = 0.125$$

$$T_S = 15 - [0.70 \times (15 - 2)] = 5.9$$

Note: Total cache hits (TOTAL_HITS) is not reported by VMPRF.

The average service time was calculated using the total cache hit ratio, but it could also be calculated with the read hit ratio because, as explained before:

$$\text{TOTAL\_HITS} = \text{READ} \times \text{CACHE\_HITS} = 0.80 \times 0.875 = 0.70$$

Assuming that instead of 100 cache misses, the application program had 50 cache misses and 50 cache bypasses, VMPRF would have reported:

$$\text{READ} = \frac{750}{950} = 0.79$$

A cache miss and a cache bypass have almost the same effect on the average service time. However, too many misses may impact the overall performance because the physical device is kept busy during the stages of data. The probability of a read operation would still be 0.80 (total reads divided by total I/O operations), but not reported as such by VMPRF. The average service time would have been the same as calculated before.

Extended Functions Report

VMPRF PRF095 is a more general report that provides all cache information for the IBM 3990, as shown in Figure 27; PRF096 and PRF097 are related VMPRF reports.

```
PRF095  Run 09/09/1993 09:19:31        DASD_BY_ACTIVITY_EF                                                    Page   26
                                DASD Extended Function Activity Ordered by Activity
From 09/09/1993 13:25:38                                                                          GG243934
To   09/09/1993 13:27:38                                                                          CPU 9121     SN  20415
For    120 Secs 00:02:00              VM/ESA STORAGE MANAGEMENT MEASUREMENTS                          VM/ESA   21.01 SLU 9303
_____________________________________________________________________________________________________________________

<------------Device------------>        <---------Rate--------> <--------------------------Percent--------------------------->
                                                                    <---------Hits-------->
                  Stg         C D D                                                  DFW CFW                    DFW
Num- Volume        Ctlr Control A F U    Total  Read  Read Write                          Tot Tot Cache Norm  Seq  De-  By-
 ber Serial Type    ID   Unit    C W L ST   I/O NonSq   Seq    FW Read  Tot Read  Wrt  DFW CFW  Wrt Wrt   I/O Stge Stge Stge pass

0961 D14AA2 3390-2  0016 3990-3  A A A 00  29.7   6.1     0  23.6   87   79   76  100  100   0  100   0   100   19    0    0    0
0960 D14AA1 3390-2  0016 3990-3  A A A 00  29.0   6.7     0  22.3   85   74   69  100  100   0  100   0   100   25    0    0    0
0962 D14AA3 3390-2  0016 3990-3  A A A 00  18.3   7.6     0  10.7   77   70   62   95   95   0  100   0   100   29    0    0    0
0963 D14AA4 3390-2  0016 3990-3  A A A 00  17.7   5.7     0  12.0   80   80   76  100  100   0  100   0   100   20    0    0    0
0965 D14AA6 3390-2  0016 3990-3  A A A 00  15.6   4.2     0  11.4   85   77   74  100  100   0  100   0   100   20    0    0    0
0964 D14AA5 3390-2  0016 3990-3  A A A 00  15.5   4.8     0  11.3   76   74   66  100  100   0  100   0   100   26    0    0    0
```

Figure 27. VMPRF PRF095 Report

The percentage of read operations (variable READ) refers to the total number of cached I/O operations; it does not include the cache inhibit and bypasses:

$$\text{READ} = \frac{\text{Cache Reads}}{\text{Cache Reads} + \text{Cache Writes}} \times 100$$

Variable TOT represents a hit ratio that includes all hits; that is, read hits, DFW hits, and CFW hits, as shown below:

$$\text{TOT} = \frac{\text{Read Hits} + \text{DFW Hits} + \text{CFW Hits}}{\text{Cachable Reads} + \text{Cachable Writes}} \times 100$$

However, this variable does not represent the total cache hit ratio because VMPRF accounts for only the cachable I/O operations; cache inhibits and cache bypasses are not included.

Some software selectively disables I/O operations for caching using inhibit and bypass modes of operation, resulting in high cachable hit ratios but low hit ratios. Therefore, high disconnect times reported in conjunction with a high total cache hit ratio indicates that there are I/O operations inhibiting or bypassing the cache.

The hits are reported separately for read and write, but be aware that they are defined relative to the corresponding operations. The write hits are further classified into DASD fast write hits and cache fast write hits:

$$\text{HITS\_READ} = \frac{\text{Cached Read Hits}}{\text{Cached Reads}} \times 100$$

$$\text{HITS\_WRT} = \frac{\text{Cached Write Hits}}{\text{Cached Writes}} \times 100$$

$$\text{HITS\_DFW} = \frac{\text{DFW Hits}}{\text{DFW Writes}} \times 100$$

$$\text{HITS\_CFW} = \frac{\text{CFW Hits}}{\text{CFW Writes}} \times 100$$

Two other variables represent the percentage of DFW (DFW_TOT_WRT) and CFW (CWF_TOT_WRT) operations relative to the total number of writes. They are the probability of a write operation being a DFW or CFW operation, respectively:

$$DFW\_TOT\_WRT = \frac{DFW\ Writes}{Total\ Writes} \times 100$$

$$CFW\_TOT\_WRT = \frac{CFW\ Writes}{Total\ Writes} \times 100$$

DFW_TOT_WRT should normally be 100%, meaning that all write operations were eligible for DFW; some will be hits and some others will me misses. A number less than 100% indicates DFW inhibits or bypasses.

CFW_TOT_WRT should normally be zero, unless MVS is using the 3990-3. However, do not expect high values in this variable because CFW is used by few components and products.

A write hit is the union of two mutually exclusive events, each event composed of two independent events:

- A DFW write and a DFW write hit
- A CFW write and a CFW write hit

Therefore, the probability of a write hit is the sum of these two probabilities:

$$HITS\_WRT = DFW\_TOT\_WRT \times HITS\_DFW + CFW\_TOT\_WRT \times HITS\_CFW \quad (16)$$

CACHE_IO is the percentage of I/O operations eligible for caching and provides an indication of how many operations are inhibiting or bypassing the cache.

$$CACHE\_IO = \frac{Cached\ I/O}{Total\ I/O} \times 100$$

Extended Functions Example

As an example, assume that an application program executes 1000 I/O operations in EF mode, distributed in the following way:

- 1000 I/O operations
  - 800 reads
    - 700 hits
    - 100 misses
  - 200 writes
    - 150 DASD fast writes
      - 138 DFW hits
      - 12 DFW misses
    - 50 cache fast writes
      - 44 CFW hits
      - 6 CFW misses
- Average service time for a miss is 15 milliseconds
- Average service time for a hit is 2 milliseconds

Since there are no cache inhibits and bypasses, the relevant variables are (multiplication by 100 has been dropped):

$$\text{READ} = \frac{800}{1000} = 0.80$$

$$\text{WRITE} = \frac{200}{1000} = 0.20$$

$$\text{HITS\_READ} = \frac{700}{800} = 0.875$$

$$\text{HITS\_DFW} = \frac{138}{150} = 0.92$$

$$\text{HITS\_CFW} = \frac{44}{50} = 0.88$$

$$\text{HITS\_WRT} = \frac{138 + 44}{200} = 0.91$$

$$\text{TOT} = \frac{700 + 138 + 44}{1000} = 0.882$$

$$\text{DFW\_TOT\_WRT} = \frac{150}{200} = 0.75$$

$$\text{CFW\_TOT\_WRT} = \frac{50}{200} = 0.25$$

$$\text{CACHE\_IO} = \frac{1000}{1000} = 1$$

$$T_S = 15 - \left[0.882 \times (15 - 2)\right] = 3.5$$

The write hit ratio can also be calculated using equation (16) on page 87:

$$\text{HITS\_WRT} = (0.75 \times 0.92) + (0.25 \times 0.88) = 0.91$$

The extended cache effectiveness, calculated with equation (15), is equal to the variable TOT because there are no cache inhibits nor bypasses:

$$\text{TOT} = \text{READ} \times \text{HITS\_READ} + \text{WRITE} \times \text{HITS\_WRT}$$
$$= (0.80 \times 0.875) + (0.20 \times 0.91) = 0.882$$

Nonsynchronous I/O Operations

This section reviews the basic principles of nonsynchronous I/O operations and discusses how the DASD I/O components are affected when they are attached to nonsynchronous storage controls.

Basic Concepts

One of the serious handicaps of synchronous I/O operations is that all channel and storage control activity related to one channel command must complete during the inter-record gap between two adjacent fields in the track. Among other shortcomings, this characteristic limits the operating distance for storage controls to 122 meters (400 feet).

The lack of synchronization between the channel, storage control, and device is called nonsynchronous operation. To get the best performance, a nonsynchronous storage subsystem must know the nature and scope of an I/O operation before it begins transferring data between the storage control and either the channel or the device.

To provide such information, the CKD command set has been extended by the addition of new commands to form the extended-CKD (ECKD) command set. This extension is not a radical deviation from CKD; the data format on the track has not changed and can be read with both CKD and ECKD channel programs.

Nonsynchronous I/O operations may take advantage of the ECKD command set by allowing the storage control to anticipate data transfer commands and start the transfer to a buffer (in the case of a read) if the desired record arrives under the actuator before the actual read command arrives, or by allowing a transfer from the channel into that buffer (in the case of a write) when the write command arrives before the record to be written to arrives under the actuator.

Nonsynchronous mode requires a buffer in the storage control to eliminate the synchronization requirements. The IBM 3990 models (except Model 1) have a 64-KB buffer for each storage path, called the ADT (Automatic Data Transfer) buffer, which is independent of the cache buffer. The IBM 9340 noncached models have a 1-MB buffer for each storage cluster; on cached models, the cache buffer is used for both cache and nonsynchronous functions.

Advantages

Nonsynchronous storage subsystems have the potential to implement the following advantages:

- The minimum size of the inter-record gap, and the maximum length of channel cables, are no longer determined by signal propagation delays and command processing in the host channel subsystem. Eliminating this constraint removes a factor that limits channel cable lengths and allows devices to have smaller inter-record gaps.
- The channel and the device need not transfer data at the same time, which permits independent processing elements in the path between the device and host.
- Channel busy may be reduced because data can be transferred at channel speeds rather than at device speeds.
- Reconnection after a Set Sector operation requires only that the storage control be available when the device reaches the target sector; RPS misses due to channel busy can be eliminated.

Nonsynchronous Storage Controls

The IBM 9340 and the IBM 3990 can operate in nonsynchronous mode as follows:

- The IBM 9340 always operates nonsynchronously.
- The IBM 3990 always operates nonsynchronously on ESCON channels.
- The 3990-3 and 3990-6 normally operates synchronously on parallel channels.
- The IBM 3990-3 operates nonsynchronously on parallel channels if:
  - The vital product data (VPD) bit is set (done by customer engineer).
  - All ECKD operations or CKD read operation experience channel busy when trying to reconnect after a cache miss.
- The IBM 3990-3 and 3990-6 write operations (CKD or ECKD) operate nonsynchronously on parallel channels if the I/O operation is within the domain of a concurrent copy session.

The type of the channel program is independent of the mode of operation of the storage control. A nonsynchronous storage control executes both CKD and ECKD commands nonsynchronously; a synchronous storage control executes both CKD and ECKD programs synchronously.

CKD Channel Programs

When a CKD program is issued to a nonsynchronous subsystem, the storage control does not know (as it does in case of ECKD programs) whether a read or write will be requested until the actual data transfer command is presented.

Since most I/O requests are read operations, the nonsynchronous storage controls assume that a CKD program intends to read, until told otherwise by the channel program. This may cause unnecessary subsystem overhead, such as:

- Reading of records not requested
- Nonproductive storage control busy
- Nonproductive physical device busy

Read Operations: In general, the nonsynchronous storage controls handle read channel programs without significant loss of performance. Some subsystem overhead is unavoidable because the storage control does not know how much data should be transferred in advance into the buffer. Geometry-dependent channel programs and programs that use special techniques, such as switching heads during the chain of commands, suffer more loss of performance.

Write Operations: For write operations, the storage control must search for a specific count (orientation) before it can start writing to the device. Nonsynchronous storage controls cannot process the ending status of the search command and get the new command for the device within the gap time, even if there are filler records in the track. Therefore, one full revolution is lost for each write command, even if there are several write commands chained in the same channel program.

VM/ESA Implementation: VM/ESA checks whether the storage control is ECKD-capable and builds ECKD channel programs for paging, spooling, and DIAGNOSE I/O according to the storage control capability. CP even tries to convert CKD channel programs (DIAGNOSE I/O) to ECKD channel programs, if the channel program is not complex.

CKD Program Example

Figure 28 is an example of a typical CKD channel program; this example reads the label of a DASD volume.

The Set File Mask command disallows any other seeks inside the channel program. The TIC commands forces the Search command to execute until the search argument matches the actual record id in the track.

```
CKD     CCW   X'07',SEEK,X'40',6     * Seek
        CCW   X'1F',MASK,X'40',1     * Set file mask
        CCW   X'23',SECT,X'40',1     * Set sector
        CCW   X'31',CYLA,X'40',5     * Search
        CCW   X'08',*-8,0,0          * TIC
        CCW   X'06',IOBUF,X'20',80   * Read
SEEK     DC    X'0000'               * Seek address
CYLA     DC    X'0000'               * Cylinder address
TRKA     DC    X'0000'               * Track address
RECA     DC    X'03'                 * Record number
SECT     DC    X'00'                 * Sector address
SFM      DC    B'01001000'            * Set file mask options
IOBUF    DS    CL80                  * I/O buffer
```

Figure 28. CKD Channel Program Example

ECKD Program Example

Figure 29 is an example of a typical ECKD channel program; this example also reads the label of a DASD volume.

The Define Extent specifies, among other things, the number of bytes to read and the range of tracks over which the channel program is allowed to operate. The Locate Record specifies the type and scope of the data transfer operation. In the example, the storage control knows that it should read data (orientation and operation field) before the actual read command is executed.

```
ECKD     CCW   X'63',DEFE,X'40',16    * Define Extent
         CCW   X'47',LOCR,X'40',16    * Locate Record
         CCW   X'06',IOBUF,X'20',80   * Read
DEFE      DC    X'48'                * Mask
         DC    X'C0'                * Global attributes
         DC    X'0050'               * Block size
         DC    X'0000'               * Cache fast write id
         DC    X'0000'               * Reserved
         DC    X'00000000'            * Beginning of extent
         DC    X'00000000'            * End of extent
LOCR      DC    X'06'                * Orientation and operation
         DC    X 00'                * Auxiliary byte
         DC    X 00'                * Reserved
         DC    X 01'                * Locate domain
         DC    X'00000000'            * Seek address (CCHH)
         DC    X'0000000003'          * Search address (CCHHR)
         DC    X'00'                * Sector number
         DC    X'0000'               * Transfer length factor
IOBUF     DS    CL80                  * I/O buffer
```

Figure 29. ECKD Channel Program Example

Filler Records

A heritage from synchronous subsystems is the insertion of small filler records between the real data records, to effectively increase the size of the inter-record gaps.

The use of filler records allows the software to chain several commands and process several data records without requiring additional revolutions. The required size of the filler record varies with the device type, and might vary as a function of storage control type.

The filler record size depends upon an assumption of synchronous operations and may require modification whenever there is a change in device or storage control operation characteristics.

In spite of this, the ICKDSF program and the CPFORMAT command used to insert filler records when formatting a 3380 (or 3390 in 3380-compatibility mode), to enable a better efficiency in the CP paging routines. Figure 30 is an example of a 3380 track formatted with filler records, where the filler records are shaded.

Figure 30. Filler Records

CKD channel programs always search before reading or writing a record, and therefore, filler records are skipped by the search command and are never read when using CKD channel programs.

ECKD channel programs may have several read or write commands within one locate record domain, and therefore, would read the filler records as data, causing unpredictable results.

To avoid this problem, VM/ESA uses CKD channel programs if the 3380 is formatted with filler records, which may cause one lost revolution for each write on those volumes, when they are connected to nonsynchronous storage controls. This would cause disastrous performance when these volumes are used for active CP system areas, particularly for paging areas.

Therefore, when migrating to VM/ESA, make sure all 3880 volumes (and 3390 volumes in 3380-compatibility mode) connected to nonsynchronous storage controls are formatted without filler records by specifying the NOFILLER parameter in the ICKDSF CPVOLUME command.

When you format a CP-owned volume without filler records, you must format cylinder zero and all CP areas, such as page and spooling; formatting only cylinder zero would provide a false indication to CP that the volume does not contain filler records and would cause unpredictable results.

You do not have to worry about CMS minidisks and SFS file pools if you used the standard CMS commands and procedures to format them, because filler records are never written in such cases.

Path Protocol

The path protocol time is longer with ESCON channels and you must also account for signal propagation delays due to longer distances the devices may be from the channel. This should add 1 to 5 milliseconds, depending on the distance the storage control is from the host.

RPS Miss Avoidance

RPS Miss Avoidance (RMA), also known as nonsynchronous operations enhancement, is an improvement to the IBM 3990 Models 3 and 6 when operating in nonsynchronous mode. RMA addresses the situations in which the device can connect to the storage control, but the storage control cannot connect to the channel.

Each cluster of the IBM 9341 or 9343 storage controls (noncached) contains a 1-MB buffer that is used to provide the same function; in case of cached 9343 storage controls, the cache is used to implement this function.

Licensed internal code is available for the IBM 9340 subsystem, which uses this buffer (or the cache) in a manner similar to the RMA on the IBM 3990 to provide enhanced performance.

In nonsynchronous mode, cache hits are processed exactly as they are in synchronous mode; the data is in cache, and as soon as a channel reconnects, data is transferred at channel speed.

In case of a miss, if the path from the device to the channel is available during the reconnection window, that path is normally used to transfer data. This is the same process that occurs with other storage controls that do not have the RMA capability.

If the channel does not reconnect to the device within the reconnect window, the data is flushed out of the ADT but is left in the cache and is converted into a cache hit. When a channel reconnects, the data is transferred at channel speed.

Therefore, the reconnection after the Set Sector operation requires only that the storage control is available during the reconnect window; RPS misses due to channel busy conditions are eliminated.

The performance of an I/O handled by RMA is not as good as it would be if the storage control were able to reconnect immediately. However, it is much better than it would be if a conventional RPS miss had occurred.

Data Transfer

Data is always transferred at channel speeds for the IBM 9340 (cached and noncached) and for the cached IBM 3990 when data is in cache. When data is in the 3990 ADT buffer, data is transferred at device speed.

In case of the IBM 9340 storage controls, the device is connected to the storage control but not to the channel during the entire data transfer time. The channel may connect to the storage control after the device starts transferring data to the buffer (read operation) or disconnects from the storage control before the device finishes transferring data from the buffer (write operation).

FBA Considerations

The discussion about I/O components and how to estimate them also applies for FBA DASD. However, the channel subsystem provides almost no statistics and the performance tools report zero or inaccurate values for utilization, service time, queue time, and response time for FBA DASD. MLOAD (paging and spooling) statistics are correct, though, because they are computed by CP.

Path Protocol

The path protocol (attachment overhead) for FBA DASD is about 10 milliseconds.

Automatic Position Sensing

FBA DASD are designed so that the device is not connected to the path while waiting for the requested record to come under the actuator.

Since there is no software assistance required by FBA DASD to automatically disconnect from the path while waiting on the requested record, there is no need to have any software support for the facility. In fact, there is no way to write software that can cause an FBA device not to disconnect.

Therefore, positioning is called Automatic Position Sensing (APS) instead of Rotational Position Sensing, but it works in the same way; however the basic data unit for sensing purposes is an FBA block rather than a sector.

Therefore you can estimate the RPS-miss time for FBA DASD in the same way you do for CKD DASD; however, FBA DASD make use of nonsynchronous I/O operations and some considerations discussed in “Nonsynchronous I/O Operations” on page 89 also applies here.

Search

There is no search command as we know it for CKD DASD. Nevertheless, after connecting for data transfer, the first block must be positioned under the actuator (FBA DASD use Locate Record CCW).

Since the basic data unit for sensing purposes is an FBA block, you can estimate the search as the amount of time that the device takes to skip one FBA block.

For a 9336, with 58 blocks/track and 16.6 milliseconds/revolution, the average time for search is:

$$T_{search} = \frac{2 \cdot 8.3}{58} = 0.29$$

Data Transfer

A CKD device can sustain its data rate for the transfer of a complete block of user data. On FBA DASD, user data is stored in a number of 512-byte blocks, and therefore, contains gaps between each 512 bytes.

During data transfer on FBA DASD, the user data is not sent in a continuous stream as it would be for CKD DASD. Therefore, the FBA instantaneous data rate cannot be sustained for multiple blocks of user data. For example, the sustained data rate for 4-KB blocks on a 9335 is approximately 2.2 MB/s compared to the 3 MB/s instantaneous data rate, as shown in Table 3 on page 43.

Chapter 6. Data-in-Memory Facilities

Data-in-memory is a general term related to those facilities that put data residing in DASD in storage to get a better performance. In other words, they trade general I/O instructions for paging operations. This chapter describes the main data-in-memory facilities available in VM/ESA.

CP Data Caching

The caching of data for CP can take advantage of expanded storage but does not require the availability of expanded storage. Benefits can be achieved by NSSs, such as CMS, that have some nonshared pages saved. Assume that CMS has been defined and saved as follows:

1. DEFSYS CMS *aa-bb* EW *cc-dd* EW *eee-fff* SR MINSIZE=256
2. I 190 CLEAR PARM SAVESYS CMS

Images of the page ranges *aa-bb*, *cc-dd*, and *eee-fff* are saved in the SPOL area on a CP owned volume. When the first user loads CMS into storage (IPL CMS), CP performs the normal IPL process and also copies the nonshared pages to an special address space in order to save I/O operations in subsequent IPLs.

The sequence of operations is as follows:

1. USER1 issues IPL CMS (this is the first user on the system to load CMS by name).
2. Pages *aa* through *bb* saved as EW (exclusive write) are copied into a CP address space and into USER1's virtual pages *aa* through *bb*. The CP copy is now maintained by the paging subsystem.
3. The same action occurs for pages *cc* through *dd*. The IPL of CMS for USER1 has not avoided using I/O to the spool area where the CMS saved segment resides.
4. Pages *eee* through *fff*, saved as SR (share read), are connected to the segment table that describes USER1's virtual storage.
5. USER2 now issues IPL CMS.
6. Page ranges *aa* through *bb* and *cc* through *dd* are copied from the CP image held in CP's address space into USER2's virtual storage. Since both ranges were saved as EW (Exclusive Write), they cannot be shared with other users; every user must have a copy.
7. The range of pages from *eee* through *fff* can be shared among several users; therefore, CP updates USER2's segment tables to point to the same page tables that describe the shared pages. Neither user is allowed to modify storage in this area.

Only USER1 caused I/O activity to IPL CMS. All other users obtained their copy of CMS from CP's address space, thus eliminating all I/O and the associated delay and wait time, provided the frames were resident in either central or expanded storage.

The CP system directory is cached in the system virtual address space, significantly reducing the I/O activity to the DRCT cylinders, as discussed in “System Virtual Address Space” on page 29.

Minidisk Caching in VM/ESA Release 2.1

When a user reads data from a minidisk, a copy of the data is stored in the minidisk cache (MDC). Any subsequent reads from this (or any other) user bypass the I/O operations by retrieving the data from the cache.

The following is a typical scenario that shows how the MDC works and how virtual machines can benefit from it:

1. USER1 reads a CMS file; if the corresponding blocks are not in the MDC, the CP minidisk cache routine:
   a. Reads the blocks into central storage
   b. Copies them into expanded storage
   c. Marks them as resident in the MDC
2. USER1 (or any other user) reads the same blocks again; if the data blocks are still in the MDC, they are fetched from there, thus avoiding the corresponding I/O operations.
3. USER1 modifies the data; the data blocks are written to the minidisk and updated into the MDC (store-through technique).

Performance benefits are achieved only by read requests because all writes to minidisks are always done first, before the MDC copy of the data is updated.

However, substantial performance gains can still be achieved for read/write minidisks; performance studies have shown that a typical user's 191 minidisk has a high read-to-write ratio. Users normally read 50% of the files on R/W minidisks more than once, with most of the reads being from the same files.

Unless controlled by the CP RETAIN MDC command, the arbiter dynamically computes the optimum size of the minidisk cache (MDC). As mentioned in “Expanded Storage” on page 7, if the paging rate is low, you may want to configure part of the total storage in an ES/9121 processor, for example, as expanded storage, and dedicate all that expanded storage to MDC with the CP RETAIN command.

RTM Variables

RTM reports the read-to-write ratio in the MDRM variable. The percentage of read operations is not reported by RTM, but can be calculated through the following equation:

$$PCT\_READ = \frac{MDRM}{1 + MDRM} \times 100$$

The MDC hit ratio is reported as variable MDRH, calculated by:

$$MDHR = \frac{\text{Read Hits}}{\text{Read Hits} + \text{Read Misses}}$$

and represents the probability of a read cache hit. The hit counter is incremented by the number of blocks in the data request only when all blocks are found in the MDC. The number of blocks per read request is an adjustable value specified in the DMSNGP ASSEMBLE file (default is 8KB).

Consequently, the cache miss counter is incremented by the number of blocks within a data request if any of the blocks in that request are not found in MDC.

Figure 31 shows that good and bad cache hit ratios are related to high and low read-to-write ratios, respectively.

```
TOD VM/ESA CPU9121 SERIAL 420415  253M DATE 09/12/93 START 13:25:59 END 14:15:25
H:M   XSTON  XSTAV LOTH XAGE MAGE BKRD BKMG BKST MDAU MDLI MDNE MDAC MDHR MDRM
    ......................
1330   256M  2228K  480   86  355    6   18   22   44  600  14K  16K  .95   17
1330   256M  1780K  580   90  100    6   19   28   86  600  15K  16K  .96   22
1331   256M  1532K  680   88  162    7   19   24   45  600  16K  16K  .95   14
1331   256M  3548K  680   83  136    5   19   20   98  600  16K  16K  .96   14
1332   256M   796K  720   86  102    7   19   26   52  600  15K  15K  .94   12
1332   256M  2564K  780   81  156    7   19   21  119  600  13K  13K  .94 7.50
1333   256M  2120K  700   81   97    6   16   25   45  600  13K  13K  .92 9.42
1333   256M  3712K  800   79  108    7   19   24  109  600  12K  12K  .95 8.45
1334   256M   108K  760   79  167    7   19   25   57  600  12K  12K  .95   11
1334   256M  3224K  720   75  122    6   16   24  126  600 9011 9011  .94 8.59
1335   256M  2856K  800   70  104    6   19   27   68  600 8299 8299  .94 9.80
    ......................
1412   256M  2660K  580  158  552    3   17   17   97  600 1688 1688  .54 2.39
1412   256M  2624K  540  164  488    4   16    0   20  600 1688 1688  .51 3.49
1413   256M  2764K  560  160  619    4   17   32   99  600 3708 4096  .57 3.09
1413   256M  3588K  600  157  598    4   16   24   15  600 4096 4096  .78 2.25
1414   256M  4124K  580  168  615    3   17    0   80  600 4096 4096  .67 2.53
1414   256M  2684K  560  166  565    4   14    0   18  600 4096 4096  .73 1.99
1415   256M  2952K  520  176  396    4   12    0   72  600 4096 4096  .79 2.18
```

Figure 31. RTM XTLOG Screen

XAGE (average age of expanded storage block) and MAGE (maximum age of expanded storage block) are variables that measure that average age of expanded storage blocks, and should be monitored to determine whether the expanded storage size is appropriate for the workload. MAGE should be greater than XAGE so the pages of active users can stay in expanded storage long enough to be re-used when users request them again.

BKRD and BKMG are variables that measure the effectiveness of block paging. Low values indicate that there is not enough paging space allocated to form large blocks. BKRD, BKMG, and BKST are further explained below:

BKRD Average number of pages read in a block as a result of a page fault; BKRD is about half the actual blocking factor, as discussed in “Block Paging” on page 34.

BKMG Average number of pages migrated out of XSTORE. Pages are read into a central storage buffer and written to DASD using block paging, as discussed in “Page Migration” on page 39.

BKST Average number of pages moved out of central storage by the available-list replenishment routine. Pages are written one by one to expanded storage as explained before.

MDAU Number of users that did at least one insert into the cache during the last interval. If MDAU is much less than the number of active users it may be an indication that minidisk caching has been disabled.

LOTH Low threshold, as discussed in “Page Migration” on page 39. Migration is triggered if an expanded storage block has to be allocated and there are only this many blocks available

VMPRF Variables

Figure 32 shows the corresponding VMPRF PRF020 report for the MDC activities; the main variables are described below.

```
PRF020  Run 09/09/1993 08:59:46        MINIDISK_CACHE_BY_TIME                                                    Page   14
                                Minidisk Cache Activity by Time
From 09/09/1993 07:41:02                                                                          GG243934
To   09/09/1993 08:51:02                                                                          CPU 9121     SN  20415
For   4200 Secs 01:10:00              VM/ESA STORAGE MANAGEMENT MEASUREMENTS                            VM/ESA   21.01 SLU 9303
_____________________________________________________________________________________________________________________

          <------XSTORE Blocks------> <-----MDC Requests----> <--------------MDC Blocks-----------> <----Rate---><Invalidation>
                <--------MDC-------->      <----Percent----> <------Read------> <---Rate--->                 <----Rate--->


                                          Par-                                          Est.   Fair     In
From  To                        Fair  Read  Full  tial                  Pct   MDC    MDC Block  Share Transt    Re-
Time  Time     CP    Max Actual  Share  Rate   Hit   Hit  Miss   Rate   Pct   Hit  PGIN  PGOUT  Life Exceed Insert quests Blocks

    ..............................
08:39 08:40  65536   4096   4096    600   102  71.3   0.4  28.3    191  77.2  67.1   129    119    34      0   0.18      0      0
08:40 08:41  65536   3566   3566    600    98  69.3   0.2  30.5    189  73.4  63.9   121    137    26      0      0      0      0
08:41 08:42  65536   3566   3566    600   115  69.6   0.3  30.1    216  73.7  66.9   145    149    24      0      0      0      1
08:42 08:43  65536   3566   3566    600   119  70.4   0.3  29.3    224  72.5  64.6   146    164    22      0   2.33      0      0
08:43 08:44  65536   3371   3371    600    98  66.7   0.3  33.0    193  74.0  68.9   133    127    26      0      0      0      0
08:44 08:45  65536   4096   4096    600   110  69.6   0.2  30.2    213  72.9  64.2   137    155    26      0      0      0      0
08:45 08:46  65536   4096   4096    600    96  79.9   0.3  19.8    185  77.5  79.8   148     91    45      0      0      0      0
08:46 08:47  65536   3720   3720    600    86  79.6   0.2  20.2    167  79.1  80.7   135     77    49      0      0      0      0
08:47 08:48  65536   3526   3526    600    86  75.5   0.3  24.2    158  79.9  74.6   118     80    44      0      0      0      0
08:48 08:49  65536   3526   3526    600    68  75.0   0.1  24.8    132  76.2  68.9    91     82    43      0      0      0      0
08:49 08:50  65536   3526   3526    600    61  75.8   0.2  24.0    122  70.2  77.8    95     79    44      0      0      0      0
08:50 08:51  65536   3526   3526    600    86  73.9   0.2  25.9    157  68.7  69.8   110    119    30      0      0      0      0
```

Figure 32. VMPRF PRF020 Report

- MDC Requests
  - READ_RATE

    Rate of requests per second to read one or more blocks of data from MDC. The number of requests is incremented by one for every read operation regardless of the number of blocks in the request; that is, if the read operation transfers either one or 10 blocks, the READ_RATE is still incremented by one. Compare this definition with variable RATE, described below.

  - FULL_HIT, PARTIAL_HIT, and MISS

    Similarly to the previous variable, these three variables refer to read operations, not to MDC blocks; therefore, they should not be compared with RTM variables.

    A full hit occurs when all blocks requested in the read operation are found in MDC. A partial hit occurs when at least one block is not in MDC. A miss occurs when none of the blocks are found in MDC. The sum of these three variables always adds up to 100.

  The *VMPRF User's Guide and Reference* uses the name “Read Requests” when defining the READ_RATE and “Cache Reads” when defining FULL_HIT, PARTIAL_HIT, and MISS; however, they are the same variable.

- MDC Blocks
  - RATE

    The number of blocks read per second from the MDC, regardless of how many operations were executed to read them. When you compare this RATE with READ_RATE in Figure 32, you can see that, most of the time, two blocks are read on one I/O request.

- PCT

  Percentage of MDC blocks transfers that are read operations; in other words, PCT represents the probability of a read operation. VMPRF does not report the read-to-write ratio, but it can be calculated by:

$$RATIO = \frac{PCT}{100 - PCT}$$

- PCT_HIT

  The equivalent of the MDHR variable reported by RTM; the same comment about partial hits counted as cache misses applies here. The cache-hit counter is incremented by the number of blocks when all blocks are found in MDC. Therefore, this variable also represents the probability of a cache hit.

  Similarly, the cache-miss counter is incremented by the number of blocks when at least one block is not found in MDC. This variable is not reported by VMPRF, but can be calculated as the PCT_HIT complement to 100.

As an example, let us calculate these variables for the following assumptions:

- 160 reads and 40 writes during 10 seconds such that:
  - 80 reads specify two blocks
  - 80 reads specify four blocks
- 148 reads found all blocks in cache (hits):
  - 80 reads specify two blocks
  - 68 reads specify four blocks
- 4 reads found only one record in cache
- 8 reads found no blocks in cache

$$READ\_RATE = \frac{160}{10} = 16 \quad \text{reads/seconds}$$

$$FULL\_HIT = \frac{148}{160} \times 100 = 92.5\%$$

$$PARTIAL\_HIT = \frac{4}{160} \times 100 = 2.5\%$$

$$MISS = \frac{8}{160} \times 100 = 5\%$$

$$RATE = \frac{80 \times 2 + 80 \times 4}{10} = 48 \quad \text{blocks read/second}$$

$$READ\_PCT = \frac{160}{200} \times 100 = 80\% \qquad RATIO = \frac{160}{40} = \frac{80}{100 - 80} = 4$$

$$PCT\_HIT = \frac{80 \times 2 + 68 \times 4}{80 \times 2 + 80 \times 4} 100 = 90\%$$

$$PCT\_MISS = 100 - 90 = 10\%$$

VMPRF also reports the invalidation rates for requests and blocks, thus providing a good indication of non-MDC I/O to eligible minidisks. Any value other than zero should be investigated and the minidisk should be considered a potential candidate for removal from MDC eligibility.

Selecting Minidisks for MDC Eligibility

Once you establish that the overall read-to-write ratio is low, the next step is to determine which minidisks are contributing with most write operations and remove their MDC eligibility.

Neither RTM nor VMPRF provides a report that identifies those minidisks. The VMPRF PRF012 report, however, provides that kind of information at the volume level, as shown in Figure 33.

```
PRF012  Run 09/09/1993 08:59:46        DASD_BY_ACTIVITY                                                    Page   26
                                DASD Activity Ordered by Activity
From 09/09/1993 07:41:02                                                                    GG243934
To   09/09/1993 07:46:02                                                                    CPU 9121     SN  20415
For    300 Secs 00:05:00              VM/ESA STORAGE MANAGEMENT MEASUREMENTS                    VM/ESA   21.01 SLU 9303
_____________________________________________________________________________________________________________________

<-----Device------>                          <---------SSCH+RSCH-------->      <----------Time------------> <-SSCHs in>
                                                                                                          <--Queue-->

                                  Mini-    On-                    Plus
Num- Volume        Control         disk   line                 Plus  Avoid   Pct
 ber Serial Type    Unit    Owner  Links   Secs   Count  Rate Avoided   Rate  Busy  Pend  Disc  Conn  Serv  Resp  Mean   Max Err

096C SET1C0 3390-2  3990-3           15    300     182   0.6   2611    8.7   0.8   0.1   9.8   3.1  12.9  12.9     0     0   0
096E SET2C0 3390-2  3990-3           14    300     186   0.6   2473    8.2   0.8   0.1  10.0   3.1  13.2  13.2     0     0   0
0DAC SET3C0 3390-3  3990-3           13    300     181   0.6   2126    7.1   0.8   0.1   9.8   3.2  13.0  13.0     0     0   0
0DAE SET4C0 3390-3  3990-3           13    300     182   0.6   1983    6.6   0.8   0.0   9.9   3.2  13.1  13.1     0     0   0
0A22 3934V1 3380-D  3880-23          26    300      83   0.3    733    2.4   0.4   2.9   6.7   3.6  13.3  13.3     0     0   0
096F SET2C1 3390-2  3990-3           12    300     634   2.1    731    2.4   2.3   0.0   8.6   2.2  10.8  10.8     0     0   0
0A27 3934PK 3380-D  3880-23          58    300      94   0.3    610    2.0   0.3   3.1   3.6   3.5  10.2  10.2     0     0   0
0A25 3934V4 3380-D  3880-23           5    300     570   1.9    570    1.9   0.9   0.5   1.9   2.5   4.9   4.9     0     0   0
096D SET1C1 3390-2  3990-3           12    300      26   0.1    132    0.4   0.1   0.0   6.7   4.8  11.6  11.6     0     0   0
0A20 3934RS 3380-D  3880-23          58    300      92   0.3    114    0.4   0.5   1.7  10.2   4.8  16.7  16.7     0     0   0
0A26 3934V5 3380-D  3880-23           6    300      99   0.3    109    0.4   0.8   2.3  15.9   4.7  22.8  22.8     0     0   0
0DAF SET4C1 3390-3  3990-3           11    300      40   0.1    108    0.4   0.2   0.1   8.2   4.2  12.5  12.5     0     0   0
0DAD SET3C1 3390-3  3990-3           11    300      22   0.1    101    0.3   0.1   0.0   7.8   5.5  13.3  13.3     0     0   0
0961 D14AA2 3390-2  3990-3            3    300      77   0.3     77    0.3   0.2   0.1   4.4   2.7   7.2   7.2     0     0   0
0962 D14AA3 3390-2  3990-3            1    300      59   0.2     59    0.2   0.2   0.1   5.1   2.6   7.7   7.7     0     0   0
0967 D14AA8 3390-2  3990-3            1    300      59   0.2     59    0.2   0.1   0.0   4.7   2.3   7.0   7.0     0     0   0
```

Figure 33. VMPRF PRF012 Report

All variables related to this discussion are under the general heading *SSCH+RSCH*.

Variables COUNT and RATE indicate the total number and rate of actual I/O operations executed to transfer data from all minidisk on the volume.

Variables PLUS_AVOIDED and PLUS_AVOID_RATE indicate the total number and rate of I/O operations including those I/O operations avoided because of MDC. Therefore, those volumes with COUNT similar to PLUS_AVOIDED (or RATE similar to PLUS_AVOID_RATE) are the volumes that contain minidisks with low read-to-write ratios.

You can estimate the MDC effectiveness for a volume or for the whole system as:

$$\text{MDC effectiveness} = \frac{\text{PLUS\_AVOIDED} - \text{COUNT}}{\text{PLUS\_AVOIDED}} \times 100\%$$

Higher values are better; volumes with low values (less than 50%) contain minidisks with low read-to-write ratios and should be investigated. However, VMPRF does not provide you with information on how to identify which minidisks in those volumes are responsible for the low effectiveness; the seek reports can help you identify only which minidisks are more active than others.

Conditions for MDC Eligibility

Some people are cautious about using MDC because they are concerned that it may impact their paging. Instead of being concerned about the page rate alone, you should be concerned with reducing the total amount of avoidable I/O operations. Your should determine the point at which the sum of minidisk real I/O operations, DASD paging, and DASD page migrations is the lowest. Usually, this can be achieved by the correct partitioning of expanded storage for paging and MDC, and a constant monitoring with RTM/ESA and VMPRF.

The following conditions must be met for a minidisk to be eligible for MDC:

- Expanded storage is available for minidisk caching.
- MDC has not been disabled for the minidisk by the MINIOPT NOMDC directory statement.
- The minidisk is not a full-pack minidisk.
- The minidisk is formatted with 4-KB blocks.
- FBA minidisks must be page aligned; that is, the starting block number and the number of blocks must be evenly divided by eight.

  In fact, other functions also require FBA disks in page boundaries, such as mapped minidisks, virtual disks, and system areas. However, virtual disks are rounded up to the next multiple of eight; the start of a CP system area is rounded up and the end is rounded down.
- Standard Diagnose I/O or IUCV *BLOCKIO interfaces are used to perform I/O to the minidisk.
- The real device that has the minidisk allocated on it has not been specified as SHARED. Note that a device can be defines as shared in different ways:
  - RDEVICE macro in HCPRIO
  - RDEVICE statement in system configuration file
  - CP SET RDEVICE command
  - CP SET SHARED command

  The CP QUERY DASD command can be used to verify whether a DASD volume is defined as shared.

Some conditions cause parts of the MDC to be flushed; for example:

- Formatting a minidisk
- Detaching all links to a minidisk
- Writing to a minidisk using non-MDC supported I/O
- Increase of the paging area by the arbiter

Performing non-MDC eligible I/O to a full-pack minidisk overlaying other minidisk cause the MDC for the whole volume to be flushed. This may happen, for example, when DIRMAINT re-writes the CP directory.

Note that it is not the existence of an overlay that causes the flushing of the MDC for the other minidisks; CP flushes the MDC only when a non-MDC write operation occurs, because CP does not know what blocks might have been changed as a result of the non-MDC I/O operation.

Disabling MDC

Sometimes you may want to disable MDC for some minidisks to improve performance or to avoid a potential data integrity problem.

You can disable MDC, one minidisk at a time, by coding the MINIOPT NOMDC directory statement after the MDISK directory statement.

You can also use the SHARED option, as discussed before, to disable MDC for all minidisks residing in a specific volume.

Performance

As shown in Figure 31 on page 97, some minidisks with low read-to-write ratios (less than four) are not good candidates for MDC. They make insertions into the MDC but seldom read them, causing steals of MDC slots containing blocks of other minidisks with high read-to-write ratios and diminishing their hit ratio. Therefore, in order to achieve a better overall performance, these minidisks should not be allowed to use MDC.

Data Integrity

When sharing minidisks with different VM images, you must be very careful not to destroy data or access incomplete data. If you have expanded storage available for MDC, you must make sure MDC is completely or partially disabled, as discussed below.

Several R/W Links: Most of the time, CMS minidisks must not be accessed in R/W mode by more than one virtual machine at the same time (minidisks processed with *BLOCKIO system service are potential exceptions). Whether the virtual machines are under one or more VM images does not make any difference; one virtual machine may access the minidisk in R/W mode and all others should access the minidisk in R/O mode.

However, if you have set up the proper controls (if you are using Cross System Extensions, for example), you may have CMS minidisks in R/W mode in two or more systems, provided that just one virtual machine has R/W access (ACCESS command) to them at a time.

In this case, you must disable MDC for all systems that have a R/W link to the minidisks; otherwise, you may lose your data.

One R/W Link and Several R/O Links: When the virtual machines are under the same VM image, the R/O virtual machines may access incomplete data because the CMS file directory is kept in the virtual storage of each virtual machine. However, a new ACCESS command corrects the problem. MDC may be enabled because CP is able to keep track of the updates in the minidisk and in the MDC.

When the virtual machines are in different VM images, the VM systems on the R/O sides do not know about the changes, and the virtual machines may access incomplete data, as before. However, if MDC is enabled, a new ACCESS command does not correct the problem because data is read from the out-of-date MDC.

Therefore, in order to be able to re-access the new data, only the virtual machine with R/W access should have MDC enabled; all the other R/O virtual machines in other images should disable MDC.

Enhanced Minidisk Caching in VM/ESA Release 2.2

Major enhancements have been made to minidisk caching in VM/ESA Release 2.2. Most of the restrictions that applied in VM/ESA Release 2.1 have been removed, extending MDC benefits to any guest. Additional commands have also been introduced, providing better controls over the caching process.

Conditions for MDC Eligibility

Minidisks are eligible for MDC provided they are defined in one of the following devices, and not dedicated to a guest:

- 3380
- 3390
- 9345
- RAMAC
- FBA

Minidisks defined in FBA devices must still be page aligned; that is, the starting block number and the number of blocks must be evenly divided by eight.

While it is allowed to cache minidisks on shared devices, there is no support to automatically update or purge tracks that have been updated by other systems. However, some applications can use the new commands to flush the MDC and successfully share infrequently updated minidisks.

Because it is not possible to ensure that this environment always works, IBM does not recommend nor support caching minidisks residing on shared devices. See “Disabling MDC” on page 102 for a discussion of some problems related to MDC on shared devices.

Expanded storage is not required for MDC; the cache can reside in central storage, expanded storage, or a combination of both. When cache is in central storage, performance is improved relatively to MDC in VM/ESA Release 2.1.

Minidisks can be in any format, and applications that use SSCH, SIO, and SIOF, in addition to DIAGNOSE I/O and *BLOCKIO, can benefit from MDC. Therefore, minidisks may be from a CMS user, VSE user, MVS user, or any other guest supported by VM/ESA Release 2.2. Full-pack minidisks are also eligible for caching, with the exception of non-page aligned FBA devices.

Track Caching

In VM/ESA Release 2.2, MDC caches whole tracks instead of individual blocks. The entire track is read with one I/O operation, resulting in improved efficiency and reduced average service time. As a consequence, minidisks no longer need to be formatted into 4-KB blocks.

Generally, when all records on CKD or ECKD tracks have a data length of 256, 512, 1K, 2K, or 4K bytes, the contents of that track are managed in the MDC on a 4-KB block basis. For example, infrequently referenced blocks can be moved to expanded storage or discarded from the cache.

Tracks that do not meet these requirements are managed on a track basis. The whole track is either in central storage, expanded storage, or is removed from the MDC based on average frequency of reference.

Controlling MDC Eligibility

There are three levels of control to determine whether a minidisk should be cached:

1. System level
2. Real device level
3. Minidisk level

For data to be eligible for MDC, all three levels must be enabled. This hierarchy offers CP commands that control which minidisks should be cached, and commands that query caching settings. This improves over the old MDC support, which used the CP Directory and HCPRIO config.

System Level

This is the highest level and controls whether MDC is enabled for the whole system. You can set and query this level by using the following commands:

```
CP SET MDCACHE SYSTEM ON
CP SET MDCACHE SYSTEM OFF
CP QUERY MDCACHE SYSTEM
```

If the system level is off, MDC is disabled for all minidisks in the system, regardless of the two other level settings.

Real Device Level

This is the second level of control for MDC. If MDC is set on at a system level, then cache eligibility of a particular real device can be changed using the CP SET MDCACHE *rdev* command with the following options:

DFLTON Enables MDC for the real devices specified in the command. All minidisks residing on the real devices are MDC-enabled unless you explicitly disable MDC for a range of minidisks using the third level of control. This is the default.

DFLTOFF Disables MDC for the real devices specified in the command. All minidisks residing on the real devices are MDC-disabled unless you explicitly enable MDC for a range of minidisks using the third level of control.

OFF Disables MDC for the real devices specified in the command. All minidisks residing on the real devices are MDC-disabled and cannot be enabled using the third level of control.

FLUSH Clears all data from this device from the MDC.

Minidisk Level

This is the third and lowest level of control for MDC. At this level, MDC can be enabled or disabled for a particular minidisk. To enable a minidisk for MDC, the system level must be enabled and the real device level must not be set to OFF. You can specify the following options:

ON Enables MDC for the specified minidisks.

OFF Disables MDC for the specified minidisks.

FLUSH Clears all data from this minidisk from the MDC.

If no option is specified and neither MDC nor NOMDC is selected on the MINIOPT directory statement, the default is taken. The default enables MDC for minidisks whose real device has DFLTON and disables MDC for minidisks whose real device has DFLTOFF or OFF.

Migration Considerations

Because the enhanced MDC in VM/ESA Release 2.2 lifts a number of restrictions, it is likely that a significant amount of data that was ineligible for MDC is now eligible; the consequences are:

- There may be some minidisks that are poor candidates for MDC, but that did not matter in the past because the type of I/O operations or format made them ineligible for MDC. With several restrictions being lifted, it is worthwhile to revisit these minidisks and make sure they have MDC disabled. Guest paging volumes may be examples of poor candidates.
- If a smaller block size is more suited for a particular minidisk, you may consider reformatting the minidisk. However, 4-KB blocks are often the optimum blocksize for minidisks, for both performance and disk capacity perspectives.
- In VM/ESA Release 2.1, RTM/ESA reports the MDC hit ratio based on the number of blocks (see "RTM Variables" on page 96). In VM/ESA Release 2.2, RTM/ESA reports the MDC hit ratio based on the number of I/O operations. Another important difference in RTM/ESA is the way the hit ratio is calculated. In VM/ESA Release 2.1, the number of hits are divided by the number of blocks related to eligible reads. In VM/ESA Release 2.2, the number of hits are divided by all read operations, except for page, spool, and virtual disks. This can lead to MDC hit ratios that appear lower on VM/ESA Release 2.2 when compared to previous releases, even though minidisk caching may actually be more effective; see a definition of MDC effectiveness in "Selecting Minidisks for MDC Eligibility" on page 100.

  VMPRF reports hit ratios based on blocks and MDC requests (see Figure 32 on page 98).

RTM Variables

The MDCLOG screen, shown in Figure 34, is one of the new screens available in RTM/ESA for VM/ESA Release 2.2. It provides a history of minidisks cache performance and can be used to determine general trends and relationships among the minidisk cache variables. The MDCACHE and XTLOG displays are also updated, now showing how much central and expanded storage is being used for MDC.

The main variables are:

VIRD The total number of virtual read operations per second to DASD devices. This number does not include paging, spooling, nor virtual disk devices.

VIWR The total number of virtual write operations per second to DASD devices. This number does not include paging, spooling, nor virtual disk devices.

RIOD The total number of real I/O operations per second to DASD devices. This number does not include paging or spooling.

RVDR The real to virtual DASD I/O rate, calculated by:

$$RVDR = \frac{RIOD}{VIRD + VIWR}$$

MDIA The total number of real I/O operations per second avoided due to minidisk cache hit.

```
TOD VM/ESA CPU9021 SERIAL 110014 1017M DATE 03/30/94 START 08:00:36 END 11:38:14
H:M  VIRD VIWR RIOD RVDR MDIA MDHR MDRD MDWR MDRM MDIN MDDL MDPF MDAB STMN STXS
1129 1321  180  444  .29 1122  .84 1151  108   10   33   32 5.90  216   95  339
1130  962  117  300  .27  799  .83  810   82 9.88   13   11 1.63  161   17  148
1130  683  103  237  .30  580  .84  617   74 8.28   58   65 4.90   65   17  682
1131  624  110  213  .29  567  .90  583   78 7.43   16   15 6.60   69    0  160
1131  748  135  232  .26  677  .90  698   93 7.45   19   17 6.06   65   34  196
1132  828  172  309  .30  710  .85  758   93 8.09   47   45 8.63   69   42  534
1132 1068  138  331  .27  932  .87  976   94   10   38   40   18   96   60  422
1133  822   90  335  .36  694  .84  708   67   10   11   11 6.80  113   26  136
1133  754   82  252  .30  687  .91  703   58   11   14   12 4.33   78   17  154
1134  712   81  267  .33  624  .87  637   51   12   13   12 1.36   75   26  151
1134  519  105  218  .34  446  .86  468   62 7.44   22   22 2.40   52   66  259
1135  500   86  209  .35  423  .84  448   45 9.82   24   24 6.43   86   34  288
1135  633   45  176  .26  508  .80  521   19   27   14   12 1.16  108   34  141
1136  415   53  101  .21  376  .90  387   26   14   11   11 1.23   30   17  121
1136  415   74  142  .29  357  .86  364   47 7.63 6.80 7.73  .73   54 8.60   81
1137 1098  196  289  .22 1025  .93 1037  106 9.76   12   10 1.93   74   17  122
1137 1572  101  148  .08 1519  .96 1532   46   32   13   15  .56   29   50  134
1138 1416  109  204  .13 1326  .93 1340   57   23   13   13 1.00   72   16  111
AVG   589   93  245  .35  468  .79  486   50 9.70   19   19 3.51  109   51  207
```

Figure 34. RTM MDCLOG Screen

MDHR The MDC hit ratio, calculated by:

$$MDHR = \frac{MDIA}{VIRD}$$

MDRD The total number of virtual read operations per second that read data from the MDC. Excluded operations are noncache eligible reads and other events that prevent MDC read, such as a fair share limit being exceeded.

MDWR The total number of virtual write operations that write to MDC.

MDRM The ratio of the total number of virtual read operations that read from MDC to the number of virtual write operations that write to MDC. It is calculated by:

$$MDRM = \frac{MDRD}{MDWR}$$

MDIN The total count of track inserts per second.

MDDL The total count of track deletions per second.

MDPF The total count of page faults per second requiring a read from DASD.

MDAB The total count of the instances where the minidisk cache interface for a virtual read operation has to be aborted. A cache eligible read is aborted from reading data from MDC if the user's insert capability is off or its fair share insert limit is exceeded. If the request is a SSCH, SIOF, or SIO, then the request can be aborted from reading the MDC if the channel program is too complex for CP to simulate. Ineligible I/O attempts are also counted here.

STMN The total count of central storage pages stolen per second from the MDC.

STXS The total count of expanded storage pages stolen per second from the MDC.

Minidisk Mapping

Minidisk mapping is a way to use data spaces as very large buffers for data stored on 4-KB formatted minidisks. Temporarily placing this data in data spaces could improve application performance if the data is used repeatedly. CP can perform paging I/O more efficiently than most application virtual I/O. The MAPMDISK services are available to any guest operating system that performs I/O operations in 4-KB formatted minidisks.

The minidisk mapping services are provided as a macro for the application to establish the association between a collection of minidisk blocks and a collection of pages of data spaces. Once this association is established, data can be referenced through ordinary CPU instructions rather than I/O instructions.

CP uses the mapping to perform the actual I/O operation when the data is first referenced, or when the corresponding page frames are stolen by CP according to the paging algorithms.

As shown in Figure 2 on page 13, when a mapping exists, an image of the data appears in the associated data space pages without the need to perform application requested I/O operations to load the data. If CP steals the real frame corresponding to the data space page, CP rewrites the page to the proper minidisk block if the page has been changed. When another reference is made to the same page, CP refreshes the contents of the minidisk block into a page frame.

The mapping services are provided by the CP MAPMDISK macro with the following functions:

IDENTIFY Identifies the minidisk pool and the pool-relative block numbers of the minidisk blocks within the pool.

DEFINE Establishes a mapping between a collection of blocks in the minidisk pool and a range of pages in data spaces.

SAVE Requests an asynchronous write of mapped pages to their corresponding minidisk blocks.

REMOVE Removes a mapping between a collection of blocks in the minidisk pool and a range of pages in data spaces.

Your virtual machine must be in XC mode to successfully use any of the functions of this macro. If data spaces are used (other than your primary address space), access-register mode must be established in order to change the mapped data space pages. If your machine is in 370 mode, then an operation exception is recognized. If your machine is in XA mode or ESA mode, then a specification exception is recognized.

Upon completion of the DEFINE operation, the data that is contained on the minidisk blocks is available in the newly-mapped pages. However, actual movement of the data from a minidisk block to the corresponding address space page does not occur until the first reference is made to the page.

Since it is unpredictable if or when a page will be stolen, the contents of a page in storage and the corresponding minidisk block are not always the same. Because of this unpredictability, when you use the minidisk mapping services you must understand that:

- You do not know whether changes committed to mapped minidisk blocks are actually written back to the minidisk. The SAVE function of the MAPMDISK macro can be used to ensure predictable saving of changed data. This function is asynchronous to the application execution.
- Actual I/O operations (SIO, SSCH, DIAGNOSE) may be used for mapped minidisk blocks, but is not recommended. I/O operations are not necessary, because the normal method of updating data on a mapped minidisk block is changing the contents of the associated mapped page. However, if an I/O operation must be used, be aware that:
  - An I/O operation used to read a mapped minidisk block may access old data.
  - If you write to a mapped minidisk block corresponding to an updated page that has not been saved, one of the two updates will be lost. Therefore, use the following sequence:
    1. Save the page with the SAVE macro and wait for the save-completion signal.
    2. Perform the actual I/O operation.
    3. Execute DIAGNOSE X'10' (release page) against the page to make the changed data visible in the mapped page.

When the SAVE function is used, CP uses special external interrupts to notify the virtual machine of the save-completion signal. The application should provide an external-interrupt-handler routine and set both the external interrupt mask (bit 7 of the PSW) and the subclass-mask (bit 22 of control register 0).

Figure 35 shows how to code these macros and the associated information required. Note in the example that blocks of five minidisks (X'120', X'250', X'220', X'320', and X'430') are mapped to one data space. If you want to map blocks of the same minidisk to different data spaces, then you need to code more than one MAPMDISK DEFINE macro.

Mapping a CMS user minidisk to data spaces is not a practical idea, because the application normally does not know which CMS blocks are used to contain the file blocks. This task is managed by the CMS file system, which allocates file blocks anywhere in the minidisk according to the allocation map maintained in the minidisk.

Even assuming that the application includes code to search the file directory and the allocation map to manage the file system by itself, the application could not use the file system macros to access data because they would perform actual I/O operations.

One example of CMS applications that may benefit from the mapping services are servers that access minidisks allocated with the CMS RESERVE command. RESERVE is used to allocate all available blocks in a minidisk to a single CMS file. The server can map all minidisk blocks to pages of data spaces and keep the mapping while resolving requests from several users. The CMS DISKID function may be used to find the offset of the first free minidisk block.

The SQL/DS licensed program (see “SQL/DS” on page 14) and the CMS Shared File System (see “Shared File System” on page 16) both use minidisk mapping services in their implementation of VM data space exploitation.

```
MAPPING  CSECT
         LAE   12,0(15)            Establish program
         USING MAPPING,12          addressability.
         ST    14,SAVE             Save return address.
      ADRSPACE CREATE,NAME='MAPSPACE',SIZE=SIZE,ASIT=ASIT,KEY=KEY
      MAPMDISK IDENTIFY,EXTCT=NUMEXT,XLDBLOCK=XBLOK1
      MAPMDISK DEFINE,ASIT=ASIT,PAGECT=SIZE,PAGEADDR=FPAGE,PRBN=PRBN
         SAC   512                 Enter access-register mode.
         .....
         .....                     Application processing.
         .....
         SAC   0                   Exit  access-register mode.
      MAPMDISK SAVE,BLOCK=SBLOK,ENTCT=ENTRIES
      MAPMDISK REMOVE,ASIT=ASIT,PAGECT=SIZE,PAGEADDR=FPAGE
EXIT     L     14,SAVE             Return to
         BR    14                  CMS.
         DEFWORKA                  Work area.
SAVE     DS    F                   CMS save area.
SIZE     DC    F'3300'             Required address space pages.
ALET     DS    F                   ALET.
ENTRIES  DC    F'20'               Total number of pages to save.
BUFFER   DS    D                   Error buffer for SAVE.
SBLOK    EQU   *                   List for SAVE function.
ASIT     DS    D                   ASIT.
         DC    CL4'TOKN'           Token for SAVE interrupt.
         DC    A(0,0)              Reserved.
         DC    A(BUFFER)           Error buffer.
         DC    A(10*4096)          First page address to save.
         DC    A(10)               Number of pages to save.
         DC    A(40*4096)          First page address to save.
         DC    A(10)               Number of pages to save.
FPAGE    DC    F'0'                First mapped page.
PRBN     DC    F'1'                First minidisk block.
NUMEXT   DC    F'10'               Number of extents
KEY      DC    X'E0'               Data space storage key.
         DS    0D                  Must be DW aligned.
* First extent (5 entries)
XBLOK1   DC    A(0)                Reserved.
         DC    A(XBLOK2)           Pointer to next extent.
         DC    A(5)                Number of entries in extent.
         DC    A(0)                Reserved.
         DC    A(0001)             Pool relative block number.
         DC    A(150)              Minidisk relative block number.
         DC    A(300)              Number of blocks mapped.
         DC    AL2(X'120')         Virtual device address.
         DC    AL2(0)              Reserved.
         DC    A(0301,750,500),AL2(X'120',0)  Second entry.
         DC    A(0801,300,200),AL2(X'250',0)  Third entry.
         DC    A(1001,650,600),AL2(X'250',0)  Fourth entry.
         DC    A(1601,000,300),AL2(X'220',0)  Fifth entry.
* Second extent (2 entries)
XBLOK2   DC    A(0,XBLOK3,2,0)
         DC    A(1901,000,300),AL2(X'320',0)  First entry
         DC    A(2201,450,500),AL2(X'320',0)  Second entry.
* Third and last extent (3 entries)
XBLOK3   DC    A(0,0,3,0)
         DC    A(2701,000,200),AL2(X'430',0)  First entry.
         DC    A(2901,500,200),AL2(X'430',0)  Second entry.
         DC    A(3101,800,200),AL2(X'430',0)  Third entry.
         END
```

Figure 35. Minidisk Mapping Example

Virtual Disk in Storage

A virtual disk in storage appears to be an FBA minidisk that may be used and shared with other virtual machines. Guests, servers, and general users can take advantage of virtual disks with no changes to existing applications. Virtual disks are completely transparent to programs that use FBA devices; these programs do not have to be recoded or modified in order to exploit the function.

Good performance is attained by allocating FBA minidisks in virtual storage, rather than on a real disk, to avoid the overhead of real I/O operations, as shown in Figure 36. A virtual disk in storage provides fast access, but it is temporary and volatile, in the same way as a temporary disk (TDISK).

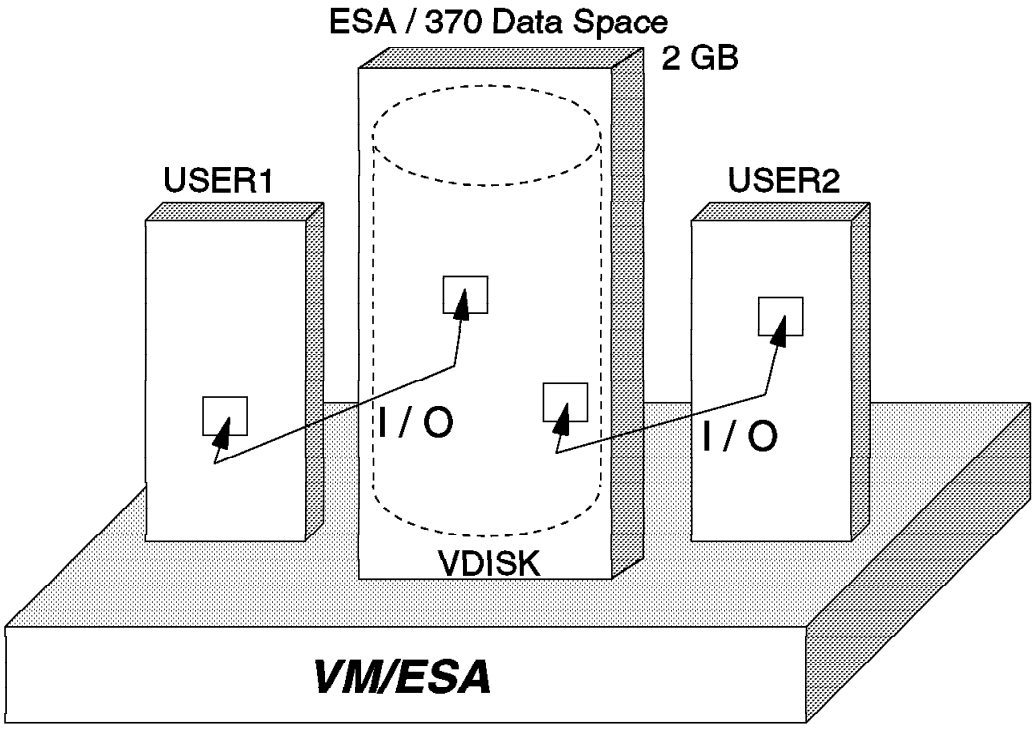

Figure 36. Virtual Disk Implementation

The VSE lock file, which is usually a source of contention among VSE systems that share DASD files, is an excellent candidate to reside on a virtual disk. This usage of a virtual disk requires that all VSE operating systems sharing the lock file are running under the same VM/ESA.

CMS temporary disks, compiler, and sort work files are also good candidates to reside on virtual disks, but you may want to control the maximum space available to each user in order not to overload the paging subsystem.

Virtual disks can be defined in the directory with the MDISK statement; when they are, they can be shared among several virtual machines. They can also be defined like temporary minidisks with the CP DEFINE command, but no sharing capability is available in this case.

Implementation

Virtual disks are implemented by CP in an ESA/370 address spaces; they do not require the VM data space function available in the ESA/390 architecture. Two consequences of this implementation are:

- The maximum size of a virtual disk is 2 GB.
- Virtual disks can be used in any system capable of running VM/ESA Release 2.1, such as the IBM 4381 Models 9XE, the ES/3090 processors, and the ES/9000 family.

The following I/O instructions and functions are simulated for virtual disks:

- SIO and SIOF
- SSCH
- DIAGNOSE codes X'20', X'A4', X'A8', and X'250'
- *BLOCKIO system service

All normal minidisk query mechanisms, such as DIAGNOSE X'24', DIAGNOSE X'210', QUERY *vdev*, Sense ID, and Read Device Characteristics CCW, also apply to virtual disks. Appropriate values are returned except for the real device type, which is always considered to be 9336.

There are two ways to define a virtual disk:

- Using an MDISK statement in the user directory

  Virtual disks defined with MDISK directory statements can be shared among several virtual machines. Virtual Reserve/Release is provided by CP when *V* is coded in the link mode, as in the following example:

  `MDISK` *`vaddr`* `FB-512 V-DISK` *`nnnnnn`* `MWV`

  CP LINK and CP DETACH commands work as they do for a normal minidisk. Access control is the same as for a normal minidisk; through read, write, and multiple-write passwords, but RACF or an external security manager can be used as well.

  The first user to log on, either with an MDISK or LINK specifying V-DISK, causes the virtual disk to be created; a successful CP LINK command to a virtual disk also causes it to be created. The last user to detach the virtual disk, either by DETACH or LOGOFF, causes it to be deleted.

- Using the CP DEFINE command

  Virtual disks created by the CP DEFINE command are private to the user issuing the command. A private virtual disk is like a temporary minidisk; it lasts for the session and cannot be shared. A private virtual disk is deleted when the owner detaches it or logs off.

Real Storage Considerations

The virtual disk implementation does not require any special hardware resources; it uses only storage resources with the following characteristics:

- CP paging subsystem is used for paging virtual disks.
- Each gigabyte allocated for a virtual disk uses 1024 pages for page tables and one page for the segment table.
- Virtual disk page tables are not pageable.

Guidelines

The guidelines below may help you to use virtual disks effectively:

- The best candidates are heavily used temporary minidisks, especially if the amount of referenced data is small.
- Do not use more virtual disks than necessary.

  Each virtual disk has one or more fixed (nonpageable) pages associated with it, thus increasing the load on central storage.
- If necessary, increase the amount of paging space.

  Although VMPRF does not have a specific report, you can determine how many page slots are being used for virtual disks from the VMPRF PRF092 report, as shown in Figure 37; virtual disks address space names start with VDISK$ followed by the owner's userid.
- Detach virtual disks as soon as possible.

  When a private virtual disk is detached, or a shared one is detached by the last user, all corresponding page frames are released.
- Set system and user limits to prevent excessive use.

  You can set this limit in the system configuration file or with the CP SET VDISK command. The system limit protects the overall system performance by not allowing over-commitment of real storage through excessive use of virtual disks. The user limit, which applies only to private virtual disks, prevents any one user from acquiring an excessive amount of space.

  The system built-in default has the following objectives:

  - Virtual disk space should not exceed 25% of the available page space.
  - Nonpageable load on real storage should not exceed 25% of the DPA.

  The QUERY VDISK command can be used to display a list of all existing virtual disks and the system or user limits.
- Monitor system performance.

  Monitor overall system performance and system paging rate before and after virtual disks are made available. If overall performance decreases due to increased paging, you should decrease the system limit for virtual disk usage. If, on the other hand, the system limit is frequently being reached and overall performance is good, you may increase the system limit.

```
PRF092  Run 09/10/1993 14:01:41        DATA_SPACES_SHARED                                        Page   13
                                Shared Data Spaces - Paging Activity
From 09/10/1993 09:22:52                                                          GG243934
To   09/10/1993 12:14:52                                                          CPU 9121     SN  20415
For  10320 Secs 02:52:00              VM/ESA STORAGE MANAGEMENT MEASUREMENTS            VM/ESA   21.01 SLU 9303
_____________________________________________________________________________________________________________

                                          <------- Rate per second at which Pages are ------->

                                                 Rd: DASD Wr: Cstr Rd: Xstr Wr: Cstr Mg: Xstr                DASD
Owning                                  Users         -to-Cstr -to-DASD -to-Cstr -to-Xstr -to-DASD Resident   Pages     Page
Userid   Address Space Name      Shared Permt   Stolen (PgRead) (PgWrit) (PageIn) (PagOut) (PgMigr)   Frames  Locked    Slots

SYSTEM   ISFCDATASPACE           Yes       0       0       0       0       0       0       0       0      0       0
SYSTEM   PTRM0000               Yes       0   33.592    0.180    0.181   33.219   33.221    0.003    -2349      0     198
SYSTEM   SYSTEM                 Yes       0    0.569    0.076    9.143    0.419    0.474    0.028      373    366     452
SYSTEM   VDISK$CMSUSER$$0999$0001 Yes       0    2.232    2.149    2.176       0    1.530    1.474      143      0    1056
SYSTEM   VIRTUAL$FREE$STORAGE     Yes       0    0.073    0.011    0.012    0.061    0.062    0.001        1      0      -3

Sum/Mean                                    7.293    0.483    2.302    6.740    7.058    0.301     -367     73     341
```

Figure 37. VMPRF PRF092 Report

Comparison

Minidisk caching is one of the data-in-memory facilities provided by VM/ESA, together with minidisk mapping (MAPMDISK) and virtual disks in storage.

Table 12 is a comparison of these facilities; the main conclusions are:

- MDC requires no user effort, but its scope is limited to applications that have a reasonable read-to-write ratio.

  MDC should be the main facility for general users that require performance gains in reading permanent data.

- Minidisk mapping provides excellent performance for both reads and writes, but requires considerable programming effort; it is limited to CMS applications running on XC-mode virtual machines.

  Because this facility requires complex Assembler language coding, it is not intended for the general user; it is intended for server virtual machines, such as SQL/DS and SFS file pool servers.

- VDISK provides excellent performance for both reads and writes, but it is limited to volatile data and to guests that support FBA devices.

  VDISK should be the main facility for users that use temporary data in FBA devices. The best candidates for virtual disks are relatively small and heavily used minidisks, such as the VSE lock file.

You can combine these facilities with other performance options. For example, using MDC for a minidisk that is defined behind a cached storage control caches the most recently read minidisk blocks in expanded in central storage and caches the next most recently read minidisk blocks in the storage control.

Table 12. Characteristics of Data-in-Memory Facilities

Characteristic	Data-in-Memory Facility			
	MDC R2.1	**MDC R2.2**	**MAPMDISK**	**VDISK**
Application enabling	Transparent	Transparent	Assembler Language	Transparent
Added complexity	No	No	Yes	No
Disk supported	Non-full-pack 4-KB formatted minidisk	3380, 3390, 9345, RAMAC, FBA	4-KB formatted minidisks	Simulated FBA minidisks
Maximum mdisk size	No limit	No limit	No limit	2 GB
Instructions used to access data	DIAGNOSE I/O *BLOCKIO	DIAGNOSE I/O *BLOCKIO SIO, SIOF, SSCH	Storage references made with normal architected instructions	DIAGNOSE I/O *BLOCKIO SIO, SIOF, SSCH
Machine mode	Any	Any	XC	Any
Disk update	Store-through	Store-through	Store-in	Not applicable
Integrity	Automatic	Automatic	Application responsibility	Volatile
Performance gains	Read	Read	Read and write	Read and write
Paged out to	Minidisk	Minidisk	Minidisk	Paging areas

Chapter 7. Storage Administration

This chapter provides a brief description of the VM/ESA scheduler and the commands you can used to understand and tune your storage resources. It describes each command and provides some guidelines about its usage and its effects on the scheduler behavior.

Two tables at the end of this chapter provide guidelines and threshold values that should cause you to investigate when they are exceeded; these tables refer to VMPRF and RTM/ESA variables and reports.

Information about CP commands can be found in *VM/ESA Performance* and *VM/ESA CP Command and Utility Reference*. Information about RTM and VMPRF reports can be found in *RTM VM/ESA Program Description/Operations Manual* and *VMPRF User's Guide and Reference*.

VM/ESA Scheduler

The VM/ESA scheduler optimizes the use of the central processors and the storage resources. The VM/ESA scheduler minimizes the response time of interactive users and, at the same time, favors some virtual machines based on your instructions. The basic scheduler goals are to:

- Identify the user's characteristics and dynamically classify the user in one of four classes.
- Favor the response times of interactive users over noninteractive users. An interactive user has short-running transactions or interacts with the terminal while a transaction is in progress.
- Move a user from one class to another according to the user's behavior.
- Calculate the user's priority and, based on this priority, insert the user in the eligible and dispatch lists.
- Favor certain virtual machines or groups of virtual machines based on your instructions.

Note: In this discussion, “user,” “virtual machine,” and “VMDBK” are generally used as synonyms. Keep in mind, however, that an MP virtual machine has multiple VMDBKs associated with it, one for each virtual CPU.

Scheduler Lists

The VM/ESA scheduler distributes virtual machines among three lists:

Dormant List The dormant list groups idle users; these are virtual machines that have no work to do or that are waiting for the completion of a long event, such as terminal I/O.

Eligible List The eligible list groups users that became active and are candidates to enter the dispatch list. One of the criteria for moving a user to the dispatch list is the availability of central storage for its working set; the continuous existence of eligible lists usually indicates the need for more central storage.

VM/ESA calculates the available central storage by subtracting the WSSs of the users in the dispatch list from the dynamic paging area size. A user in the eligible list may be promoted to the dispatch list if its WSS fits in the available central storage; see "WSS Fit Logic" on page 133 for additional considerations.

Limit List The limit list is a subset of the dispatch list and contains the users that have exceeded their maximum share. Depending on the option specified in the SET SHARE command, the user must not absorb any additional processor resource when the maximum share is reached; see "SET SHARE" on page 119 for details.

Dispatch List The dispatch list groups the multiprogramming users that compete for resources. Virtual machines remain dispatchable until they either complete their time slices or become idle. A user is dropped from the dispatch list and moved to the eligible list when the time slice expires. A user is dropped from the dispatch list and moved to the dormant list when the user becomes idle. The number of users in the dispatch list is called the multiprogramming level.

Figure 38 illustrates the lists of users (VMDBKs) maintained by the scheduler and indicates their flow from one list to another.

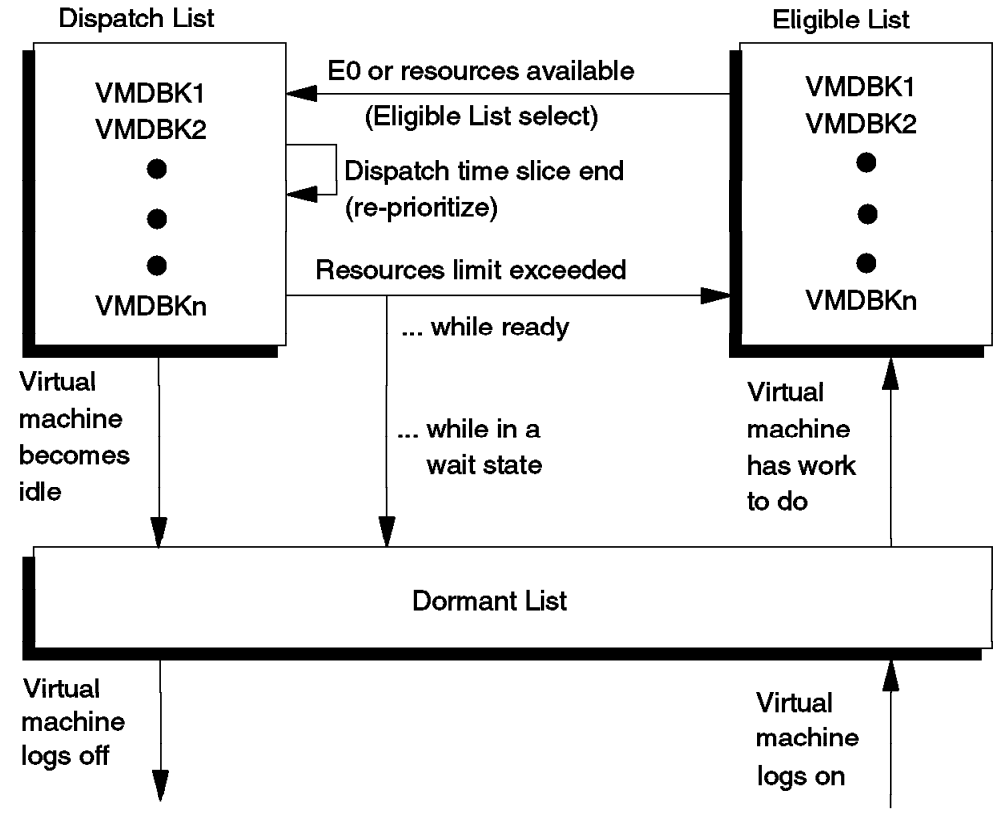

Figure 38. Virtual Machine Scheduling Flow

Users (VMDBKs) normally are in the dormant list and they are moved to the eligible list when they have work to do. The number of VMDBKs in the dormant list is generally much larger than either the eligible or dispatch lists.

When a user enters the eligible list, it is assigned an eligible list priority based on its share, its resource requirements, and the contention for resources on the system.

Eligible list priority is calculated as the time-of-day (TOD) plus an offset, calculated when users are added to the list, and represents the adjusted TOD at which each user is expected to be moved to the dispatch list. In this way, users already waiting for a long time have better priority than a user recently added to the list. The adjusted TOD is a clock maintained by the scheduler that runs at the same rate as the TOD clock, but stops during system overhead time.

As resources become available, users are moved from the eligible list to the dispatch list, where they are prioritized by the scheduler. The dispatcher uses the priority in selecting users to run.

The promotion to the dispatch list occurs when all performance criteria established by SHARE, STORBUF, LDUBUF, and DSPBUF settings are met (except for E0 users); see “WSS Fit Logic” on page 133 for details.

As users consume resources in the dispatch list, they are examined and re-prioritized after each dispatch time slice (DSPSLICE). Once they consume a given amount of resource (elapsed time slice expired, or the working set is too big as discussed in “SET SRM MAXWSS” on page 128), or become idle, or in rare cases, are preempted in favor of certain E1 users, they move back to the eligible list if still ready, or to the dormant list if not ready.

Dispatch list priorities are calculated as the adjusted TOD plus an offset, and represents the adjusted TOD at which the user is expected to finish its next dispatch time slice. The dispatch list priority for each user is calculated when the user is added to the dispatch list and at the end of each dispatch time slice.

Scheduler Classes

The scheduler classifies the users in one of four classes. These classes are named Q0, Q1, Q2, and Q3 when the user is the dispatch list, E0, E1, E2, and E3 when the user is the eligible list.

We generally refer to a user as a class 1 user when it is either on Q1 (dispatch list) or E1 (eligible list). The basic VM/ESA scheduler goal is to provide fast response times to class 1 users.

VM/ESA gives class 1 users faster access to the dispatch list. However, they stay in the dispatch list for only a short time. Class 2 and class 3 users have low priority to access the dispatch list but they stay there longer. Class 0 users are special cases, as discussed later.

A user promoted to the dispatch list is allowed to stay there for a major time slice (also called the elapsed time slice, because it is an elapsed time). During the major time slice, the user is allowed to use the processor during a minor time slice (also known as the dispatch time slice) after which priorities are re-calculated; any other user may be dispatched for the next minor time slice.

Class 1

Users are classified as class 1 when they are starting a new transaction. The scheduler assumes that a user is starting a new transaction if the user returns to the ready state after being in a wait state with no outstanding I/O for more than 300 milliseconds. If it returns in less than 300 milliseconds, it is continuing its previous transaction.

VM/ESA dynamically adjusts the elapsed time slice in such way that 85% of all transactions can finish during one elapsed time slice, which may range from 50 milliseconds to 16 seconds. This basic elapsed time slice is called the Q1 elapsed time slice.

Since class 1 users are frequently interacting with the terminal, you should monitor and avoid many E1 users by increasing their share of system resources. However, E1 users tend to have better priorities because priorities basically depend on the major time slice, which are much shorter for E1 than for E2 or E3 users.

Class 2

This user is expected to have transactions of medium length. A user is classified as class 2 when it does not complete its transaction during one Q1 elapsed time slice.

The user is classified as E2 and is promoted to the dispatch list as Q2. It is allowed to stay in the dispatch list for a Q2 elapsed time slice, which is eight times longer than a Q1 time slice.

E2 users tend to have better priorities than E3 users because major time slices are much shorter for E2 users.

Class 3

This user is expected to have long-running transactions. A user is classified as class 3 when it does not complete its transaction during one Q2 time slice.

The user is classified as an E3 user and is promoted to the dispatch list as Q3. It is allowed to stay in the dispatch list for a Q3 elapsed time slice, which is the larger of:

- 48 times a Q1 time slice
- 4 times the time required to page in its working set

After being classified as Q3, the user continues as a class 3 user until the end of its transaction.

Class 0

A class 0 user is a favored user that stays in the eligible list just long enough to have its projected working set sizes calculated. These users are then promoted to the dispatch list regardless of any promotion criteria that may be in effect. This class may result from the QUICKDSP option or because the transaction has been identified as a hot-shot or a lock-shot transaction. See “SET QUICKDSP” on page 128 and “Bias Factors” on page 131 for more information.

Class 0 users in the limit list (L0) are kept in that list in such a way that they can absorb the scheduler resources as specified in the SET SHARE command; see “SET SHARE” on page 119 for details.

Tuning Commands

The commands described in this section have complex effects on VM/ESA behavior. Since they interact with each other, it is not uncommon that one command changes the effect of another command. Generally, you should leave the VM/ESA defaults alone unless you are prepared to measure and understand the effect of the modifications. You should carefully log the modifications and monitor VM/ESA behavior before and after each modification. However, we recommend that you change some defaults that are known to be too conservative.

Make only one modification at a time; otherwise it is difficult or impossible to determine the effect of each change on the system. In fact, if some changes improve performance and others degrade performance, they can effectively cancel each other and lead to erroneous conclusions.

Many of the commands described below may be used to favor some users. You should favor only a few virtual machines. Otherwise, the normal VM/ESA scheduling mechanisms will not work properly.

SET SHARE

The SET SHARE command, as well as the SHARE directory statement, sets the VM/ESA objectives for the allocation of resources to a user. VM/ESA uses the share and the actual resource consumption to determine the user's priority and position within the various scheduler queues.

A share setting can be absolute or relative. First, VM/ESA allocates the absolute shares to the users. The remaining resources are then prorated among the users with relative shares. If the sum of all absolute shares is greater than 100%, VM/ESA uses a maximum of 99% and prorates the absolute shares accordingly.

As you can see in Figure 39, you can specify two share settings in the SET SHARE command. The first (or only) setting is the target minimum share for the specified user. The second setting (if specified) represents the maximum share. Note that the absolute share can be specified in tenths of one percent.

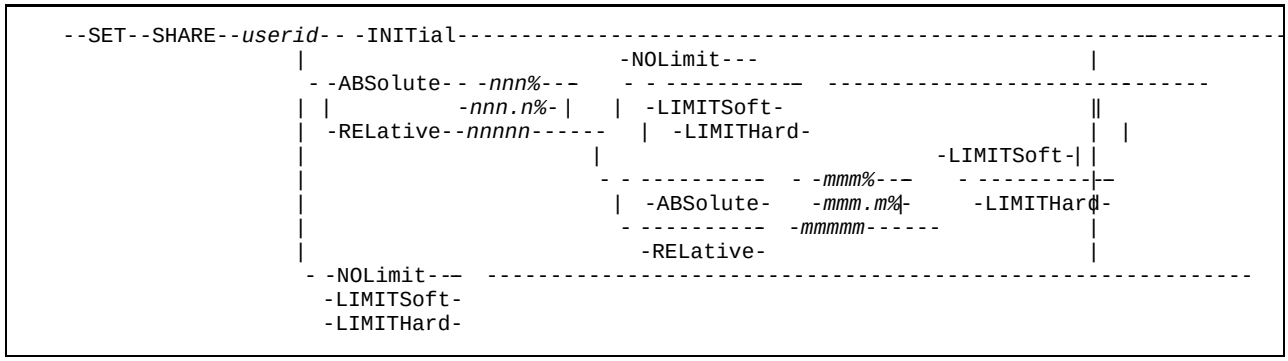

Figure 39. CP SET SHARE Command Syntax

Minimum Share

VM/ESA allocates three resources to the virtual machine according to its target minimum share. Processor time, central storage, and paging I/O capability are allocated proportionally to the user's share when these resources become a bottleneck. VM/ESA does not directly control the data I/O subsystem.

Figure 40 shows some examples with the corresponding answer from the system. Let us assume that you have issued the commands shown in Figure 40.

```
set share mvs absolute 50%
USER MVS     : ABSOLUTE SHARE= 50%
               MAXIMUM SHARE= NOLIMIT

set share vtam relative 800
USER VTAM    : RELATIVE SHARE= 800
               MAXIMUM SHARE= NOLIMIT

set share cms1 relative 100
USER CMS1    : RELATIVE SHARE= 100
               MAXIMUM SHARE= NOLIMIT

set share cms2 relative 100
USER CMS2    : RELATIVE SHARE= 100
               MAXIMUM SHARE= NOLIMIT
```

Figure 40. Minimum Share

If central storage becomes the scarcest resource, then the VM/ESA objective is to give 50% of the central storage to user MVS, 40% to the VTAM virtual machine, and 5% to each of the CMS users. These shares are also called normalized shares.

In the examples shown in Figure 40, if the specified share is below the user's needs you would be restricting resource usage only if there are not enough resources for all users. This was the only possible way to set shares in VM/ESA Release 2.1 and prior releases. In VM/ESA Release 2.2, this behavior was chosen using the minimum share only and NOLIMIT by default.

Surplus

When a virtual machine's actual resource usage is less than its normalized share, the difference is considered surplus.

In VM/ESA Release 2.1 and prior releases, the scheduler did not distribute the surplus in an explicit manner and the surplus tended to be given to the users with the largest share setting. The interactive users, who normally have less share, seldom benefited from the surplus resources.

In VM/ESA Release 2.2, the surplus is given out proportionally to the normalized shares. If applications were dependent on getting surplus share to run acceptably in VM/ESA Release 2.1, the system may need to be re-tuned now that surplus is distributed proportionally. A proper tune, however, should rely on correct settings of share commands and not on possible surplus. Surplus should be considered as bonus on top of the expected performance.

Maximum Share

If you specify a second share setting, you are also requesting a target maximum of the scheduled processor resource. The processor is the only resource to be allocated according to the maximum share. You can also specify the options NOLIMIT, LIMITSOFT, and LIMITHARD as shown in the examples in Figure 41:

NOLIMIT There is no maximum share limit imposed; the effect is the same in VM/ESA Release 2.1 and prior releases.

LIMITSOFT This user is limited to the maximum share unless there is surplus and there is no other user with NOLIMIT that can use the surplus. If there are NOLIMIT users, the LIMITSOFT users are limited to their maximum and the NOLIMIT users get all the surplus available.

If a maximum share is not specified, the user's minimum share is also its maximum share of the processor resource.

LIMITHARD This user is limited to the maximum share even if there is surplus available.

If a maximum share is not specified, the user's minimum share is also its maximum share of the processor resource.

```
set share mvs absolute 50% limitsoft
USER MVS    : ABSOLUTE SHARE= 50%
              MAXIMUM SHARE= LIMITSOFT ABSOLUTE 50%

set share mvs absolute 10.5% absolute 20% limitsoft
USER MVS    : ABSOLUTE SHARE= 10.5%
              MAXIMUM SHARE= LIMITSOFT ABSOLUTE 20%

set share mvs absolute 10.5% 20% limithard
USER MVS    : ABSOLUTE SHARE= 10.5%
              MAXIMUM SHARE= LIMITHARD ABSOLUTE 20%

set share mvs relative 5000 10000 limithard
USER MVS    : RELATIVE SHARE= 5000
              MAXIMUM SHARE= LIMITHARD RELATIVE 10000

set share mvs relative 5000 20% limithard
HCPSRM002E Invalid operand - 20%

set share mvs relative 5000 absolute 20% limithard
USER MVS    : RELATIVE SHARE= 5000
              MAXIMUM SHARE= LIMITHARD ABSOLUTE 20%
```

Figure 41. Minimum and Maximum Share

In general, you should use relative shares instead of absolute shares. Specify a high relative share to server virtual machines, such as VTAM. Excessively high shares, 10000 for example, is most often the cause of surplus share problems.

RTM/ESA USER screen displays the share of all users (variable SHARE). The XSLOG screen displays the sum of the absolute (variable ABSH) and relative (variable RLSH) shares of users in the dispatch list.

SET SRM STORBUF

The SET SRM STORBUF command allows you to control the access of the users to the central storage. This command establishes one of the criteria for promotion from the eligible list to the dispatch list. This is the basic command to determine the commitment of central storage by V=V virtual machines.

When the response times are increased by paging wait times, you may reduce the commitment of the central storage by varying the relation of the three parameters. You can also reduce the access to central storage by users of a specific class.

The default settings for VM/ESA Release 2.1 and older releases is 100% 85% 75%. These settings are low for most of the systems and the VM/ESA Release 2.2 default was changed to:

```
SET  SRM  STORBUF  125%  105%  95%
```

VM/ESA interprets this command as shown in Figure 42:

1. The first percentage value (125%) is a high threshold value for adding any virtual machine to the dispatch list. If the percentage of storage corresponding to the sum of the working sets for all Q1, Q2, and Q3 virtual machines plus the working set of the virtual machine under consideration exceeds the first percentage value, the virtual machine is not added to the dispatch list.
2. The second percentage value (105%) further restricts the promotion of E2 and E3 virtual machines. If the percentage of storage corresponding to the sum of the working sets for all Q2 and Q3 virtual machines plus the working set of the candidate E2 or E3 virtual machine exceeds the second percentage value, the E2 or E3 virtual machine is not added to the dispatch list.
3. The third percentage value (95%) further restricts the promotion of E3 virtual machines. If the percentage of storage corresponding to the sum of the working sets for all Q3 virtual machines plus the working set of the candidate E3 virtual machine exceeds the third percentage value, the E3 virtual machine is not added to the dispatch list.

The above percentages tell VM/ESA to promote users from the eligible list to the dispatch list in such way that all three conditions below apply:

- The sum of the working set sizes of Q1, Q2, and Q3 users does not exceed 125% of the dynamic paging area.
- The sum of working set sizes of Q2 and Q3 users does not exceed 105% of the dynamic paging area.
- The sum of working set sizes of Q3 users does not exceed 95% of the dynamic paging area.

An alternative view of the example above is to consider storage buffers reserved to different transaction classes. The 20% of storage (125-105) represents a storage buffer available to Q1 virtual machines that E2 and E3 virtual machines are not allowed to occupy. An E1 virtual machine competes only against Q1 virtual machines to occupy this 20% of the storage buffer.

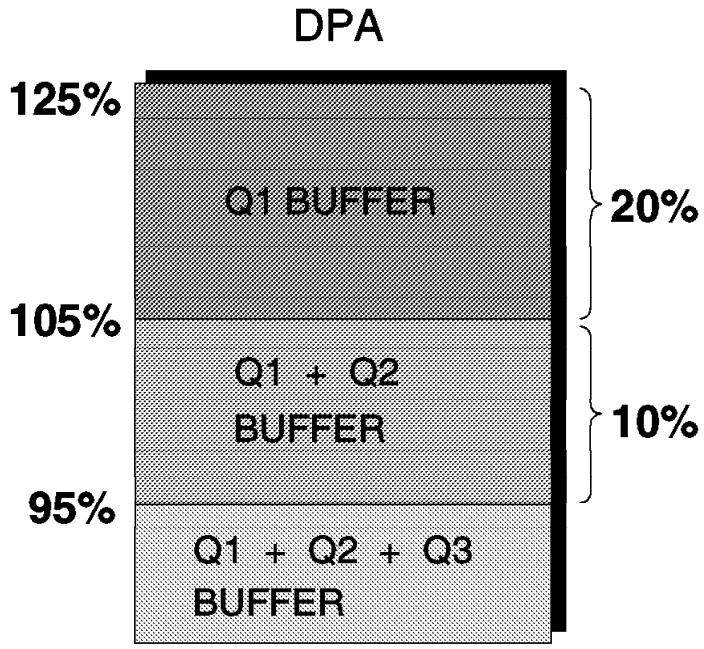

Figure 42. SET SRM STORBUF Command

The 10% of storage (105-95) represents a storage buffer available to Q1 and Q2 virtual machines that E3 virtual machines are not allowed to occupy. An E2 user competes against Q1 and Q2 users to occupy this 10% of storage buffer.

Class 0 users are not subject to this restriction to be promoted to the dispatch list. However, the central storage they occupy reduces the actual DPA available to the normal users. You may also create a buffer for Q0 users by under-committing the DPA, specifying the first percentage as less than 100%.

Note that when the paging system is efficient, VM/ESA is able to hold a sum of WSSs larger than the DPA. Therefore, a 100% or lower percentage is actually an under-commitment of central storage, and a Q0 buffer exists in practice.

When the paging system is efficient, you may over-commit the central storage by specifying percentages greater than 100%, as the default values already specify. Over-commitment may be necessary when there are a few large V=V guests competing for resources; the 95% default for class 3 can lock out other machines when one or more of these guests are dispatched. The over-commitment of the DPA is highly recommended when expanded storage is used for paging.

The guideline is to consider part of the expanded storage as though it were central storage. Suppose that you have 128 MB of central storage with 100 MB being used for DPA, and 128 MB of expanded storage with 60 MB being used for paging. As a starting point, you may assume that the DPA is 160 MB and over-commit the DPA by 60%, as exemplified below (later, you may adjust the percentages more precisely):

```
SET  SRM  STORBUF  160  136  120
```

See “SET SRM XSTORE” on page 129 for an alternative to the above example.

SET SRM LDUBUF

This command controls the commitment of the paging I/O subsystem and is the primary command to discriminate against V=V virtual machines with high paging requirements. The loading user buffer is the number of positions in the dispatch list that can be occupied by loading users at the same time.

VM/ESA expects high DASD paging activity from users that recently caused high paging I/O rates. VM/ESA also expects users whose working sets have been moved to DASD auxiliary storage to be loading users because their working sets must be brought back to central storage when they return to the dispatch list. A virtual machine is classified as a loading user by the scheduler when:

- It has just logged on.
- It has five or more page faults during one dispatch time slice.
- It has had referenced pages stolen during a transaction, and thus is expected to page them back in.

A virtual machine that is paging heavily to expanded storage is not necessarily a loading user. Paging to expanded storage does not use the paging I/O resources and is therefore not affected by the SET SRM LDUBUF command. The INDICATE QUEUES EXP command shows who the loading users are.

During system initialization, CP determines the number of DASD page exposures (expanded storage is not considered). VM/ESA assumes that each loading user keeps one paging exposure busy. Let us suppose that you issue the following command and have five auxiliary storage DASD devices used for paging.

```
SET  SRM  LDUBUF  300  200  100
```

VM/ESA permits 300% of the five paging exposures to be committed by loading users, in such way that all three conditions below apply, as shown in Figure 43:

- The sum of Q1, Q2, and Q3 loading users does not exceed 15.
- The sum of Q2 and Q3 loading users does not exceed 10.
- The number of Q3 loading users does not exceed five.

As with the STORBUF command, the LDUBUF command sets aside a buffer for loading users.

Figure 43 shows that five dispatch list positions (20-15) represent a buffer for Q1 loading users that E2 and E3 virtual machines are not allowed to occupy. An E1 loading user competes only against Q1 virtual machines to occupy one of these five slots.

In the same manner, five dispatch list positions (10-5) represent a buffer for Q1 and Q2 loading users that E3 virtual machines are not allowed to occupy. An E2 loading user competes against Q1 and Q2 users to occupy one of these five slots.

The initial default, 100%, 75% and 60%, is normally low; usually, the DASD paging system is efficient and can be over-committed. Therefore, in most situations, the recommended LDUBUF setting is the one exemplified above.

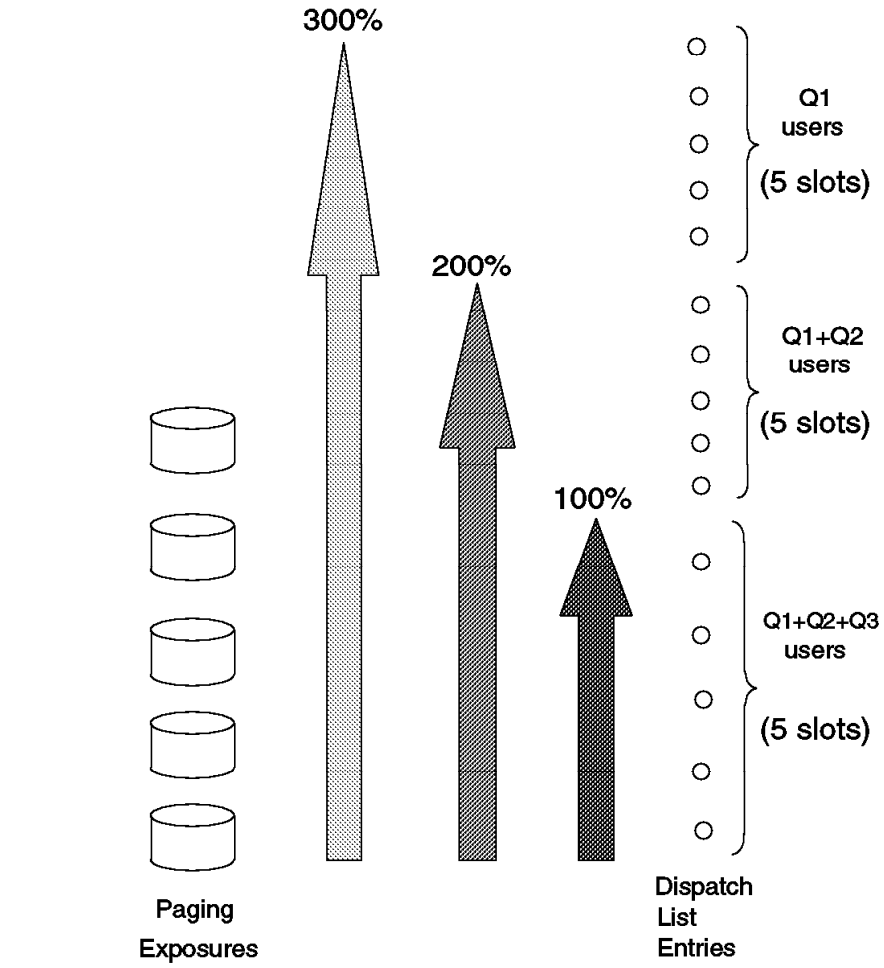

Figure 43. SET SRM LDUBUF Command

Note the following:

- A virtual machine that pages to expanded storage is not a loading user; thus, it is not affected by the SET SRM LDUBUF command.
- Paging is often the most influential component of overall response time. In general, a transaction uses a predictable amount of processor time and I/O time, regardless of system activity. However, the number of paging operations can be quite different in various executions of the same transaction.
- Class 0 users are not affected by this command; they are never kept waiting in the eligible list. The Q0 class can be established through the SET QUICKDSP command, as discussed in “SET QUICKDSP” on page 128. Users can also be temporarily considered as Class 0 in case of a hot-shot or a lock-shot, as discussed in “Bias Factors” on page 131.
- Preferred (V=R and V=F) guests have dedicated storage in the V=R area and are not affected by this command.

SET SRM DSPBUF

This command imposes a limit on the number of users in the dispatch list. Other commands, such as SET SRM STORBUF and SET SRM LDUBUF, are more appropriate to control the dispatch list size because they are related to specific resources such as central storage and paging capacity. However, the SET SRM DSPBUF can be used when other controls are not effective.

The example shown in Figure 44 limits the dispatch list to 300 users at most, in such way that all three conditions below apply:

- The sum of Q1, Q2, and Q3 users does not exceed 300.
- The sum of Q2 and Q3 users does not exceed 180.
- The number of Q3 users does not exceed 80.

As with the STORBUF and LDUBUF commands, the DSPBUF command sets aside a buffer for Q1 and Q1+Q2 virtual machines in the dispatch list.

Because this command is not related to specific resources, such as CPU, storage, and paging, it can be used to generally reduce the dispatch list. Therefore, it can be used to reduce the data I/O activity (although very hard to control), which is a resource not directly controlled by VM/ESA.

Q0 users are not subject to this restriction to be promoted to the dispatch list.

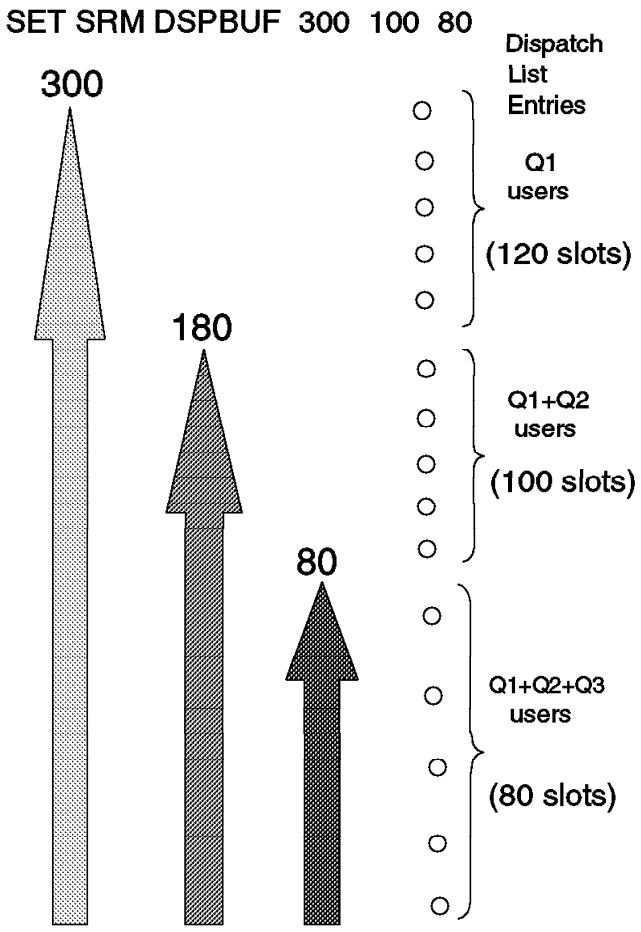

Figure 44. SET SRM DSPBUF Command

SET RESERVED

This is the basic central storage administration command to favor V=V virtual machines. The SET RESERVED command guarantees, most of the time, a minimum amount of central storage to a V=V virtual machine.

This minimum amount of storage should be assigned to a virtual machine to provide it with generally consistent paging activity and, therefore performance, because most of the active pages of the virtual machine tend to remain in central storage regardless of the other activity in the system.

Since VM/ESA tries to keep the most-frequently referenced pages in central storage, the SET RESERVED command should be used for server virtual machines that have long dormant periods but must react quickly when their users request services.

Not as drastic as the V=R option, this command is appropriate to server virtual machines, such as VTAM and PVM, that alternate dormant periods with bursts of activity. Their working sets are likely to have been paged out of the central storage when they become active. Bringing back their working sets would add delays to the interactive users that depend on the server.

This option is normally combined with the QUICKDSP option or with high shares specified through the SET SHARE command.

The number of reserved pages should be based on the average WSS the virtual machine had before the SET RESERVED command was issued. You may use the INDICATE USER command to find the average WSS, and then set the number of reserved pages to a number less than or equal to the WSS.

Note the following:

- Starting with VM/ESA Release 2.1, the maximum number of pages that you can reserve is limited by the number of pages in all private address spaces owned by the virtual machine.
- Many reserved pages reduce the central storage available to the other V=V virtual machines.
- Even reserved pages may be stolen by VM/ESA when central storage is over committed.

The QUERY RESERVED command can be used to display the number of page frames reserved through the SET RESERVED command. For each user, the number of reserved page frames and the actual number of pages resident in central storage is displayed; totals are also displayed, as shown in Figure 45.

```
query reserved

RSCS     RSV=000080 ACT=000036
VTAM     RSV=000100 ACT=000100
VMSERVU  RSV=000120 ACT=000000
REQUESTED FRAME TOTAL=000300; ACTUAL FRAME TOTAL=000136
```

Figure 45. QUERY RESERVED Command

SET QUICKDSP

The SET QUICKDSP command, as well as the OPTION QUICKDSP directory statement, tells VM/ESA to bypass the normal criteria for promotion from the eligible list to the dispatch list. For better results, you may want to combine the QUICKDSP option with the SET RESERVED command.

This means that a QUICKDSP virtual machine is moved to the dispatch list as soon as it leaves the idle state in the dormant list. The QUICKDSP virtual machine is classified as Q0 permanently. It may appear as E0 just for a very short period of time; a Q0 virtual machine never waits in the eligible list.

This option is usually given to preferred virtual machines (V=R or V=F) and to server virtual machines. Since preferred guests have their own reserved storage, you would not normally want them to wait for storage in the eligible list.

Do not give the QUICKDSP attribute to too many virtual machines because this can nullify the effectiveness of the SRM controls, such as STORBUF, LDUBUF, and DSPBUF.

SET SRM MAXWSS

The SET SRM MAXWSS command limits the maximum working set that any normal user may have; it is specified as a percentage of the dynamic paging area (DPA).

This command prevents certain virtual machines with very high working sets from dominating central storage. For example, you may notice that when a large FORTRAN program starts, the interactive users experience long response times. In this cases, the following command does not let any user occupy more than a percentage of the DPA:

```
SET  SRM  MAXWSS  50%
```

After the above command has been issued, VM/ESA takes the user out of the dispatch list whenever the WSS exceeds the specified percentage of the DPA. A Q1 user, is reclassified as Q2; a Q2 user is reclassified as Q3; and then moved to the eligible list.

It is likely that most of the user's working set has been paged out when he returns to the dispatch list. The user runs for a while and probably will increase his working set again, and once again he will be dropped from the dispatch list. The intent is to delay this user and protect the working sets of the other users.

Note the following:

- The MAXWSS limit works only if an eligible list exists.
- The MAXWSS limit works only if it is lower than the STORBUF percentage. A user that exceeds the STORBUF percentage is dropped from the dispatch list before a larger MAXWSS limit is reached.
- A user with an ABSOLUTE share larger than the MAXWSS percentage may exceed the MAXWSS limit. However, he may neither exceed the ABSOLUTE share nor the STORBUF percentage.
- The MAXWSS limit does not apply to Q0 users such as QUICKDSP users.

SET SRM XSTORE

You may use this command to tell the scheduler to over-commit the central storage when expanded storage is available. Suppose that you have 128 MB of expanded storage and want to tell the scheduler to handle 60 MB (47%) of expanded storage as though it were central storage. You could issue the following command:

```
SET  SRM  XSTORE  47
```

This command is an alternative to the use of the STORBUF technique to over-commit central storage, when expanded storage is available, as described in “SET SRM STORBUF” on page 122; the example above has the same effect as the STORBUF example given on page 123. However, the SET SRM STORBUF command provides better control over the different classes of users.

RETAIN XSTORE

The RETAIN XSTORE command reserves part of the expanded storage for exclusive use of CP (paging and MDC).

When VM/ESA depends on the fast paging capability provided by expanded storage, you may use the RETAIN XSTORE command to preserve the required portion of the expanded storage for CP paging.

CP always uses the portion of expanded storage that is not dedicated to any guest. At log-on time, VM/ESA dedicates portions of expanded storage to virtual machines that have the XSTORE statement coded in their directory entries. Using the RETAIN XSTORE command prevents this automatic dedication of expanded storage from excessively reducing the required CP area.

The MDC parameter command may be used to control the range within which the arbiter can vary the size of the MDC. Systems with constant workload profiles tend not to require additional control. The arbiter proves to be efficient at determining the optimum size.

Systems in which the workload varies significantly between high I/O rates to minidisks and high rates of virtual storage accesses may need further control. In this case, you should plan partition the expanded storage among guests, MDC, and CP paging.

The CP RETAIN command allows you to set a minimum and maximum limit that the arbiter can vary the MDC size. Sometimes, the MDC area is almost completely flushed, followed by flushing of the paging area to compensate for the high I/O wait experienced by users. This has been found to cause high and inconsistent response times.

The CP RETAIN XSTORE MDC OFF command prevents CP from using expanded storage for MDC; the MDC size is set to 0 MB.

When the total paging rate is low and the DASD configuration available for paging is not a bottleneck, part of central storage (ES/4381, ES/9221, and ES/9121), may be configured as expanded storage to attend exclusively to the MDC requirements. In this case, using the CP RETAIN XSTORE MDC ALL command should be used to bypass all paging to expanded storage.

SET SRM DSPSLICE

The default dispatch time slice is based on processor speed. During IPL, CP executes a group of instructions and set the default dispatch time between a minimum of five and a maximum of 100 milliseconds. This command allows you to set the dispatch time slice from one to 100 milliseconds.

Changing the dispatch time slice changes the frequency of re-prioritizing; it is considered a fine tuning item and should be done very carefully. The consequences of changing the dispatch time slice to a shorter value are:

- SHARE allocations are more accurate
- System overhead increases

The reverse effects occur when the dispatch time slice is changed to a larger value.

LOCK and UNLOCK

The LOCK command locks specific virtual storage pages of a virtual machine into central storage, as shown in Figure 46. It is not an easy task to determine which pages should be locked. The documentation of some products includes guidelines about the best candidates for locking. Usually, virtual page 0 is a good candidate. Fairly heavily used pages that otherwise would be constantly paged in and out are also good candidates for locking.

```
lock vtam 0 0 map

VIRTPAGE    REALPAGE
00000000    012A6000
```

Figure 46. LOCK Command

Generally, it is preferable to use the SET RESERVED command rather than the LOCK command. The heavily used pages of a virtual machine will probably be included among the reserved page frames. Therefore, the LOCK command is usually not required.

Note the following:

- The LOCK command may also be used to lock pages from the system virtual address space and data spaces.
- The QUERY FRAMES command displays the number of frames currently locked by the LOCK command.
- Many locked or reserved pages reduce the central storage available to the other V=V virtual machines.

Use the UNLOCK command to release page frames locked by the LOCK command. These pages become part of the dynamic paging area (DPA).

In addition, the UNLOCK command can be used to release the frames occupied by the V=R area or the RIO370 area. When unlocked, these pages become part of the dynamic paging area. However, once released, these areas cannot be restored unless VM/ESA is IPLed.

Bias Factors

The VM/ESA scheduler uses various bias factors, some of which can be supplied by you, in determining virtual machine priority. Some of these bias are:

- Interactive bias
- Paging bias
- Hot-shot bias
- Lock-shot bias

The interactive bias can be modified using the SET SRM IABIAS command; its objective is to favor really short-running transactions that finish using few dispatch time slices.

The paging bias and hot-shot bias are automatically triggered by the scheduler. The objective of the paging bias is to give a priority boost when users have their pages stolen during a transaction. The objective of the hot-shot bias is to give a priority boost to users interacting with the terminal.

Interactive Bias

The interactive bias is established through the SET SRM IABIAS command. The command has two parameters: an intensity, which is a percent value; and a duration, which represents a number of dispatch time slices.

The dispatch list priority is modified by the IABIAS command in such way that during the number of dispatch time slices specified, the user gets a boost in performance proportional to the intensity, as shown in Figure 47.

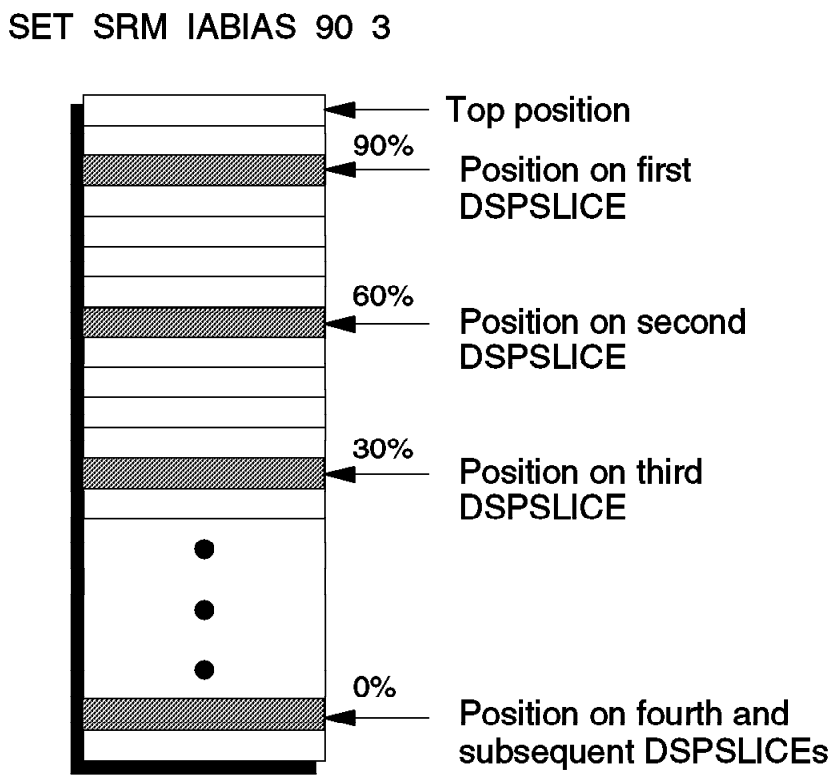

Figure 47. Interactive Bias and Dispatch List Priorities

The user receives performance boosts of 90%, 60%, and 30%, which corresponds to 3/3, 2/3, and 1/3 of the 90%, respectively. Therefore:

- During the first dispatch time slice, the user gets a dispatch list priority 90% better compared with the top priority user.
- During the second dispatch time slice, the user gets 2/3 of the initial boost (60%).
- During the third time slice, the user gets 1/3 of the initial boost (30%).
- During the fourth and subsequent dispatch time slices, the command has no effect on dispatch list priority.

The scheduler has a feedback mechanism that adjusts the dispatch time slices after the first one; any deficit or credit is carried over to the next dispatch time slice.

This automatic feedback mechanism may penalize users that received the boosts during the duration specified in the SET SRM IABIAS command, but were unable to finish the transaction within that number of dispatch time slices.

If the transaction is long enough, the initial boosts may be completely neutralized. Therefore, this command is effective only for very short transactions.

Paging Bias

Paging bias is a priority boost given when a user is being added to the dispatch list, but is continuing a previous transaction. Additionally, the user must have had referenced pages stolen while waiting in the dormant or eligible lists.

Since a user will be I/O bound for a short period while bringing the working set back into central storage, the boost is given in order to get the paging operations starting as soon as possible.

The boost is similar to the interactive bias boost, except that it is a one-shot boost (duration is 1) and the intensity is determined by the scheduler (90%).

Hot-shot Bias

Hot-shot bias is a special case where a transaction is progressing but the user interacts with the terminal. In this case, it gets one short dispatch time slice with very good priority.

The hot-shot dispatch time slice is shorter than normal (40% of the dispatch time slice) to minimize the effects of the boost, but long enough to allow the user to see a very good response time. Its objective is to quickly complete trivial transactions such as a #CP INDICATE LOAD command.

The boost is similar to the interactive bias boost, except that it is a one-shot boost (duration is 1) and the intensity is determined by the scheduler (95%).

Lock-shot Bias

Lock-shot virtual machines hold certain CP locks. They are placed immediately in the dispatch list (class 0) and remain there until the locks are released. This mechanism prevents these CP locks being held while virtual machines wait for resources in the eligible list.

WSS Fit Logic

This section describes the particular case of the STORBUF command and the working set size fit in the Dynamic Paging Area (DPA).

The combination of the settings established by commands, such as SHARE, STORBUF, LDUBUF, and DSPBUF, determines whether the user is promoted from the eligible list to the dispatch list.

The logic for working set size fit in the eligible list selection can be described by the pseudo-code in Figure 48. You can use a similar reasoning for the LDUBUF and DSPBUF settings.

```
IF user is E0 OR WSS = 0 (V=R/F)
THEN SELECT user
ELSE
  IF (STORBUF1/100*DPA) - (sum of ALL D-list WSS's) >= user WSS
  THEN
    IF user is E2 OR E3
    THEN
      IF (STORBUF2/100*DPA) - (sum of Q2 & Q3 WSS's) >= user WSS
      THEN
        IF user is E3
        THEN
          IF (STORBUF3/100*DPA) - (sum of Q3 WSS's) >= user WSS
          THEN /* E3 passed 3 tests */ SELECT user
        ELSE   /* E2 passed 2 tests */ SELECT user
        ENDIF
      ENDIF
    ELSE /* E1 user passed 1 test */ SELECT user
    ENDIF
  ENDIF
ENDIF
```

Figure 48. Eligible List Selection Pseudo-code

The pseudo-code described in Figure 48 can be translated in the following statements:

1. Class 0 users and preferred guests are not bounded by the STORBUF directives. Class 0 users are exempted by specific selection (QUICKDSP option); preferred guests are exempted because they do not use the DPA.
2. Class 1, 2, and 3 users must fit in considering the first STORBUF directive.
3. Class 2 and 3 users must additionally fit in considering the second STORBUF directive.
4. Class 3 users must additionally fit in considering the third STORBUF directive.

We are assuming that the users are not behind schedule; that is, the adjusted TOD at which the user is expected to be promoted to the dispatch list has not been reached yet.

When an E3 user is behind schedule and still does not fit in the DPA, class 3 is blocked in the eligible list and the scheduler does not look past him to select any other E3 users, except for those with WSS=0 (V=R and V=F).

When an E2 user is behind schedule and still does not fit in the DPA, class 2 is blocked in the eligible list and the scheduler does not look past him to select any other E2 or E3 users, except those with WSS=0.

When an E1 user is behind schedule and still does not fit in the DPA, class 1 is blocked in the eligible list and the scheduler does not look past him to select any other E1, E2, or E3 users, except those with WSS=0.

When an E1 user is quite far behind schedule, a Q2 or Q3 user is preempted in the dispatch list in a way that the E1 user can be added to the dispatch list; the preempted user is put at the top of the eligible list. This should be a very rare event that does not occur in well tuned systems.

In the example described in Figure 49, we have the following:

- The sum of the WSS of all users in the dispatch list is 31 MB.
- The first STORBUF directive is 200/100*20 = 40 MB. All users must fit in 40 MB and there is a hole of 9 MB.
- The second STORBUF directive is 150/100*20 = 30 MB. Q2 and Q3 users must also fit in 30 MB and there is a hole of 24 MB.
- The third STORBUF directive is 100/100*20 = 20 MB. Q3 users must also fit in 20 MB and there is a hole of 29 MB.

```
DPA    :  20 MB
STORBUF:  200% 150% 100%

Dispatch list

UserA  :  Q1,  WSS = 15 MB
UserB  :  Q1,  WSS = 10 MB
UserC  :  Q2,  WSS =  5 MB
UserD  :  Q3,  WSS =  1 MB

Eligible list in priority order

User1  :  E1,  WSS = 15 MB
User2  :  E2,  WSS =  4 MB
User3  :  E3,  WSS =  8 MB
User4  :  E3,  WSS =  5 MB
```

Figure 49. Example of WSS Fit in Central Storage

User1 has a WSS of 15 MB and does not fit in the 9 MB hole; therefore, User1 fails the first (and only) directive and is not promoted to the dispatch list.

User2 has a WSS of 4 MB WSS and it fits in the 9 MB and in the 24 MB holes; therefore, User2 is promoted to the dispatch list.

The first two holes are reduced to 5 MB and 20 MB, respectively; the third hole remains 29 MB.

User3 has a WSS of 8 MB; although it passes the second and third directives, it fails the first one and is not promoted to the dispatch list.

User4 has a WSS of 5 MB and passes all three directives; User4 is promoted to the dispatch list.

Querying Commands

The commands described in this section can be used as a first and quick tool to understand system behavior. However, a complete analysis of the system should use more sophisticated tools, such as VMPRF and RTM/ESA.

QUERY SRM

The QUERY SRM command displays the current scheduler parameters set by the SET SRM commands. Figure 50 shows the QUERY SRM command and its response.

```
query  srm  all

IABIAS : INTENSITY=90%; DURATION=2
LDUBUF : Q1=300% Q2=250% Q3=200%
STORBUF: Q1=300% Q2=250% Q3=200%
DSPBUF : Q1=32767 Q2=32767 Q3=32767
DISPATCHING MINOR TIMESLICE = 5 MS
MAXWSS : LIMIT=9999%
...... : PAGES=999999
XSTORE : 0%
```

Figure 50. QUERY SRM Command

QUERY FRAMES

This command displays the current allocation of central storage. Figure 51 shows the QUERY FRAMES command and its response.

```
query  frames

SYSGEN  REAL    USABLE  OFFLINE
016384  016384  016384  000000
V=R     RESNUC  PAGING  TRACE   RIO370
000000  000494  015884  000006  000000
AVAIL   PAGNUC  LOCKRS  LOCKCP  SAVE    FREE    LOCKRIO
015096  000557  000021  000000  000022  000188  000000
```

Figure 51. QUERY FRAMES Command

The response is given as three groups of numbers of pages. The first group refers to the real central storage resource:

SYSGEN — Storage available to VM/ESA; 64 MB in the above example (16384 times 4 KB).

REAL — This is the physically configured central storage.

USABLE — The lower of SYSGEN or REAL less the number of OFFLINE frames.

OFFLINE — The number of frames taken offline because of error.

The second group provides a breakdown of the usable frames, as follows:

V=R The V=R area reserved for preferred virtual machines.

RESNUC The size of the resident CP nucleus.

PAGING The dynamic paging area (DPA).

TRACE The internal CP trace table; consider changing the VM/ESA default, which is normally too large.

RIO370 Reserved for programs issuing DIAGNOSE X'98' instructions in 370 mode.

The third group provides a breakdown of the DPA (PAGING field):

AVAIL Available frames; that is, none of the following fields except LOCKRIO.

PAGNUC Pageable CP nucleus frames currently in central storage.

LOCKRS Frames currently locked for I/O operations.

LOCKCP Frames currently locked by the LOCK command.

SAVE Frames currently in use by the save area manager.

FREE Frames currently in use for free storage.

LOCKRIO The number of frames inside the RIO370 area currently in use by DIAGNOSE X'98'; it helps you adjust the size of the RIO370 area.

QUERY ALLOC

The QUERY ALLOC displays the number of cylinders or pages that are allocated, in use, and available for CP-owned DASD volumes.

If you use the ALL option, the information displayed is in number of cylinders; cylinders are flagged INUSE even if only one page is allocated on them. This is an old format still kept for compatibility with older releases of VM/ESA and VM/XA, as shown on Figure 52.

```
query alloc

DASD 0E72 ESAR21 3390 CKD-ECKD (UNITS IN CYLINDERS)
    TDISK TOTAL=000200 INUSE=000000 AVAIL=000200
    PAGE  TOTAL=000000 INUSE=000000 AVAIL=000000
    SPOOL TOTAL=000000 INUSE=000000 AVAIL=000000
    DRCT  TOTAL=000017 INUSE=000001 AVAIL=000016, ACTIVE
DASD 0E60 SPOOLV 3390 CKD-ECKD (UNITS IN CYLINDERS)
    TDISK TOTAL=000000 INUSE=000000 AVAIL=000000
    PAGE  TOTAL=000000 INUSE=000000 AVAIL=000000
    SPOOL TOTAL=002225 INUSE=000079 AVAIL=002146
    DRCT  TOTAL=000000 INUSE=000000 AVAIL=000000
DASD 0E62 PAGEVL 3390 CKD-ECKD (UNITS IN CYLINDERS)
    TDISK TOTAL=000000 INUSE=000000 AVAIL=000000
    PAGE  TOTAL=002225 INUSE=000178 AVAIL=002047
    SPOOL TOTAL=000000 INUSE=000000 AVAIL=000000
    DRCT  TOTAL=000000 INUSE=000000 AVAIL=000000
IPL NUCLEUS ACTIVE ON VOLUME ESAR21
```

Figure 52. QUERY ALLOC ALL Command

A better alternative is using the MAP option that provides information in page units, as shown in Figure 53. You can restrict the response by specifying that you want only PAGE, SPOOL, TDISK, or DRCT information; volume serial numbers specified in the command further restricts its action to the corresponding volumes.

```
query alloc map

          EXTENT EXTENT                    % ALLOCATION
VOLID  RDEV  START    END  TOTAL IN USE   HIGH USED TYPE
------ ---- ------ ------ ------ ------ ------ ---- -------------
ESAR21 0E72     1     17     17      1      1   5% DRCT ACTIVE
             101    300    200      0      0   0% TDISK
SPOOLV 0E60     1   2225 400500  38897  55440   9% SPOOL
PAGEVL 0E62     1   2225 400500  10126  63327   2% PAGE
```

Figure 53. QUERY ALLOC MAP Command

The main fields in the response of the command are:

EXTENT START — The starting or ending cylinder or page number of the extent. For CKD volumes, this number is in cylinders; for FBA volumes, this number is in pages.

EXTENT END — The ending of the extent; similar to EXTENT START.

TOTAL — The total number of pages in the extent.

IN USE — The number of pages in use in the extent.

HIGH — The highest page number in use in the extent. When HIGH equals TOTAL, there was at least one occasion on that page extent when page allocation was forced to wrap around due to end-of-extent before 70% of the preceding page slots become free (the normal criterion). This means that paging space limitations are starting to impair performance.

USED — The percentage of pages used in the extent; make sure you have enough paging areas (in different volumes) to keep this percentage around 30%.

If you use this option of the QUERY ALLOC command through DIAGNOSE X'08' requesting the response in a buffer, complete volume information (volid and rdev) is returned for each extent displayed on the volume. This provides you with easier parsing of the command output.

QUERY XSTORE

The QUERY XSTORE command displays the current assignment of the expanded storage to CP paging, MDC, and guests, as shown in Figure 54. The QUERY XSTORE MAP command provides a shorter description of expanded storage allocation.

Check particularly that enough expanded storage is available for CP paging (appears as userid=SYSTEM in the response).

The MDC function competes against CP paging, but the RETAIN XSTORE command can impose a maximum limit on the MDC function.

```
query xstore

XSTORE= 64M online= 64M
XSTORE= 54M userid= SYSTEM usage= 96% retained= 0M pending= 0M
XSTORE MDC min=32M, max=54M (10M pending), usage=79%
XSTORE= 10M userid= MVS
XSTORE= 54M userid=  (none)  max. attach= 54M
```

Figure 54. QUERY XSTORE Command

The command in Figure 54 provides the following information:

- 64 MB of expanded storage is available and online.
- 54 MB is usable by CP, but it is not dedicated (RETAIN command). The operator is able to attach this 54 MB to another virtual machine, in which case CP has to flush it before attaching it to the guest. Of this 54 MB, 96% is currently in use.
- MDC has a minimum of 32 MB and a maximum of 64 MB to allocate. However, 10 MB is not available at the moment, and will be made available only when the 10 MB dedicated to MVS is detached. MDC is using 79% of expanded storage usable by CP.
- One line is displayed for each guest that has expanded storage dedicated.
- The last line indicates that 54 MB is available to be attached to a guest or retained by CP; in the meantime, CP is using it.

QUERY NSS

Use the QUERY NSS command to check that saved segments are being used correctly. Figure 55 is an example of two variations of this command.

The first command provides complete information about the segment, such as pages saved, load address, and number of users attaching the segment.

Make sure you do not have many of class-P (pending purge) segments because they waste storage and also because they probably are obsolete; each user should purge the segment or IPL CMS again.

The second command shown in Figure 55 can be used, for example, to determine who is using a pending-purge segment.

```
query nss name cmsfiles map

FILE FILENAME FILETYPE MINSIZE  BEGPAG ENDPAG TYPE CL #USERS PARMREGS VMGROUP
0032 CMSFILES DCSS      N/A    01450  015FF   SR  A  00004   N/A      N/A

query nss users cmsfiles

FILE FILENAME FILETYPE CLASS
0032 CMSFILES DCSS     A

   VMSERVU  VMSERVR  VMSERVS  VMSERVT
```

Figure 55. CP QUERY NSS Command

INDICATE SPACES

This command displays information about address spaces that you own (class G) or belonging to any other user (class E), as shown in Figure 56.

```
indicate spaces

Spaceid=CMS212:BASE  Owned size=11M  PRIVATE
Pages:  Main=21  Xstore=1  Dasd=104  Locked=0
Private paging:
  Xstore:  Reads=1269      Writes=1809      Migrates=526
  Dasd:    Reads= 688      Writes= 635
Shared paging:
  Xstore:  Reads=   0      Writes=   0      Migrates=  0
  Dasd:    Reads=   0      Writes=   0
```

Figure 56. INDICATE SPACES Command

The command response indicates where the pages of each address space is allocated and the paging activity in expanded storage and DASD.

Different counts are maintained for private and shared paging; these counts are cumulative for the life of the address space and do not change when the state (private or shared) of the address space changes.

Note that a data space becomes shared when the owner grants access to another user (ADDRSPACE PERMIT macro); the data space remains shared until it is isolated or destroyed.

INDICATE LOAD

Use the INDICATE LOAD command to display the current load on the system, as shown in Figure 57. The INDICATE LOAD command shows smoothed values which are slow changing averages of the last minutes rather than instantaneous indicators. The smoothed values result from a weighted average of the last observations.

```
indicate load

AVGPROC-071% 02 AVGVEC-000% 02 VECTOR-00000
XSTORE-002181/SEC MIGRATE-0198/SEC
MDC READS-000216/SEC WRITES-000160/SEC HIT RATIO-095%
STORAGE-086% PAGING-0704/SEC STEAL-025%
Q0-00001(00000)                       DORMANT-00019
Q1-00002(00000)          E1-00000(00000)
Q2-00001(00000) EXPAN-001 E2-00000(00000)
Q3-00003(00000) EXPAN-001 E3-00000(00000)

PROC 0000-074% VEC-000% PROC 0001-069% VEC-000%

LIMITED-00001
```

Figure 57. INDICATE LOAD Command (Class E Response)

Most of the information shown by this command affects storage. The command should be interpreted as follows:

XSTORE This field shows the page activity (2181 pages/second) from central storage to expanded storage and from expanded to central storage.

MIGRATE Indicates the number of pages per second (198) migrating from expanded storage to DASD.

MDC READS Identifies the number of pages per second (216) moved from expanded storage (MDC) to central storage.

WRITES Identifies the number of pages per second (160) moved from central storage to expanded storage for MDC.

HIT RATIO Indicates the percentage (95%) of successful lookups in the minidisk cache. This is the percentage of eligible read I/Os that were avoided because of the MDC.

STORAGE An indication of the commitment of central storage. It is the sum of the WSSs of the users in the dispatch list divided by the size of the DPA. This is not a good indication for storage contention, because this field does not account for other loads on storage, such as CP modules, buffers, free storage, trace table entries, and pages of users in the dormant and eligible lists.

PAGING Indicates the number of pages per second flowing between central storage and auxiliary DASD; it does not include paging to the expanded storage and it does not include spool information. Pages per second is not the same as I/O operations per second because CP uses block paging most of the time.

STEAL A percentage of the page read requests from both expanded storage and DASD. It is the percentage of the page reads that caused frames to be stolen from users in the dispatch list or from pages reserved by the SET RESERVED command because the available list was empty. There is no direct correlation between STEAL and response times, because the paging waiting time may vary a lot.

Qn The number of users of each class in the dispatch list. The number in parenthesis indicates how many are loading users. The number of users in the dispatch list is the multiprogramming level. It affects the storage utilization and the paging rate.

En The number of users of each class in the eligible list. The number in parenthesis indicates how many are loading users. Long eligible lists indicate central storage contention. The users in the eligible lists experience longer response times. The interactive users, at least, should not stay long in the eligible list.

EXPAN An indication of the expansion of the corresponding response times due to contention for the resources required to run the users in Q2 and Q3.

DORMANT The idle users; their pages in central storage are likely to be used by the active virtual machines.

INDICATE USER

This command displays the resources currently used by a virtual machine, including the special user SYSTEM. It displays central-storage related information, such as the working set size, as shown in Figure 58.

```
indicate user fconx

USERID=FCONX    MACH=XA  STOR=0032M VIRT=V XSTORE=NONE
IPLSYS=CMS     DEVNUM=00010
PAGES: RES=000029 WS=000126 LOCK=000000 RESVD=000000
NPREF=000350 PREF=000000 READS=019577 WRITES=018505
XSTORE=000154 READS=049121 WRITES=061274 MIGRATES=011440
CPU 00: CTIME=05:32 VTIME=000:03 TTIME=000:07 IO=003777
       RDR=011592 PRT=000000 PCH=011612
       VVECTIME=000:00 TVECTIME=000:00
```

Figure 58. INDICATE USER Command

The central-storage related fields are interpreted as follows:

STOR The defined virtual storage size.

VIRT=V This can be VIRT=V, VIRT=R or VIRT=F.

XSTORE The expanded storage attached to the user. (The second XSTORE field is described below).

RES The number of pages currently resident in central storage. This value usually fluctuates around the WSS value, but it can be significantly less than the working set depending on the central storage activity.

WSS The most recent estimate of the working set size.

LOCK The number of pages currently locked in central storage by the LOCK command.

RESVD The number of pages reserved by the SET RESERVED command.

NPREF The number of pages currently allocated in auxiliary DASD storage. (NPREF and PREF are names used in previous VM operating systems retained for compatibility).

READS The number of page reads from DASD since logon.

WRITES The number of page writes to DASD since logon.

XSTORE The number of page blocks currently allocated in expanded storage for user pages.

READS The number of page reads from expanded storage since logon.

WRITES The number of page writes to expanded storage since logon.

MIGRATES The number of pages transferred from expanded storage to DASD since logon.

There is a variation of this command (with the EXPANDED option) that provides additional information about all address spaces owned by the user with the corresponding paging information related to private and shared pages.

INDICATE QUEUES

The INDICATE QUEUES command displays the users that are currently in the dispatch list followed by the users in the eligible list. Each list is displayed in priority order.

Besides showing the existence of queues, this command also shows which resources the users are waiting for. It can be used to check whether too many users are in paging wait status, for example. The repeated use of this command helps to identify the resources that cause most contention in the system. The main system resources tracked by this command are the CPU, the central storage, the paging capacity, and the data I/O capacity.

The EXP operand requests expanded information. The INDICATE QUEUES EXP command can be used to identify which users are loading users, for instance, as shown in Figure 59.

```
indicate queues exp

 (1)    (2) (3) (4)   (5)   (6)   (7)    (8)  (9)

RSCS         Q0 PS  000521/000516 .... -.3972 A01
BSPCAL        Q1 IO  000315/000289 .I.. -.3358 A01
JCSVRT        Q1 IO  000648/000608 .I.. -.3324 A01
CMS12        Q1 R00 000000/000003 .I.. -.1155 A00
TCPMAINT      Q3 IO  000641/000636 ...L  .1053 A01
SFSLD        Q3 R01 000026/000003 ..D.  .2018 A01
SPOOLLD       Q2 R   000062/000056 ..D.  .2054 A00
PRIVLD        Q1 R   000017/000017 ..D.  .2067 A00
CMS10        Q3 R   013890/013880 ....  .2095 A01
CMS15        Q2 IO  000304/000272 ..D.  .4041 A02
LOOPLD       Q3 TI  000001/000003 ..D.  99999 A00
VMSERVU       Q0 PS  000084/000091 ....  99999 A00
NETVIEW       Q1 PS  000983/000961 .I..  99999 A00
CENTROAT      Q3 PS  000119/000441 ....  99999 A00
```

Figure 59. INDICATE QUEUES Command

The response to the above command displays one line for each user in the dispatch and eligible lists. The line with the numbers between parentheses in Figure 59 is not part of the command response; these numbers correspond to the items in the numbered list below:

1. The userid of the virtual machine.
2. The virtual processor for multiprocessing virtual machines. The format is *MPpp*, where *pp* is the virtual CPU number.

   This field is blank for uniprocessing virtual machines and is not shown in Figure 59.

   Each virtual processor is displayed on a separate line. Virtual processors are separately dispatchable by VM/ESA and are represented by their own VMDBK control block.
3. The current list and class of the user. This field can be Q0, Q1, Q2, Q3, L0, L1, L2, L3, E1, E2, or E3. VM/ESA can briefly classify a user as E0, but this command reports that user as Q0.

4. The user status; it can be one of the following indicators:

   Rcc The user is running on real processor *cc*.

   R The user is ready to be dispatched. The user holds all needed resources except processor time. The user is in a wait status because higher priority users are using all the real processors.

   PG The user is in page wait. A page-in operation from auxiliary storage to central storage must complete before the user can continue.

   IO The user is in I/O wait. The user is waiting for an I/O operation to start or for a synchronous DIAGNOSE I/O operation to complete.

   PS The user is in enabled PSW wait state. The most common cause of this state is that the user is waiting for an asynchronous I/O operation to complete.

   EX The user is in instruction simulation wait.

   AP The user is waiting for an APPC/VM function.

   TI Test idle status; if the user remains idle for more than 300 milliseconds, he is moved to the dormant list.

   - Other status.

5. The number of pages resident in central storage.
6. The last estimated working set size (WSS) in pages.
7. Additional status; the current indicators are:

   H... The user is receiving a hot-shot bias.

   L... The user is receiving a lock-shot bias.

   .I.. The user is receiving an interactive bias.

   ..D. The user is currently past its eligible list deadline.

   ..M. The user was moved to the eligible list for exceeding the maximum allowed WSS (MAXWSS).

   ..P. The user returned to the eligible list temporarily because of preemption.

   ..G. The user returned to the eligible list temporarily for exceeding the WSS-growth limit.

   ..L. The user is in the eligible list after having been in the dispatch list for lock-shot.

   ...L The user is a loading user; he is expected to require high paging activity.

8. The user's priority is indicated as the remaining number of seconds to reach the deadline priority.

   The deadline priority is the time that the user is scheduled to finish his next time slice if he is on the dispatch list, or the time he is scheduled to be promoted to the dispatch list is he is on the eligible list. A negative number of seconds indicates that the user has already passed the deadline.

9. *App* indicates the processor affinity; the user is preferred to run on processor *pp*, but is not necessarily run on that processor.

INDICATE PAGING

Use the first command in the example below to display those users in page wait; use the second command to display all users. In both cases you will see the number of pages each user currently has in auxiliary expanded storage and in auxiliary DASD storage:

```
INDICATE  PAGING  WAIT
INDICATE  PAGING  ALL
```

To determine whether your paging subsystem needs tuning, check the paging with the CP INDICATE LOAD and CP INDICATE PAGING commands.

If the paging rate is low and few users are in a page wait, you do not have a paging problem. If many users are in page wait, the page rate itself can still be low because of configuration inadequacies, such as few page volumes. If the paging rate is high, you may have to determine whether you have enough paging devices; you need other performance tools, such as VMPRF and RTM/ESA.

Guidelines

Table 13 presents general guidelines for storage administration with a brief explanation and the corresponding page number where it is discussed.

Table 13 is organized as follows:

Hint A general suggestion about storage usage or performance

Comment A short comment describing the hint

Page The page number where the subject is discussed

Table 13 (Page 1 of 2). Storage Administration Hints

Hint	Comment	Page
No paging is better than fast paging	Balance central storage with expanded storage	7
Avoid duplicate functions	Reduce double paging and CCW translation	20
Dedicate volumes to paging	Nonpaging activity in these volumes should be low	21
Allocate enough DASD for paging	For efficient block paging	21
Allocate only one contiguous PAGE area per volume	Decrease average seek length	22
Avoid mixing device types for paging	Faster devices will probably get filled first	22
Allocate separate volumes for SPOL areas	If there is a lot of spooling activities	22
Balance V=R area and DPA	Start with CP and preferred machines having similar paging rates	27
RIO370 versus large V=R area choice	RIO370 reduces the V=R area to less than 16 MB	27
Move CP modules to resident nucleus	Highly used pageable CP nucleus modules may be made resident	28

Table 13 (Page 2 of 2). Storage Administration Hints

Hint	Comment	Page
Allocate only one contiguous DRCT area	Otherwise VM/ESA issues I/O operations for directory references	30
Reduce trace table size	VM/ESA default is too high	31
Use Shared Segments	Many program products provide this installation option	31
Check system MDC effectiveness	Optimize expanded storage usage	100
Check volume MDC effectiveness	Disable MDC for some minidisks	100
No I/O is better than fast I/O	Minimize real I/O operations	101
Trade normal I/O operations for paging	Use VM data spaces, MDC, and virtual disks	113
Disable 3990 cache for page volumes	Paging is not good candidate for caching	54
Disable 3990 cache for volumes with low cache hit ratio	Response time can be even worse when caching everything	54
Let 9340 cache choose data for caching	Adaptive cache algorithm works quite well	62
Efficient DASD layout	Optimize layout to have short average seeks	68
Reduce eligible lists	Tune VM/ESA or add central storage	115
Avoid class 1 users in the eligible list	Use the SET SRM commands	118
Make only one modification at a time	The tuning commands interact with each other	119
Do not favor many users	The normal scheduler mechanisms may not work properly	119
Use relative shares instead of absolute shares	Use SET SHARE ABSOLUTE carefully	121
Set high RELATIVE shares to servers	Example: VTAM, SQLDBA, SFS servers	121
Over commit central storage with expanded storage	Start considering expanded storage as central storage	123
SET SRM LDUBUF 300% 200% 100%	The VM/ESA defaults are too low	124
SET RESERVED pages for servers	Avoid interactive users delays	127
Avoid the LOCK command	Prefer the SET RESERVED command	130
Use INDICATE QUEUES	To identify scarce resources	142

Table 14 presents rules-of-thumb for the storage resources, with the following columns:

1. Resource to be observed.
2. Name of the corresponding variable in the performance reports.
3. Normal values that the variables may assume.
4. Actions values that should require your attention.
5. Report identification where the variables are found.
6. Additional information

A short description of the main variables is presented after Table 14.

Table 14. Rules-Of-Thumb for Storage and DASD I/O

Resources to be Observed	VAR	Normal Values	Action Values	RTM VMPRF	Comments
Pct of users in page wait	%PGW	< 10	> 10	SRC	Interactive users suffer more from page wait.
Number of tasks waiting for page frames	WPG	0	> 1	SYSTEM	(WPG¸ACT)*100: Percent of active users waiting for paging. See also SCN_PAS2; if low, then paging is the only problem.
Percent of pages slots in use	INUSE	< 30	> 30	PRF088	Block page sizes depend on finding enough contiguous page blocks: check block size and only increase space if block size on reads falls well below 10.
Average XSTORE block age	XAGE	-	> 10 s	SYSTEM	Pages should be in expanded storage when users after think time.
Average XSTORE block age compared to maximum age of XSTORE block.	MAGE ¸ XAGE	> 2	< 2	SYSTEM	If < 2, XSTORE is used only as a buffer for pages written to DASD, pages should stay longer to be used when users wake up again to avoid paging from DASD.
Number of times global cyclic list was searched by migration task	MIG_SRCH	< XST_MIGS	> XST_MIGS	SYSTEM	Indicates demand for XSTORE; migrate task may be having a difficult time keeping up with demand for XSTORE.
Number of times XSTORE migration buffer threshold was increased	MIG_THRH	low	high	SYSTEM	Insufficient DASD paging band width; few page volumes or paths to them.
Number of times no XSTORE was available to allocate	MIG_NOBK	near 0	> 0	SYSTEM	Insufficient DASD paging band width; migration task is not being able to migrate pages to DASD fast enough to free up XSTORE for other requests.
MDC used by active users	MDAU ¸ ACT	> 1	<< 1	SYSTEM	Few active users used MDC during the last minute. Check if minidisks are not disabled for MDC.
MDC hit ratio	MDHR	> 0.80	< 0.80	SYSTEM	If <80%, cache is XSTORE constrained; consider more XSTORE for MDC.
MDC read to write ratio Read percentage	MDRM PCT_HIT	> 6 > 85	< 4 < 80	SYSTEM PRF020	Disable MDC for minidisks with heavy write activity.
Eligible lists	E*n*CT ELIST	0	>0	SCLOG PRF002	Central storage may not be enough for the current load.
Heavy loading users	PAG STR			SRC	Lower the SHARE if a user is consuming more resources than you would like.
Function pending time	FPT PEND	0.3	> 1	DEVICE PRF012	Indicate excessive queueing on the channel subsystem. Add additional paths to device.
Response time ¸ service time	RESP ¸ ACC	1.3	> 1.5	PRF012	Excessive queue time
Service time	ACC SERV	< 15	> 20	DEVICE PRF012	Average service time for 3390s.
Disconnected time	DCT DISC	< 15	> 15	DEVICE PRF012	May indicate RPS misses: balance channels or add more paths to device. May indicate long seeks: move active areas closer or out of disk.
I/O wait percentage	%IOW	< 3	> 10	SCR	Interactive users suffer more from I/O wait.
Device utilization	%UT PCT_BUSY	< 30	> 35	DEVICE PRF012	Average transaction time may be impacted.
Percentage of deferred lock requests	%LK	0	> 3	DEVICE	A nonzero value indicates that devices are being shared among real processors.
Channel utilization	BUSY	< 25	< 55	PRF013	1 path: < 25% 2 paths: < 40% 3 paths: < 50% 4 paths: < 60%

The RTM variables are described as follows; see *RTM VM/ESA Program Description/Operations Manual* for more details.

(WPG/ACT)*100

If there are many active users waiting for paging, look at variable SCN-PAS2, which indicates the number of times the scan for available pages completed after the second pass. If this variable is low, it means that paging response time may be responsible for the page delays.

MAGE,XAGE

MAGE is the average age of an XSTORE block before being migrated. XAGE is the average age of an XSTORE block.

MIG_SRCH

The migrate task searches through the cyclic list (VMDBKs) in order to find XSTORE blocks to migrate out and thus free XSTORE. It chooses blocks to migrate based on the migrate target time (initially set equal to the XSTORE average age). If half the cyclic list is searched without satisfying the blocks needed to migrate, then the target time is lowered and an additional search is made through the cyclic list. If this number is greater than the migrate routine invocations (XST_MIGS), it may mean that CP gets enough blocks to migrate but the paging subsystem is not fast enough to move the pages to DASD.

MIG_THRH

The migration threshold buffer is the number of buffers needed for block paging migration blocks. Sustained high values indicate insufficient DASD paging band width, such as few devices and few paths to them, or insufficient XSTORE. Look at sum of xstore page-ins (XST_PGIF + XST_PGIS); if it is about the same as the number of pages migrated to DASD (MIG_TOT), the page-ins from XSTORE are used only to move pages out to DASD. In other words, few page-ins are being used to resolve virtual machine page faults, which means that XSTORE is not big enough for this load. Otherwise, the buffer is being increased because the page subsystem does not respond to the number of pages being migrated; in other words, there is insufficient page band width.

MIG_NOBK

The number of times there is no XSTORE blocks to allocate should be close to zero. The migration function cannot keep up with the user demand on XSTORE. The same comments of the previous variable apply here.

MDAU,ACT

MDAU is the number of users that did at least one insert into the cache during the last minute. If this number is low compared to the number of active users, MDC may be inactive due to some inhibit condition, such as minidisks not 4-KB formatted or real disks defined as shared.

:PAG and :STR

:PAG is the percentage of paging resources consumed by a user relative to all other users. :STR is the percentage of storage resources consumed by a user relative to all other users.

Part 2. Probability and Queueing Theory

This part contains advanced material intended to provide a solid foundation for a better understanding of system performance. Various elementary queueing models are discussed with emphasis on their performance aspects.

Although this part contains detailed mathematical descriptions of the models, most of the time you can follow and understand the discussion if you are familiar with high-school algebra. On the few occasions we use more advanced mathematics, we never go beyond very elementary calculus.

This part consists of the following chapters:

- Chapter 8: Probability Theory Refresher

  Describes the main definitions and concepts of probability theory needed for the study of queueing theory models, including several examples. The sum of terms of the most common series (arithmetic, geometric, and power of natural numbers) is also discussed. It also presents two complete examples:

  - DASD seek model

    Calculates the average seek length and the average seek time of a DASD I/O operation in a random seek pattern.

  - RPS-miss time

    Calculates the RPS-miss time component of DASD I/O service time as a function of effective path utilization.

- Chapter 9: Introduction to Queueing Theory

  Describes a general methodology for analyzing elementary queueing models (birth-death process) and some queueing models (M/M/1, M/M/c, M/M/1/K, M/M/1//M, and M/M/c//M).

  The main parameters, service time, queue time, response time, and the number of elements in service, queue, and system are derived for each model.

  Also, in each case examples help you understand the main concepts under discussion.

- Chapter 10: Simulation

  Describes simulation as a tool to analyze queueing processes. A complete example (REXX program) that simulates an M/M/1 system is discussed.

  A REXX program rather than a simulation tool, such as RESQ (Research Queueing Package), is used to explain the underlying processes related to the simulation technique.

Chapter 8. Probability Theory Refresher

This chapter reviews the basic concepts of the probability theory. It describes its main concepts, definitions, and random variables. Several examples are discussed to help you understand the concepts. Although not part of probability theory, we also discuss the most important series (arithmetic, geometric, and power of natural numbers) due to their importance in queueing theory.

Concepts and Definitions

Consider a single experiment of tossing an unbiased coin. This experiment has two mutually exclusive outcomes, namely heads and tails. The various factors influencing the outcome of the experiment are too numerous to take into account, at least if the coin tossing is fair; therefore, the outcome of the experiment is said to be random.

You would certainly say that the probability of getting heads and the probability of getting tails both equal 1/2. Your answer is based on the intuitive idea that the two outcomes are equally likely, or equiprobable, because of the very nature of the experiment. Almost anyone would agree with you, but few would actually understand what they mean by probability.

Let us consider an experiment with a finite number of outcomes that are equally equiprobable, and let A denote some event associated with a possible outcome of that experiment. The probability of event A is defined as the fraction of the outcomes in which A occurs:

$$P[A] = \frac{N(A)}{N} \qquad (17)$$

where N is the total number of outcomes of the experiment and N(A) is the number of outcomes leading to the occurrence of the event A.

Despite its seeming simplicity, equation (17) can lead to nontrivial calculations. In fact, before using equation (17) in any problem, you must find all equiprobable outcomes, and then identify all those leading to the occurrence of the event A in question; in most cases, you have to use combinatorial analysis.

The accumulated experience of innumerable observations reveals a remarkable regularity of behavior, allowing us to assign a precise meaning to the concept of probability not only in the case of experiments with equiprobable outcomes, but also in the most general case. Suppose the experiment under consideration can be repeated any number of times, say *n* times, and let *n(A)* be the number of experiments in which the event A occurs; the ratio n(A)/n is called the relative frequency of the event A in that experiment. It turns out that the relative frequency observed in different series of experiments are virtually the same for large *n*, clustering about some constant:

$$P[A] \quad \frac{n(A)}{n} \qquad (18)$$

called the probability of the event A. More exactly, equation (18) means that:

$$P[A] = \lim_{n \to \infty} \frac{n(A)}{n} \qquad (19)$$

Roughly speaking, the probability of an event A equals the fraction of experiments leading to the occurrences of A in a large series of trials.

The main definitions and properties related to the probability theory are:

- The probability of an event A, $P[A]$ is a number between zero and one:

  $$0 \leq P[A] \leq 1$$

- The set that contains all possible events of an experiment is called the sample space S for that experiment; therefore:

  $$P[S] = 1$$

- A and B are called complementary events when:

  $$P[A] = 1 - P[B] = P[\overline{B}]$$

- The conditional probability of event B given A, denoted by $P[B|A]$, is the reevaluation of the probability of B in the light of the information that A has already occurred. Thus, A becomes our new sample space and we are concerned only in that part of B that occurs with A; that is, the intersection of A and B, denoted by $A \cap B$:

  $$P[B|A] = P\frac{[A \cap B]}{P[A]} \qquad (20)$$

- As a consequence of definition (20), the probability of the intersection of two events A and B can be written as:

  $$P[A \cap B] = P[A] \cdot P[B|A] \qquad (21)$$

  If the occurrence of A does not affect the probability of B, we say that A and B are independent events; in this case, $P(B|A) = P(B)$, and we can write:

  $$P[A \cap B] = P[A] \cdot P[B] \qquad (22)$$

- If A and B are independent events, A and $\overline{B}$; $\overline{A}$ and B; and $\overline{A}$ and $\overline{B}$; are also independent events.

- The probability of the union of two events A and B is the probability that at least one the events A and B occurs; it can be calculated by:

  $$P[A \cup B] = P[A] + P[B] - P[A \cap B] \qquad (23)$$

- If A and B are independent events, equation (22) allows us to write:

  $$P[A \cup B] = P[A] + P[B] - P[A]\,P[B] \qquad (24)$$

- A and B are said to be mutually exclusive events when the occurrence of one event excludes the occurrence of the other; in this case, equation (24) can re-written as:

  $$P[A \cup B] = P[A] + P[B] \qquad (25)$$

- The union of two events is the complement of the intersection of the complementary events:

$$P[A \cup B] = 1 - P[\overline{A} \cap \overline{B}]$$

- In calculating the probability of an event E, it is often convenient to use conditional probabilities as an intermediate step. Suppose $A_1, A_2, \ldots A_k$ is a full set of mutually exclusive events; that is, one (and only one) of the events A_k always occur, as shown in Figure 60. This can be translated by:

$$P[A_1 \cup A_2 \cup \cdots \cup A_k] = 1$$

As you can see from Figure 60, any event E in the same sample space always intersects one or more A_k events. In the example, event E intersects events A_2, A_3, A_4, A_6, A_7, A_8, A_{10}, and A_{11}.

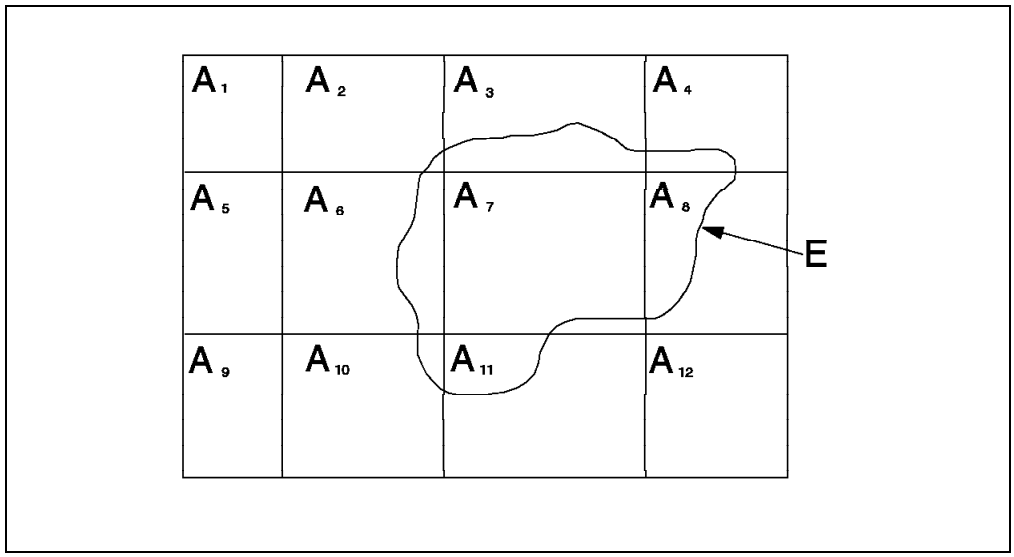

Figure 60. Total Probability Theorem

Since the intersections of event E with the events A_1, A_5, A_9, and A_{12} are empty, you can consider event E as the union of the intersections of event E with each of A_k events:

$$E = \{(E \cap A_1) \cup (E \cap A_2) \cup \cdots \cup (E \cap A_k)\}$$

Since events A_k are mutually exclusive, a generalization of equation (25) allows us to write:

$$P[E] = \sum_{n=1}^{k} P[E \cap A_k] \qquad (26)$$

Using equation (21), we can re-write equation (26) as:

$$P[E] = \sum_{n=1}^{k} P[A_k]\, P[E \mid A_k] \qquad (27)$$

Equation (27) is known as the total probability theorem.

Examples

Assume you choose one number *n* at random from the set of integers 0-9:

1. What are the probabilities related to the events A ° [n > 2] and B ° [n = even] ?

   Using Figure 61 as a reference (known as a Venn diagram):

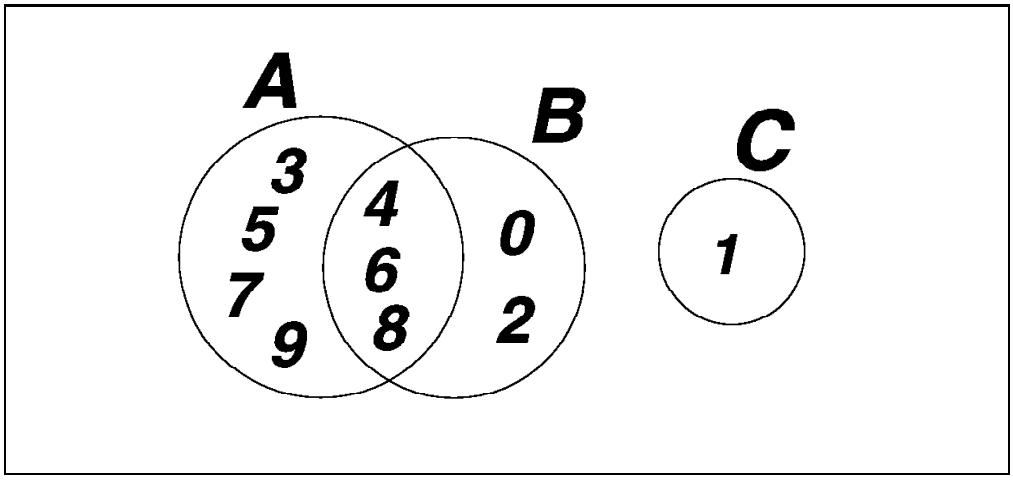

Figure 61. Event A (n > 2) and Event B (n = even)

$$P[A] \circ P[n > 2] = \frac{7}{10}$$

$$P[B] \circ P[n = \text{even}] = \frac{5}{10}$$

$$P[A \cap B] = \frac{3}{10}$$

$$P[A|B] = \frac{P[A \cap B]}{P[B]} = \frac{3}{10} \cdot \frac{10}{5} = \frac{3}{5}$$

$$P[B|A] = \frac{P[A \cap B]}{P[A]} = \frac{3}{10} \cdot \frac{10}{7} = \frac{3}{7}$$

Events A and B are not independent because:

$$P[A] \neq P[A|B]$$

$$P[B] \neq P[B|A]$$

When you know that the number is greater than two, you are excluding two even numbers (0 and 2) and just one odd number (1). Therefore, the occurrence of event A (n > 2) gives you valuable information about the occurrence of event B (n=even), and its probability has to be reevaluated.

The conditional probabilities are obtained by a straight application of the definition, but they can also be obtained by the diagram, considering the number of elements in the intersection as successes, and the number of elements of the event that occurred as the new sample space.

2. What are the probabilities related to the events $A \circ [n > 3]$ and $B \circ [n = \text{even}]$?

   Using Figure 62 as a reference:

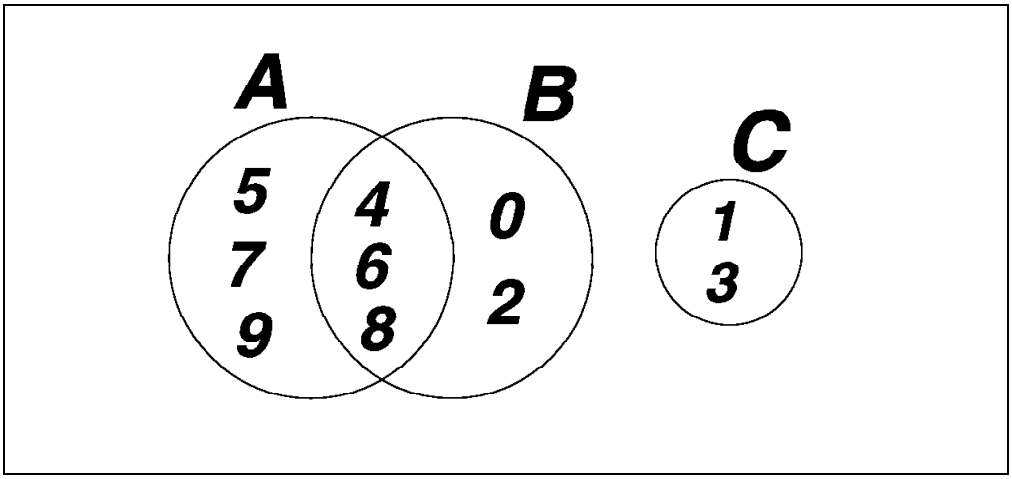

Figure 62. Event A (n > 3) and Event B (n = even)

$$P[A] \circ P[n > 3] = \frac{6}{10} = \frac{3}{5}$$

$$P[B] \circ P[n = \text{even}] = \frac{5}{10} = \frac{1}{2}$$

$$P[A \cap B] = \frac{3}{10}$$

$$P[A|B] = \frac{P[A \cap B]}{P[B]} = \frac{3}{10} \cdot \frac{10}{5} = \frac{3}{5}$$

$$P[B|A] = \frac{P[A \cap B]}{P[A]} = \frac{3}{10} \cdot \frac{10}{6} = \frac{1}{2}$$

Events A and B are independent because:

$$P[A] = P[A|B]$$

$$P[B] = P[B|A]$$

Knowing that the number is greater than three gives you no additional information about the number being even, because the number of even and odd numbers greater than three is the same; therefore, these events are independent.

In fact, we say that two events are independent, verifying that the probability of their intersection is the product of their probabilities; or you can verify that the conditional probability of one event, given the other has occurred, is the same as the probability of the unconditional event.

However, in practical cases, you will have the intuitive feeling that two events are independent and then calculate the probability of their intersection by multiplying their individual probabilities.

3. What are the probabilities related to the events A ° [n > 6] and B ° [n = even] ?

   Using Figure 63 as a reference:

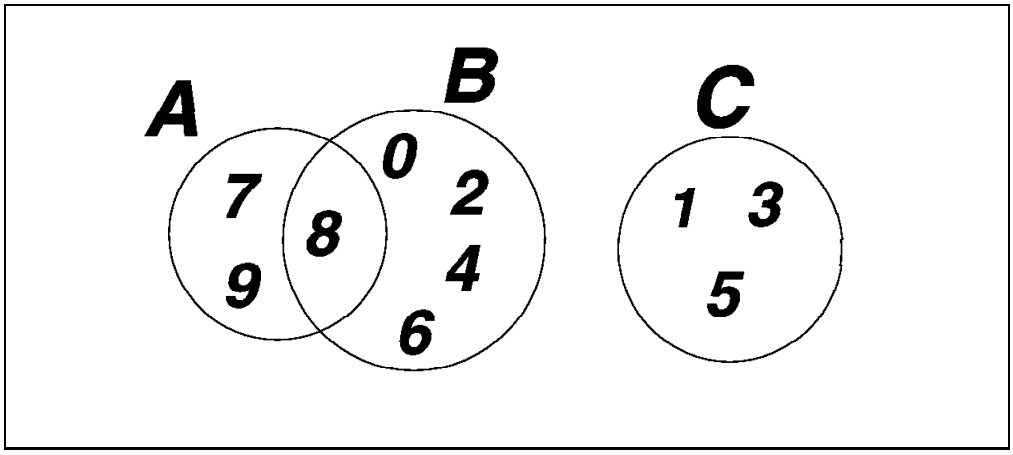

Figure 63. Event A (n > 6) and Event B (n = even)

$$P[A] \circ P[n > 6] = \frac{3}{10}$$

$$P[B] \circ P[n = \text{even}] = \frac{5}{10} = \frac{1}{2}$$

$$P[A \cap B] = \frac{1}{10}$$

$$P[A|B] = \frac{1}{10} \cdot \frac{10}{5} = \frac{1}{5}$$

$$P[B|A] = \frac{1}{10} \cdot \frac{10}{3} = \frac{1}{3}$$

$$P[A \cup B] = \frac{7}{10}$$

$$P[A] + P[B] - P[A \cap B] = \frac{3}{10} + \frac{5}{10} - \frac{1}{10} = \frac{7}{10}$$

If we did not subtract the probability of the intersection of A and B, it would be counted twice when the probabilities of A and B are added.

Events A and B are not independent because:

$$P[A] \neq P[A|B]$$

$$P[B] \neq P[B|A]$$

As with example 1 on page 154, these events are not independent because knowing that the number is greater than six reduces the rest of the numbers to one even number and two odd numbers.

4. What are the probabilities related to the events $A \circ [n > 7]$ and $B \circ [n < 3]$?

   Using Figure 64 as a reference:

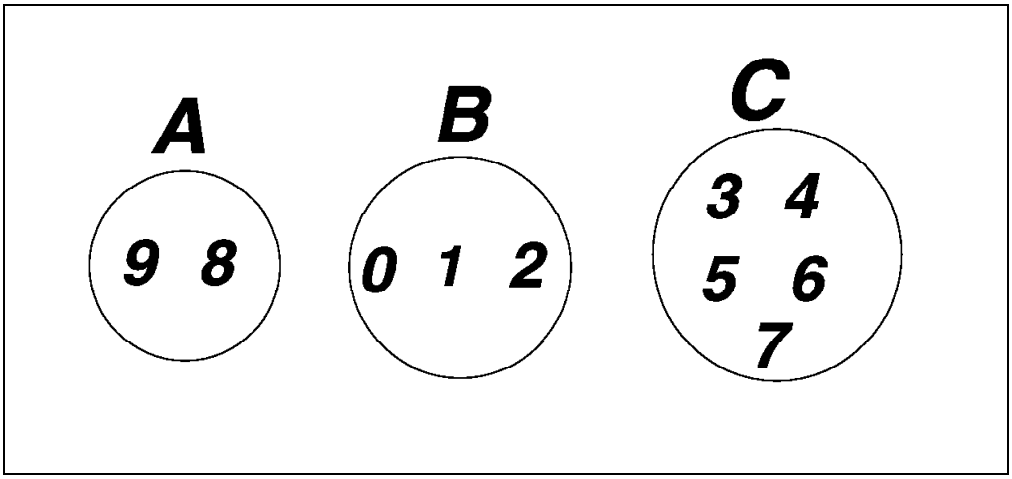

Figure 64. Event A (n > 7) and Event B (n < 3)

$$P[A] \circ P[n > 7] = \frac{2}{10}$$

$$P[B] \circ P[n < 3] = \frac{3}{10}$$

$$P[A \cap B] = 0$$

$$P[A|B] = 0$$

$$P[B|A] = 0$$

$$P[A] + P[B] = \frac{2}{10} + \frac{3}{10} = \frac{5}{10}$$

$$P[A \cup B] = \frac{5}{10}$$

You can see that events A and B are mutually exclusive, either from Figure 64 or by using equation (25):

$$P[A \cup B] = P[A] + P[B]$$

The concept of two events A and B being independent should not be confused with the concept of their being mutually exclusive. In fact, mutually exclusive events are not independent (except in the trivial case that at least one of them has zero probability) because the occurrence of one affects (excludes) the occurrence of the other. As you can see in this example of mutually exclusive events, they are not independent because:

$$P[A] \neq P[A|B]$$

$$P[B] \neq P[B|A]$$

5. The next two examples show you that sometimes physical intuition alone is not sufficient to decide whether two events are independent.

   a. Two fair coins are tossed; let E be the event “not more than one head” and F the event “at least one of each face.” Are the events E and F independent?

      The sample space for these events is defined as:

      $$S = \{HH, HT, TH, TT\}$$

      Therefore, we calculate the following probabilities:

      $$P[E] = P[\{HT, TH, TT\}] = \frac{3}{4}$$

      $$P[F] = P[\{HT, TH\}] = \frac{2}{4}$$

      $$P[E \cap F] = P[\{HT, TH\}] = \frac{2}{4}$$

      Hence:

      $$P[E \cap F] \neq P[E]\,P[F]$$

      and events E and F are not independent.

   b. Three fair coins are tossed; let E and F be events as described in the previous example. Are the events E and F independent?

      The sample space for these events is defined as:

      $$S = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}$$

      Therefore, we calculate the following probabilities:

      $$P[E] = P[\{HTT, THT, TTH, TTT\}] = \frac{4}{8}$$

      $$P[F] = P[\{HHT, HTH, HTT, THH, THT, TTH\}] = \frac{6}{8}$$

      $$P[E \cap F] = P[\{HTT, THT, TTH\}] = \frac{3}{8}$$

      Hence:

      $$P[E \cap F] = P[E]\,P[F]$$

      and events E and F are independent.

   It is not unusual for people to feel that the events E and F should either be independent or dependent in both examples. In fact, events E and F are independent only when the number of coins is equal to three; for all other cases, events E and F are dependent.

6. In a family with two children, what is the probability that both are boys (assume equal probabilities for having boys or girls) when you know:

   - Nothing else

     The sample space is S = {BB, BG, GB, GG} and you can easily see that the probability corresponding to event {BB} is 1/4.

   - At least one child is a boy

     The sample space reduces to the first three events, and therefore, the probability is 1/3.

   - The oldest child is a boy

     The sample space reduces to the first two events, and therefore, the probability is 1/2.

   Note the increase in the probability when your knowledge about the system (sample space) increases; although obvious from the definition, this point is often forgotten.

7. Out of 1,000 coins in a safe, 999 coins are perfect and one has two heads. Assuming that you get 10 heads out of 10 trials with a coin taken at random from the safe, what is the probability that the coin chosen is the one with two heads?

   Let us denote:

   $$P[2H] \circ \text{probability that the coin has two heads}$$
   $$P[10S] \circ \text{probability of getting 10 heads}$$

   We want to evaluate the conditional probability that the coin has two heads given that we got 10 heads; by definition:

   $$P[2H|10S] = \frac{P[2H \cap 10S]}{P[10S]}$$

   $$P[10S|2H] = \frac{P[2H \cap 10S]}{P[2H]} = 1$$

   The second equation is equal to 1 because you would always get 10 heads if the coin has two heads; hence:

   $$P[2H \cap 10S] = P[2H]$$

   $$\therefore \quad P[2H|10S] = \frac{P[2H]}{P[10S]}$$

   The probability $P[2H]$ is clearly 1/1000; the probability $P[10S]$ is the union of 1,000 mutually exclusive events:

   - Getting the coin with two heads
   - Getting any other coin and still get 10 heads (999 pairs of events)

   $$P[2H|10S] = \frac{\frac{1}{1000}}{\frac{1}{1000} + \frac{999}{1000} \cdot \frac{1}{2^{10}}} = 0.506$$

8. Two persons alternately flip a coin; the one who gets heads first is the winner. What are the probabilities of winning for each player?

   Let the event TTH, for example, mean:

   a. First player got tails
   b. Second player got tails
   c. First player got heads

   The first player wins ($\{W_1\}$) with the occurrence of either of the following mutually exclusive events:

   a. H
   b. TTH
   c. TTTTH
      ⋮

   Since each flip of the coin is independent of the others, the probability of winning for the first player is:

$$P[W_1] = \frac{1}{2^1} + \frac{1}{2^3} + \frac{1}{2^5} + \cdots = \sum_{k=0}^{\infty} \frac{1}{2^{2k+1}}$$

   This is a geometric series (see “Geometric Series” on page 174) with first term 1/2 and common ratio 1/4; therefore:

$$P[W_1] = \frac{\frac{1}{2}}{1 - \frac{1}{4}} = \frac{2}{3}$$

   In a similar way, the second player wins ($\{W_2\}$) with the occurrence of either of the following mutually exclusive events:

   a. TH
   b. TTTH
   c. TTTTTH
      ⋮

$$P[W_2] = \frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^6} + \cdots = \sum_{k=1}^{\infty} \frac{1}{2^k}$$

   This is also a geometric series with first term 1/4 and common ratio 1/4; therefore:

$$P[W_2] = \frac{\frac{1}{4}}{1 - \frac{1}{4}} = \frac{1}{3}$$

   Therefore, the first player has the double of the probability of winning compared to the probability of the second player winning.

9. A hiker leaves point A shown in Figure 65, choosing one of the possible roads at random. At each subsequent crossroads he again chooses a road at random. What is the probability of the hiker arriving at the point D?

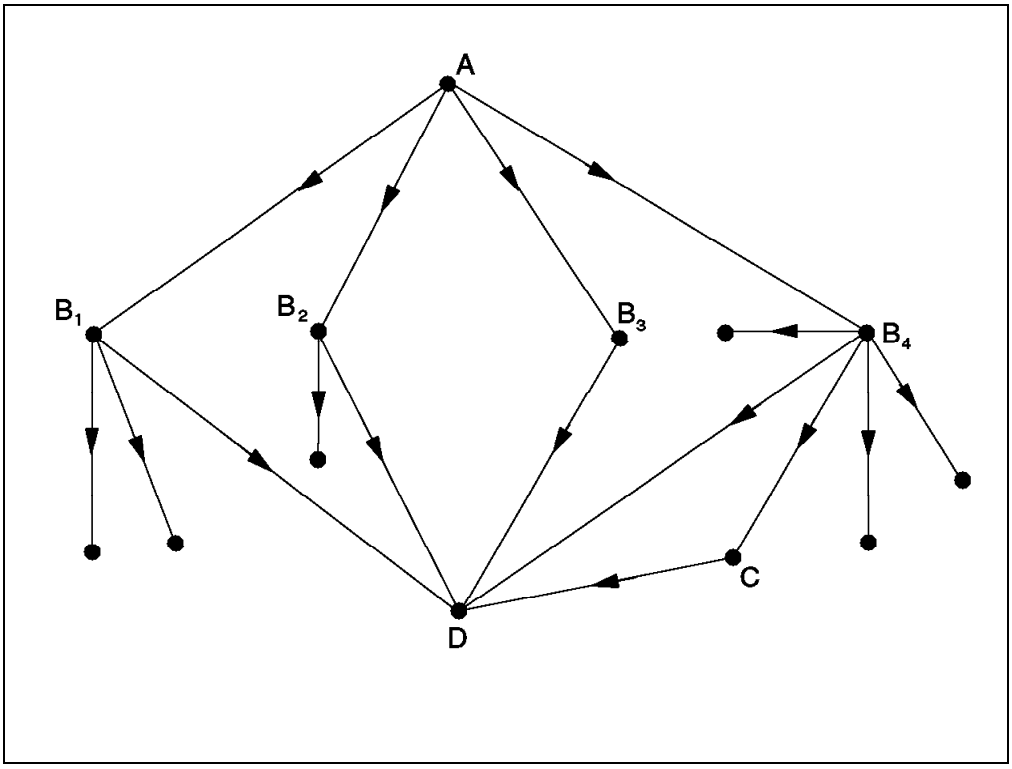

Figure 65. Hiker Paths

Let the event that the hiker passes through a point B_k (k=1, 2, 3, or 4) be denoted by $P[B_k]$. Then B_1, B_2, B_3, and B_4 form a full set of mutually exclusive events, since the hiker must pass through one of these points.

$$P[B_k] = \frac{1}{4}$$

Once having arrived at B_k, the hiker can proceed to D only by making the proper choice of one of the equiprobable roads. Hence the conditional probabilities of arriving at D from B_k are:

$$P[D|B_1] = \frac{1}{3} \qquad P[D|B_2] = \frac{1}{2}$$

$$P[D|B_3] = 1 \qquad P[D|B_4] = \frac{2}{5}$$

Using the total probability theorem given by equation (27):

$$P[D] = P[B_1]P[D|B_1] + P[B_2]P[D|B_2] +$$

$$+ P[B_3]P[D|B_3] + P[B_4]P[D|B_4]$$

$$\therefore \quad P[D] = \frac{1}{4} \cdot \left(\frac{1}{3} + \frac{1}{2} + 1 + \frac{2}{5}\right) = \frac{67}{120} = 0.558$$

There is a probability of 55.8% that the hiker arrives at point D.

10. What is the probability that at least two out of *n* people ($n < 365$) in a room have the same birthday? This is one of the most famous problems in probability theory and illustrates its nonintuitive aspects.

    Assuming that nobody was born on February 29th, there are 365^n possible cases for the birthdays of *n* people.

    Let E be the event that at least two of the *n* people have the same birthday. It is easier to evaluate the complement event ($\bar{E}$) that everyone has a different birthday:

    - The first person has 365 possible birthdays.
    - The second person has 364 possible birthdays for each of the 365 possible cases of the first person.
    - The third person has 363 possible birthdays.
      ⋮
    - The n^{th} person has $365 - n + 1$ possible birthdays.

    Therefore:

$$P[\bar{E}] = \frac{365 \cdot 364 \cdot 363 \cdot \cdots \cdot (365 - n + 1)}{365^n}$$

$$P[E] = 1 - P[\bar{E}]$$

    Table 15, produced with the sample REXX program shown in Figure 66, lists the results for various values of *n*.

    It is rather surprising to find that, with 23 people in a room, the probability of at least two with the same birthday is better than flipping a coin. With 60 or more people, you are almost sure to find at least two people with the same birthday.

Table 15. Birthday Problem

n	P[E] (%)	n	P[E] (%)	n	P[E] (%)
2	0.273973	3	0.820417	5	2.713557
10	11.694818	15	25.290132	20	41.143838
22	47.569531	**23**	**50.729723**	24	53.834426
30	70.631624	40	89.123181	50	97.037358
60	99.412266	75	99.971988	100	99.999969

```
/* Probability of coincident birthdays for n people */
/* Parameter: maximum number of persons              */

  arg max
  if max = '' then max = 100
  P.1 = 1
  do n = 2 to max
    j = n-1
    P.n = P.j * (365-n+1)/365
    say n (1-P.n)*100
  end n
```

Figure 66. Birthday Problem

Discrete Random Variables

A random variable is merely a function that maps one set of events of a sample space to a set of real numbers. A random variable *X* is said to be discrete when it takes only a finite or a countably infinite number of distinct values *x*, with the following properties:

$$1)\quad p(x) = P[X = x] \tag{28}$$

$$2)\quad P[a \le x \le b] = \sum_{a}^{b} p(x) \tag{29}$$

$$3)\quad \sum_{x=-\infty}^{\infty} p(x) = 1 \tag{30}$$

where *p(x)* is called the probability mass function (pmf), and the summation is over all values taken by the random variable.

Probability Mass Function

Let us define a random variable consisting of the sum of both faces in a 2-dice throwing experiment, for example. The sample space is composed of 36 pairs of possible outcomes of the experiment; the domain of the pmf is the set containing 11 elements corresponding to the sum of both faces of the dice (2 to 12), and its range is the corresponding probabilities. Figure 67 represents the pmf for the 2-dice throwing experiment.

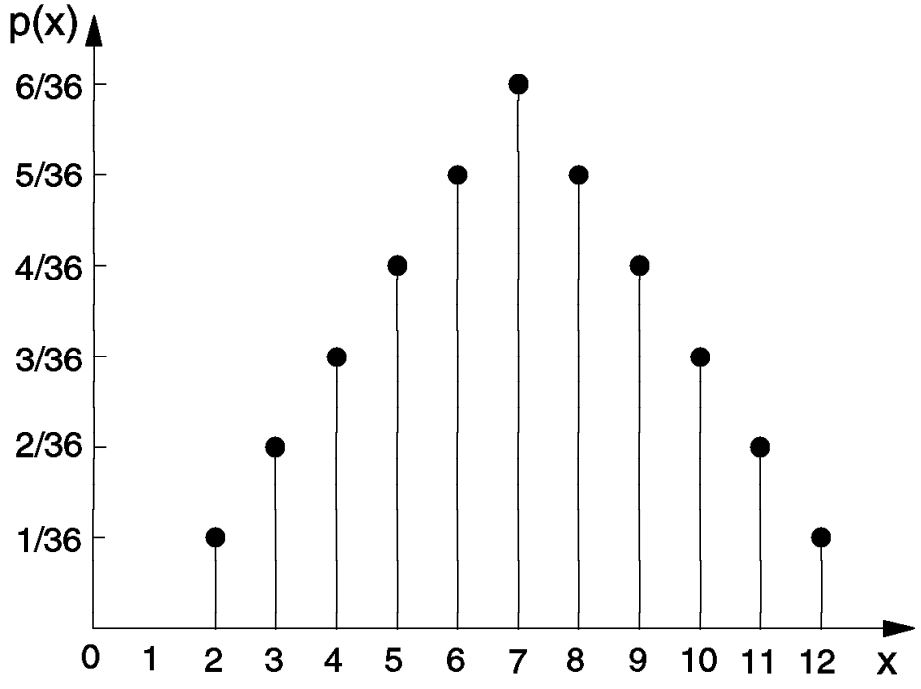

Figure 67. Probability Mass Function Example

Probability Distribution Function

We are often interested not in the probability that the value of a random variable *X* is a particular value, but rather in the probability that *X* has a value less than or equal to some number.

The function that provides the probabilities for $X \leq x$ is called the probability distribution function (PDF) or distribution function of the random variable *X*:

$$F(x) = P[X \leq x]$$

In other words, the PDF has the set of all real numbers as its domain, and the function assigns to each real number *x* the probability that *X* has a value less than or equal to the number *x*.

Let us use the 2-dice throwing experiment to define the properties of the PDF for discrete variables: The corresponding values for this PDF are:

$$F(1) = P[X \leq 1] = 0$$

$$F(2) = P[X \leq 2] = p(2) = \frac{1}{36}$$

$$F(3) = P[X \leq 3] = F(2) + p(3) = \frac{1}{36} + \frac{2}{36} = \frac{3}{36}$$

$$F(4) = P[X \leq 4] = F(3) + p(4) = \frac{3}{36} + \frac{3}{36} = \frac{6}{36}$$

...

$$F(12) = P[X \leq 12] = F(11) + p(12) = \frac{35}{36} + \frac{1}{36} = \frac{36}{36} = 1$$

Note: You can see that the PDF accumulates all values up to the corresponding range value; this explains why this function is also known as the cumulative distribution function (CDF).

Continuing in this way, you can tabulate all the values corresponding to the range of the random variable, using the equation:

$$F(x) = P[X = x] = p(x)$$

The domain of the PDF is the set of all real numbers; hence, you must find the value F(x) for all numbers, not just those in the range of the random variable. For example, to find F(3.5) for the dice experiment, you note that the event $X \leq 3.5$ corresponds to p(2) + p(3)

The graph for the PDF of a discrete distribution is, therefore, a step function with a finite number of jumps, as shown in Figure 68, representing the PDF for the dice experiment. The PDF has a discontinuity at each point of positive probability with jumps equal to the corresponding probabilities.

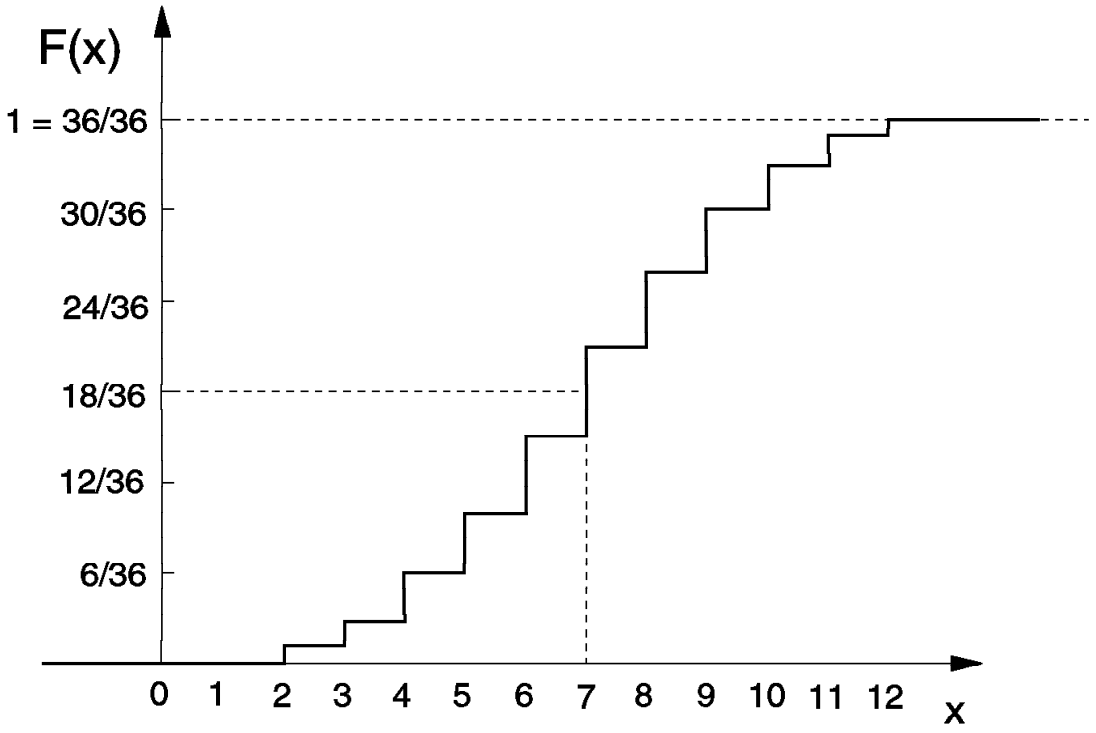

Figure 68. Distribution Function for the Dice Experiment

Some properties of the PDF of a discrete random variable are:

- The value of a PDF is always between zero and one:

$$0 \leq F(x) \leq 1 \tag{31}$$

- F(x) is a nondecreasing function:

$$F(b) \geq F(a) \quad \text{if } b \geq a \tag{32}$$

- The probability for $X > x$ is the complement of the PDF:

$$P[X > x] = 1 - P[X \leq x] = 1 - F(x) \tag{33}$$

- Using p(x) for the pmf and F(x) for the PDF, we can write:

$$\begin{aligned} P[a < X \leq b] &= F(b) - F(a) \\ P[a \leq X \leq b] &= F(b) - F(a) + p(a) \\ P[a \leq X < b] &= F(b) - F(a) + p(a) - p(b) \\ P[a < X < b] &= F(b) - F(a) - p(b) \end{aligned} \tag{34}$$

You can prove the first statement when you consider that the events $a < X \leq b$ and $X \leq a$ are mutually exclusive:

$$P[X \leq b] = P[X \leq a] + P[a < X \leq b]$$

The other statements can be proven in a similar way.

Continuous Random Variables

A random variable is said to be continuous if it can assume any numerical values on a given interval; the following properties apply:

$$1)\quad F(x) = P[X \leq x] = \int_{-\infty}^{x} f(x)dx \qquad (35)$$

$$2)\quad P[a \leq x \leq b] = \int_{a}^{b} f(x)dx \qquad (36)$$

$$3)\quad \int_{-\infty}^{\infty} f(x)dx = 1 \qquad (37)$$

The function *f(x)* is a nonnegative continuous function, called the probability density function (pdf) of the random variable, and the integral is over all values taken by the random variable. The function *F(x)* is the probability distribution function (PDF) with similar properties of PDFs for discrete random variables.

Some of the consequences of the definitions are:

- The PDF is the derivative of the pdf; therefore, once one of the functions is known, the other can be evaluated by integration or derivation.
- The probability of a continuous random variable *X* assuming a value in the interval *(a,b)* is the area under the pdf between the values *a* and *b*.
- The total area under the pdf function is equal to one.

Probability Density Function

You can see the similarity between the definition of the pdf for a continuous random variable and the definition of the pmf for a discrete random variable. However, equation (36) shows that the probability of *X* assuming a value in the interval *(x,x+dx)* is equal to *f(x) dx*. Therefore, the probability that *X* is exactly equal to *x* is zero. In other words, in a continuous distribution where the random variable can assume an infinite number of values within an interval, the event corresponding to the random variable exactly equal to one of the values has a probability of zero:

$$P[X = x] = p(x) = 0 \qquad (38)$$

Whereas the pmf describes the probabilities that the random variable assumes for each value of its range, the density function says that the probability the random variable is within an infinitesimal interval *dx* is equal to *f(x) dx*.

Since the pmf of a continuous random variable is always equal to zero, it is useless to describe a continuous distribution. Therefore, whenever we study a continuous random variable we either refer to the density function (pdf) or the distribution function (PDF).

Suppose a point is tossed at random onto an interval *(a,b)*. This means that the probability of the point falling in a subinterval (x,x + Dx) does not depend on the location of the subinterval. Therefore, the probability of the point falling onto any subinterval (x,x + Dx) is proportional to the length Dx of the subinterval.

These considerations are translated into the following equations:

$$P[x \leq X \leq x + dx] = \frac{dx}{b - a}$$

$$P[a \leq X \leq b] = \int_a^b \frac{dx}{b - a} = 1$$

This random variable is said to be uniformly distributed and its pdf is given by equation (39). The pdf is plotted in Figure 69 and you can easily see why this random variable is said to be uniformly distributed.

$$f(x) = \begin{cases} \frac{1}{b - a} & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases} \qquad (39)$$

The probability of the point falling in interval *(c,d)* is given by the area under the pdf between points *c* and *d*, as shown by the following equation:

$$P[c \leq X \leq d] = \int_c^d \frac{dx}{b - a}$$

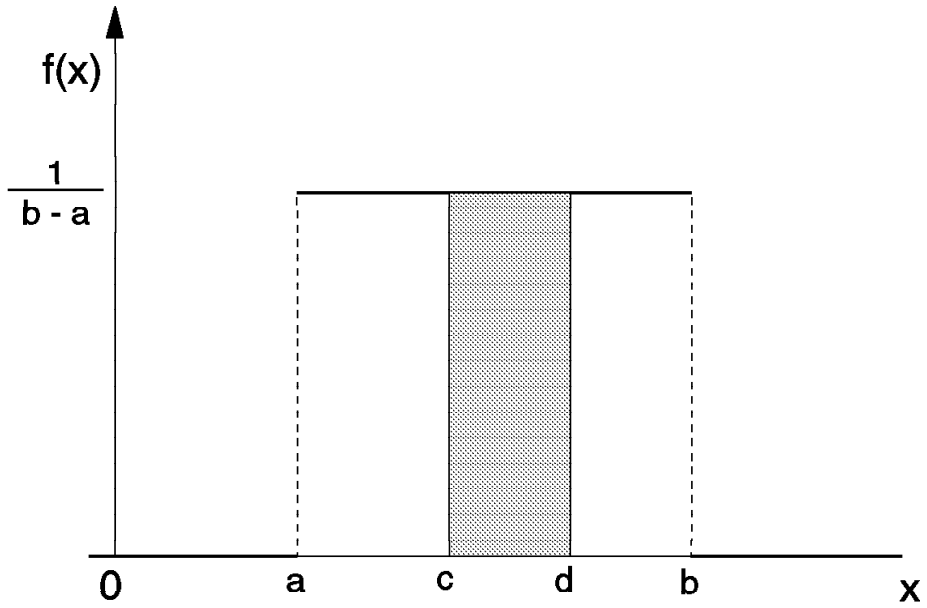

Figure 69. Probability Density Function Example

Probability Distribution Function

The properties of discrete distribution PDFs described by equation (31), equation (32), and equation (33) are also valid for continuous distributions. However, the properties described by equations (34) must be modified as a consequence of the pmf being zero for continuous distributions, as established by equation (38):

$$\begin{aligned} P[a < X \leq b] &= P[a \leq X \leq b] \\ &= P[a \leq X < b] \\ &= P[a < X < b] \\ &= F(b) - F(a) \end{aligned} \tag{40}$$

The PDF of a random variable *X* describes how the probability mass of *X* is distributed along the real line. From this point of view, *X* determines how the probability mass of one unit is spread out over the real numbers. A discrete random variable allocates the mass in nuggets or mass points, while a continuous random variable diffuses the probability mass out in a continuous manner.

Since the PDF is the integral of the pdf, as shown by equation (35), the PDF for the continuous uniform distribution, plotted in Figure 70, is given by:

$$F(x) = \int_a^x \frac{dx}{b-a} = \frac{x}{b-a}\Big|_a^x = \frac{x-a}{b-a} \tag{41}$$

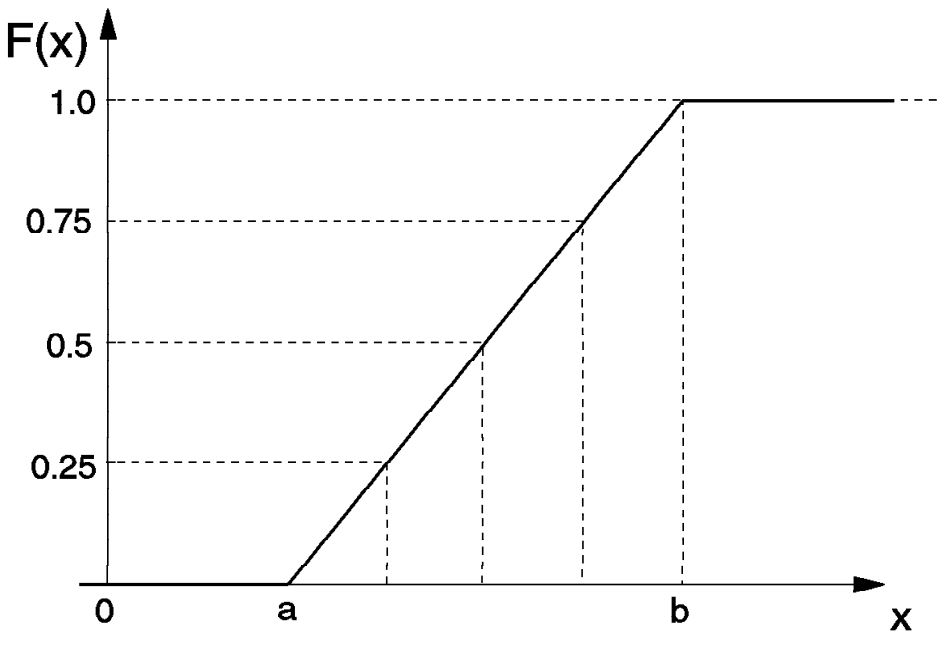

Figure 70. Distribution Function for a Uniform Random Variable

Parameters of Random Variables

In many problems, the random variable has a rather complicated distribution; therefore, it is desirable to be able to describe some features of the random variable by means of a few numbers, such as mean and variance, that can be computed from its distribution. For some purposes, these numbers, rather than the entire function, are all that is needed.

All possible probability computations involving a random variable can be made from its pmf, if it is discrete; from its density function (pdf), if it is continuous; or from its distribution function (PDF), in either case.

Mean

The mean, or expected value, of a random variable is a measure of location in the sense that it roughly locates a middle or average value of the random variable. Although not strictly correct, whenever people refer to an average value they are normally referring to the mean value.

As an example, if a student wants to compute his average grade in a course in which his six grades are 75, 90, 75, 87, 75, and 90, he would add all grades and divide by the number of grades:

$$\frac{75 + 90 + 75 + 87 + 75 + 90}{6} = 75 \cdot \frac{3}{6} + 87 \cdot \frac{1}{6} + 90 \cdot \frac{2}{6} = 82$$

This indicates that the average is computed by multiplying each grade by its relative frequency, or probability, with which it occurs among all other grades.

The mean of a discrete random variable X with pmf $p(x)$, denoted by $\overline{X}$ or $E[X]$, is defined as:

$$E[X] = \overline{X} = \sum_{i=-\infty}^{\infty} x_i p(x) \qquad (42)$$

Applying this definition to our 2-dice experiment, we have:

$$\overline{X} = 2 \cdot \frac{1}{36} + 3 \cdot \frac{2}{36} + 4 \cdot \frac{3}{36} + \cdots + 12 \cdot \frac{1}{36} = 7$$

Similarly, the mean of a continuous random variable X with pdf $f(x)$ is defined as:

$$E[X] = \overline{X} = \int_{-\infty}^{\infty} x f(x) dx$$

Applying this definition to the uniform random variable, we can write:

$$E[X] = \int_a^b \frac{x dx}{b - a} = \left. \frac{x^2}{2(b - a)} \right|_a^b = \frac{b^2 - a^2}{2(b - a)} = \frac{a + b}{2}$$

Mode

The mode is the most frequent value in the distribution; in other words, it is the most likely value that the random variable may assume.

The mode is determined by evaluating the point where the pmf (discrete random variable) or the pdf (continuous random variable) assumes a relative maximum; therefore, a random variable may have any number of modes.

Analytically, the mode is calculated by equating the first derivative of the pmf or pdf to zero and checking whether the second derivative is negative for the corresponding value.

The mode of our dice example is 7 (same as the mean) since it yields the maximum for the pmf, as shown in Figure 67 on page 163.

A continuous uniform distribution has no modes because no value has probability greater than the others, as you can see from Figure 69 on page 167.

Median and Percentiles

The median is the central value after the set of values has been ordered; it is calculated for both discrete or continuous distributions, as the value that makes the PDF equal to 1/2.

For discrete random variables, if the number of values is odd, the median is the central value; for an even number of values, the median can be any value between the two central values, but it is usually taken as the arithmetic average of these two values.

In our dice example, the median is 7 (same as the mean and mode), which is clear from Figure 68 on page 165. In fact, in all symmetric distributions, the mean and the median are always the same; if the symmetric distribution has just one mode, it is also equal to the mean and median.

In the case of continuous uniform distribution, the median can be calculated by making the PDF in equation (41) equal to 0.5:

$$\frac{x-a}{b-a} = \frac{1}{2} \quad \backslash \quad x = \frac{a+b}{2}$$

which is the same value as the mean.

In the same way that the median divides the distribution in two regions of equal probability, the percentiles determine regions according to an established percentage. For example, Figure 70 on page 168 shows the uniform distribution divided in regions of 25% probability each, and we say that they are the 25th, 50th (which is the median), and 75th percentile values (also called quartiles).

For any random variable *X*, the *r*th percentile, denoted as $p(r)$, is defined as:

$$P[X \leq p(r)] = \frac{r}{100} \qquad (43)$$

To calculate the percentiles, you proceed as explained for the median, but replacing the 0.5 by the corresponding percentile value.

Variance

The mean value represents the average, or expected value, of a random variable. However, because different distribution functions may have the same mean value, sometimes we need more information to fully describe one particular distribution.

It is necessary then to define something that describes the dispersion of the values; that is, how much each value assumed by the random variable differs from the mean value.

The variance of a discrete random variable X, denoted by σ^2, is defined as:

$$\sigma^2 = E[(X - \overline{X})^2] \tag{44}$$

This definition says that the variance of a random variable is the expected value of the square of differences between each value on the range of the random variable and its mean. The value σ, called standard deviation, measures how much on average the values differ from the mean.

Using equation (42) and p(x) as the probability function of the random variable, we can rewrite equation (44) as:

$$\sigma^2 = \sum_{i=-\infty}^{\infty} (x_i - \overline{X})^2 p(x) \tag{45}$$

We can use equation (45) to compute the variance of a random variable, but it usually requires less work if we use another technique derived from equation (44):

$$\sigma^2 = E[X^2] - 2\overline{X}E[X] + \overline{X}^2 = E[X^2] - 2\overline{X}^2 + \overline{X}^2$$

$$\therefore \quad \sigma^2 = E[X^2] - (E[X])^2 \tag{46}$$

The term $E[X^2]$ is called second moment of the random variable *X*, in the same way that the mean is also called first moment in a mechanical analogy with the center of mass and moment of inertia.

Similarly, the variance of a continuous random variable *X* with pdf *f(x)* can be calculated in either of the following ways:

1) $$\sigma^2 = \int_{-\infty}^{\infty} (x - \overline{X})^2 f(x)dx$$

2) $$\sigma^2 = \int_{-\infty}^{\infty} x^2 f(x)dx - \overline{X}^2$$

As an example, let us calculate the variance for our dice example using both methods. According to equation (44):

$$s^2 = (2-7)^2\frac{1}{36} + (3-7)^2\frac{2}{36} + (4-7)^2 + \cdots +$$

$$+ (10-7)^2\frac{3}{36} + (11-7)^2\frac{2}{36} + (12-7)^2\frac{1}{36} = 5.833$$

Using the second method, described by equation (46):

$$E[X^2] = 4 \cdot \frac{1}{36} + 9 \cdot \frac{2}{36} + \cdots + 121 \cdot \frac{2}{36} + 144 \cdot \frac{1}{36} = \frac{1974}{36}$$

$$\therefore \quad s^2 = \frac{1974}{36} - 7^2 = 5.833$$

Let us now calculate the variance of the continuous uniform random variable:

$$s^2 = \int_a^b \frac{\left(x - \frac{a+b}{2}\right)^2}{b-a}\,dx = \left.\frac{\left(x - \frac{a+b}{2}\right)^3}{3(b-a)}\right|_a^b$$

$$= \frac{\left(b - \frac{a+b}{2}\right)^3 - \left(a - \frac{a+b}{2}\right)^3}{3(b-a)} = \frac{(b-a)^3}{12(b-a)}$$

$$\therefore \quad s^2 = \frac{(b-a)^2}{12}$$

Calculating the moments of a random variable, and therefore the mean and the variance, can be a tedious process that can be overcome by transform methods such as the moment generation function and z-transforms; however, these techniques are outside the intended scope of this document.

Coefficient of Variation

The squared coefficient of variation of a random variable X is defined as:

$$c_x^2 = \frac{s^2}{\overline{X}^2} \qquad (47)$$

This coefficient is widely used in intermediate and advanced queueing theory because it measures the degree of irregularity of a random variable. In this document we refer to the squared coefficient of variation simply as the coefficient of variation.

Arithmetic Series

An arithmetic series is characterized by the first element, which can be arbitrary, and a constant known as common ratio. Every element after the first is equal to the previous element plus the common ratio. The sum of the first *n* elements of an arithmetic series with first element a_1 and common ratio *r* is represented by:

$$S = \sum_{k=0}^{n-1} (a_1 + kr)$$

and can be written in two different ways:

$$S = a_1 + a_2 + a_3 + \cdots + a_{n-2} + a_{n-1} + a_n$$

$$S = a_n + a_{n-1} + a_{n-2} + \cdots + a_3 + a_2 + a_1$$

Adding these two equations and grouping the terms, we can write:

$$2S = (a_1 + a_n) + (a_2 + a_{n-1}) + \cdots + (a_{n-1} + a_2) + (a_n + a_1) \qquad (48)$$

All terms in equation (48) are equal because:

$$a_2 + a_{n-1} = a_1 + r + a_{n-1} = a_1 + a_n$$

$$a_3 + a_{n-2} = a_2 + r + a_{n-2} = a_2 + a_{n-1} = a_1 + a_n$$

$$a_4 + a_{n-3} = a_3 + r + a_{n-3} = a_3 + a_{n-2} = a_1 + a_n$$

...

Therefore we can replace all these *n* terms in equation (48) by their single common value:

$$2S = (a_1 + a_n)n$$

$$\therefore \quad S = \frac{(a_1 + a_n)n}{2} \qquad (49)$$

Equation (49) is used when you know the first term, the last term, and the number of elements of the arithmetic series. An alternative is using the first term, the number of elements and the common ratio; since the *n*th element of an arithmetic series is:

$$a_n = a_1 + (n - 1)r$$

the sum of the first *n* elements can also be written as:

$$S = \frac{[2a_1 + (n - 1)r]n}{2}$$

Geometric Series

A geometric series is characterized by the first element, which can be arbitrary, and a constant known as common ratio. Every element after the first is equal to the previous element multiplied by the common ratio.

The sum of the first *n* elements of a geometric series with a first element a_1 and common ratio *q* can be written as:

$$S = a_1 + a_1 q + a_1 q^2 + \cdots + a_1 q^{n-1} = \sum_{k=0}^{n-1} a_1 q^k = \frac{a_1(q^n - 1)}{q - 1}$$

For $q < 1$, the geometric series converge when the number of elements approaches infinity because q^n approaches zero:

$$S = \lim_{n \to \infty} \frac{a_1(q^n - 1)}{q - 1} = \frac{a_1(-1)}{q - 1}$$

$$\therefore \quad S = \frac{a_1}{1 - q} \qquad (50)$$

Power of Natural Numbers

This type of series is composed of *n* natural numbers to the *m*th power. The sum of the elements is:

$$S = \sum_{k=1}^{n} k^m$$

The general expression for the sum of the elements of these type of series are quite complex; therefore, we only present the expressions for the first three cases (*m=1,2,3*). Note that for *m=1* we have an arithmetic series with common ration equal to 1.

- $m = 1 \quad \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
- $m = 2 \quad \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$
- $m = 3 \quad \sum_{k=1}^{n} k^3 = \left[\frac{n(n+1)}{2}\right]^2$

DASD Seek Model

Let us analyze the case of a DASD volume with *n* cylinders that has been loaded with data in such way that successive seeks are independent of one another.

Let *X* be the random variable representing the number of cylinders passed between consecutive seeks (seek length) and p_i the probability that a randomly chosen seek takes the actuator to cylinder *i*.

1. What is the pmf for *X*, assuming the first cylinder is cylinder zero?

   The probability of a zero-length seek is the probability that the access mechanism is at cylinder *i* and stays in the same cylinder *i*. Since these are independent events, each zero-length seek has probability p_i^2; adding the probabilities of all possible zero-length seeks we have:

$$p(x) = \sum_{i=0}^{n-1} p_i^2 \qquad x = 0 \tag{51}$$

   The probability of a x-length seek ($x > 0$) is the probability of being at cylinder *i* and the occurrence of a seek to cylinder *i+x*. Since each such event may occur twice, from *i* to *i+x* and from *i+x* to *i*, we have:

$$p(x) = 2\sum_{i=0}^{n-1-x} p_i p_{i+x} \qquad x = 0,1,2, \ldots , n-1 \tag{52}$$

   Equation (52) means, for example, that the probability of a 5-cylinder seek (*x=5*) in a 1440-cylinder disk device is twice the sum of the probabilities of:

   - Being on cylinder 0 and seek to cylinder 5 ($p_0\, p_5$)
   - Being on cylinder 1 and seek to cylinder 6 ($p_1\, p_6$)

     ⋮
   - Being on cylinder 1434 and seek to cylinder 1439 ($p_{1434} p_{1439}$)

2. What is the pmf for *X* in case of a perfectly random seek pattern?

   For a perfectly random seek pattern, the probability to seek to any cylinder is the same, and therefore, equal to *1/n*. The pmf expressed by equations (51) and (52) can be simplified to:

$$p(x) = \left(\sum_{i=0}^{n-1} \frac{1}{n}\right)^2 = \frac{n}{n^2} = \frac{1}{n} \qquad x = 0$$

$$p(x) = 2\left(\sum_{i=0}^{n-1-x} \frac{1}{n}\right)^2 = \frac{2(n-x)}{n^2} \qquad x = 0,1,2, \ldots , n-1$$

3. What is the mean and variance for the seek length?

   By definition, the average seek length is:

$$E[X] = \sum_{x=0}^{n-1} x\,p(x) = 0 \cdot \frac{1}{n^2} + \sum_{x=1}^{n-1} \frac{2x(n-x)}{n^2}$$

$$= \frac{2}{n^2}\left(n\sum_{x=1}^{n-1} x - \sum_{x=1}^{n-1} x^2\right) = \frac{2}{n^2}\left(n\sum_{x=1}^{n-1} x - \sum_{x=1}^{n-1} x^2\right)$$

   The two summations represent the sum of *n-1* natural numbers to the first and second power respectively, as described in “Power of Natural Numbers” on page 174. Replacing the summations by their respective expressions and remembering to correct the expressions to reflect *n-1* elements:

$$E[X] = \frac{2}{n^2}\left[\frac{n(n-1)n}{2} - \frac{(n-1)(n)(2n-1)}{6}\right]$$

$$= \frac{2}{n^2}\left[\frac{3n^2(n-1) - n(n-1)(2n-1)}{6}\right]$$

$$= \frac{2}{n^2}\left[\frac{n(n-1)(n+1)}{6}\right]$$

$$= \frac{n^2-1}{3n}$$

   Normally, the number of cylinders is in the order of thousands and the error of writing $n^2 - 1 = n^2$ is negligible:

$$\therefore \quad E[X] \approx \frac{n}{3}$$

   Therefore, the average seek length of a disk device with a large number of cylinders ($n > 100$) is one-third of the total number of cylinders.

   The second moment is given by:

$$E[X^2] = \sum_{x=1}^{n-1} \frac{2x^2(n-x)}{n^2} = \frac{2}{n^2}\left(n\sum_{x=1}^{n-1} x^2 - \sum_{x=1}^{n-1} x^3\right)$$

   If you develop the expression of the series, you will find that:

$$E[X^2] \approx \frac{n^2-1}{6}$$

$$s_x^2 = E[X^2] - (E[X])^2 \approx \frac{n^2}{18}$$

4. Assuming that the seek time (*T*) is a linear function of the seek length (*X*), what is the average and variance of the seek time?

   We can translate the assumption by:

$$T = \begin{cases} 0 & x = 0 \\ T = aX + b & x = 0,1,2,\ldots,n-1 \end{cases}$$

   We are basically assuming that:

   - The time to switch heads (zero-length seeks) is negligible.
   - The seek time is the same linear function of the seek length for any number of cylinders crossed.

   In order to evaluate the two constants *a* and *b*, we consider that the seek is composed of a basic overhead, which is the time spent to accelerate and decelerate the access mechanism (*b*), and the time to cross one cylinder multiplied by the number of cylinders crossed.

   If we consider the one-cylinder seek as basically overhead (called minimum seek time), we can write:

$$a = \frac{\text{maximum\_seek\_time} - \text{minimum\_seek\_time}}{n-1}$$

$$b = \text{minimum\_seek\_time}$$

   The average seek time is:

$$E[T] = \sum_{x=0}^{n-1} (ax + b)p(x) = aE[X] + b$$

$$\therefore \quad E[T] \approx \frac{an}{3} + b$$

   The variance is calculated in a similar way:

$$E[T^2] = a^2E[X^2] + 2abE[X] + b^2$$

$$(E[T])^2 = a^2(E[X])^2 + 2abE[X] + b^2$$

$$s_t^2 = E[T^2] - E[T]^2 = a^2 s_X^2$$

$$\therefore \quad s_t^2 \approx \frac{a^2n^2}{18}$$

   The time to cross cylinders during the acceleration and deceleration of the actuator is greater than the time to cross the same number of cylinders when the actuator is fully accelerated; the same process occurs when the actuator is decelerating.

   The seek time random variable may be better modelled with two or three linear functions of the seek length rather than one fixed function, as considered in this example.

RPS-miss Time

RPS-miss time is not as important as it used to be because the current storage controls have the required logic to reduce it or even eliminate it, as discussed in “RPS Miss Avoidance” on page 93. However, RPS misses are still relevant for other storage controls such as the IBM 3990 Models 1 and 2, and FBA adapters.

As discussed before, RPS misses occur because the device is able to disconnect from the path during certain activities that are related exclusively to the device, such as seek and latency.

The device tries to reconnect to the path when the device is about to be oriented to the requested record. Since the device is in continuous rotation, if the path is not free at the moment of the reconnection, the device must orient itself again to the requested record in the next revolution.

Note: Actually, there is a window (two or three sectors, or one FBA block) in which the device is able to reconnect. Since this analysis considers just one moment for the probability of reconnection, the estimated RPS-miss time equation is conservative.

The probability of missing exactly one revolution is the probability of the intersection of two independent events:

- The path is busy during the first attempt.
- The path is free during the second attempt.

The probability of missing exactly two revolutions is the probability of the intersection of three independent events:

- The path is busy during the first attempt.
- The path is busy during the second attempt.
- The path is free during the third attempt.

Therefore, using r as the probability of path busy, the probabilities of missing 1, 2, 3, ... n revolutions are:

$$P_1 = r(1 - r) = r(1 - r)$$

$$P_2 = rr(1 - r) = r^2(1 - r)$$

$$P_3 = rrr(1 - r) = r^3(1 - r)$$

$$\vdots$$

$$P_n = rr \ldots r(1 - r)$$

$$\therefore \quad P_n = r^n(1 - r) \qquad (53)$$

A First Approach

Equation (53) is the probability mass function (pmf) for the number of missed revolutions; its average can be calculated using equation (42) as described in “Mean” on page 169:

$$\bar{n} = \sum_{n=1}^{\infty} nP_n = \sum_{n=1}^{\infty} n(1-r)r^n = (1-r)\sum_{n=1}^{\infty} nr^n \tag{54}$$

$$\bar{n} = (1-r)\left(\sum_{n=1}^{\infty} n + \sum_{n=2}^{\infty} n + \sum_{n=3}^{\infty} n + \sum_{n=4}^{\infty} n + \cdots\right) \tag{55}$$

$$\bar{n} = (1-r)\sum_{i=1}^{\infty}\sum_{n=i}^{\infty} n$$

Equation (55) is composed of several geometric series that converge when $r < 1$, which is the case for the probability of the path busy.

Replacing each geometric series with its limit, as established by equation (50) in “Geometric Series” on page 174:

$$\bar{n} = (1-r)\left(\frac{r}{1-r} + \frac{r^2}{1-r} + \frac{r^3}{1-r} + \cdots\right)$$

$$= (1-r)\sum_{i=1}^{\infty}\frac{r^i}{1-r}$$

$$= (1-r)\frac{r}{(1-r)^2}$$

$$= \frac{r}{1-r}$$

The average number of lost revolutions is then:

$$\bar{n} = \frac{r}{1-r} \tag{56}$$

Using *PPB* for the probability of path busy, as used in equation (3) in “RPS Miss” on page 69, the RPS-miss time is the average number of lost revolutions multiplied by the revolution time:

$$T_{rps} = \frac{PPB}{1-PPB}T_{rev} \tag{57}$$

Using Derivatives

In the previous section we evaluated the expression (equation (54)):

$$\bar{n} = \sum_{n=1}^{\infty} nP_n = (1-r)\sum_{n=1}^{\infty} nr^n \qquad (58)$$

expanding the argument of the summation into several geometric series. This is cumbersome and becomes harder to use when the multiplier of the geometric series is more complex. This section presents a general technique to evaluate these type of expressions, using derivatives and their properties.

Let us review some basic concepts of derivatives relevant to this technique:

- The derivative of u^n is nu^{n-1}
- The derivative of a sum is the sum of the derivatives
- The derivative of the ratio of two functions is:

$$\frac{d}{dx}\left(\frac{u}{v}\right) = \frac{v\frac{du}{dx} - u\frac{dv}{dx}}{v^2}$$

We have to remove as many powers of r as necessary from under the summation in order to make the argument look like a derivative of a power function; therefore, we rewrite equation (58) as:

$$\bar{n} = r(1-r)\sum_{n=1}^{\infty} nr^{n-1} \qquad (59)$$

We can, therefore, rewrite equation (59) as:

$$\bar{n} = r(1-r)\sum_{n=1}^{\infty}\frac{d}{dr}r^n = r(1-r)\frac{d}{dr}\left(\sum_{n=1}^{\infty} r^n\right)$$

The term under the summation is now a geometric series and according to equation (50) we can write:

$$\bar{n} = r(1-r)\frac{d}{dr}\left(\frac{r}{1-r}\right) = r(1-r)\frac{1}{(1-r)^2}$$

$$\therefore \quad \bar{n} = \frac{r}{1-r} \qquad (60)$$

If the multiplier of the geometric series is more complex, such as *(n+1)n*, you have to use second derivatives or other techniques known as transforms.

Chapter 9. Introduction to Queueing Theory

A queueing system is one in which some commodity flows, moves, or is transferred through one or more channels in order to go from one point to another.

A queueing system can be represented by Figure 71, which provides a graphical view of its dynamics and also provides the details of the underlying stochastic processes. The diagram shown is for a single server to make it simpler, but it can be easily generalized.

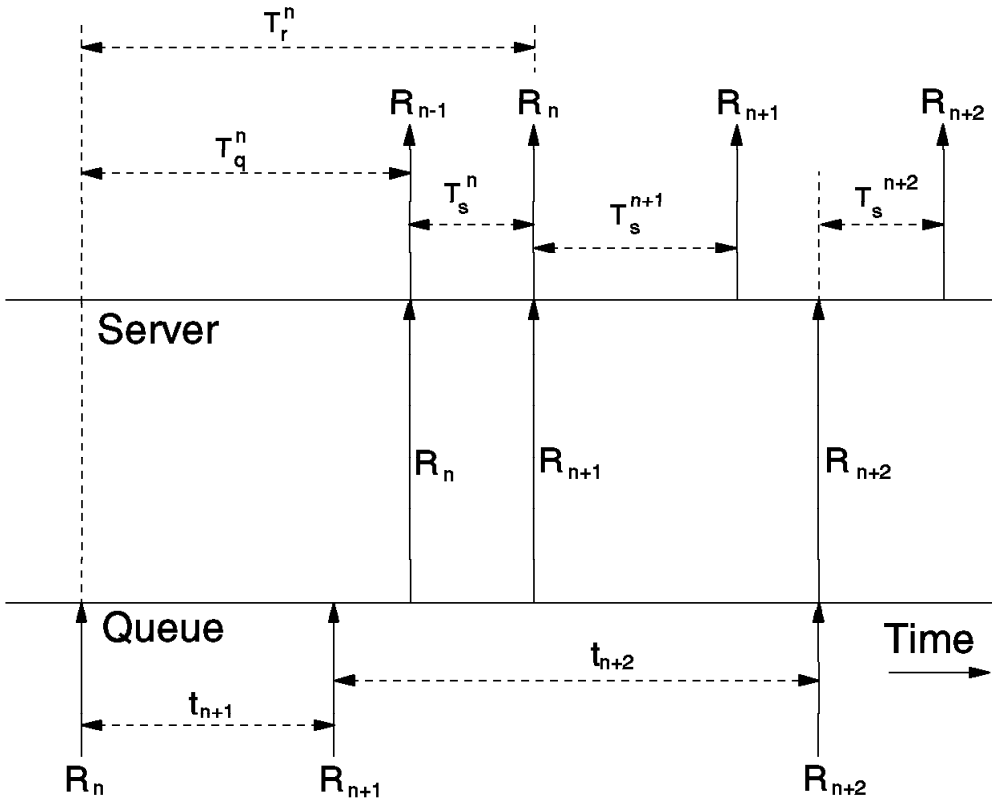

Figure 71. Queueing Systems Time Diagram

An arrow approaching the queue or server line from below indicates that an arrival has occurred; arrows emanating from the server line indicate departures. In this diagram we see that the element R_{n+1} arrives before the element R_n enters service; only when R_n departs from service may R_{n+1} enter service and, of course, these two events occur simultaneously.

The queue time (T_q^n), the service time (T_s^n), and the response time (T_r^n) for element R_n is shown; the inter-arrival times t^{n+1} and t^{n+2} are also shown.

Note that when R_{n+2} arrives, it finds the system empty and so immediately proceeds through an empty queue directly into the server facility.

Notation

Queueing systems can be classified in several ways depending on inter-arrival and service-time distributions, size of waiting room and population, and queue discipline.

The notation suggested by D. G. Kendall is now widely used to describe queueing models. In this notation, a model is generically described as A/B/c/K/M/Z, where:

A Inter-arrival distribution, represented by:

- **M** Exponential
- **D** Deterministic (constant or steady flow)
- **G** General (arbitrary)
- **E_r** r-stage Erlangian
- **H_R** R-stage hyperexponential

B Service-time distribution (same as inter-arrival distribution)

c Number of servers (1, 2, 3, ... ¥)

K Size of the waiting (storage) room; if finite, incoming elements are not allowed into the system (are lost) when the capacity is reached.

M Population; if finite, the arrival rate is variable.

Z Queue discipline, which may be:

- **FIFO** First in, first out
- **LIFO** Last in, first out
- **SIRO** Service in random order
- **PRI** Service with priorities

The first three elements of the notation are always mentioned; the others may take the default ¥/¥/FIFO. This chapter analyzes the following models:

- M/M/1

  This is the simplest of the interesting models; it consists of one server with Poisson arrivals, exponentially distributed service times, unlimited storage room, and constant arrival rate; the main random variables are studied in detail.

- M/M/c

  A multiple-server system, otherwise identical to M/M/1; its most important result, known as the Erlang-C formula, is derived.

- Other models

  The models M/M/1/K, M/M/1//M, and M/M/c//M are presented as examples of limited storage and limited population.

- M/G/1

  This is a single-server system with Poisson arrivals, arbitrary service time distribution, unlimited waiting room, and constant arrival rate. The mean values for the main random variables are presented.

Some examples of other models are D/D/1, M/M/¥, G/M/c, and G/G/1.

The following notation is generally applicable to all queueing models:

λ	Number of arrivals per second (I/O rate, for example)
μ	Number of services completed per second (server capacity)
ρ	Utilization factor
u	Traffic intensity ($\frac{\lambda}{\mu}$)
N_q	Number of elements in queue
N_s	Number of elements being serviced
N_t	Number of elements in the system
T_q	Time spent by one element in the queue (queue time)
T_s	Time spent by one element in service (service time)
T_r	Time spent by one element in the system (response time)

Note: A bar over a symbol indicates the average of the random variable.

The following relations are consequences of this notation:

$$N_t = N_q + N_s$$

$$T_r = T_q + T_s$$

Utilization Factor and Traffic Intensity

The utilization factor ρ is the ratio of the rate at which work enters the system to the maximum rate (capacity) at which the system can perform work. In single-server systems, the utilization factor is equal to the traffic intensity; that is:

$$\rho = u = \frac{\lambda}{\mu} \tag{61}$$

If the system is capable of servicing μ elements per second, each element is serviced in $1/\mu$ seconds, on the average, and ρ becomes:

$$\rho = \lambda \overline{T}_s \tag{62}$$

You can interpret this by saying that each arriving element brings into the system an amount of work equal to $\overline{T}_s$ seconds. The total work brought into the system in one second, is, therefore, $\lambda \overline{T}_s$, but the server is capable of a maximum of one second of work for each elapsed second of time.

In case of multiple servers, say *c* servers, the traffic intensity is equal to $c\rho$ because the servers are capable of *c* seconds of work for each second of elapsed time, and ρ is the average server utilization:

$$\rho = \frac{u}{c} = \frac{\lambda}{c\mu} = \frac{\lambda \overline{T}_s}{c} \tag{63}$$

Little's Law

One of the most general and useful theorems in queueing theory, known as Little's Law, Little's Theorem, or Little's Formula, was first established in a formal way by J. D. C. Little in 1961. It states that the average number of elements in a queueing system is equal to the average arrival rate of elements multiplied by the average time spent in the system.

Referring to Figure 72, let us position ourselves at the input of the queueing and count how many elements enter the system as a function of time (arrivals):

$a(t) \circ$ number of arrivals in (0,t)

Alternatively, we may position ourselves at the output of the queueing system and count the number of elements that leave the system (departures):

$d(t) \circ$ number of departures in (0,t)

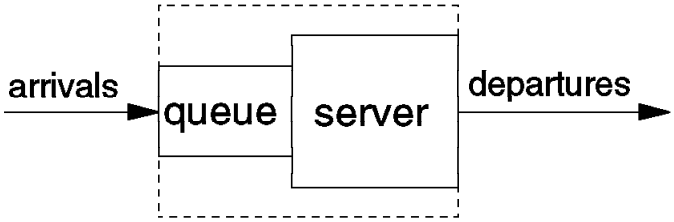

Figure 72. General Queueing System

Sample functions for these two stochastic processes are shown in Figure 73, where the top curve represents the arrivals and the bottom curve indicates the departures. Clearly, the number of elements in the system at a certain time t must be given by:

$$N(t) = a(t) - d(t)$$

On the other hand, the total area between these two curves up to some point t, represents the total time all elements have spent in the system, measured in units of element-seconds, during the interval (0,t). Let us denote this cumulative area by g(t).

Moreover, let l_t be defined as the average arrival rate (elements per second) during the interval (0,t); that is:

$$l_t = \frac{a(t)}{t} \qquad (64)$$

We may define T_t as the system time per element averaged over all elements in the interval (0,t). Since g(t) represents the accumulated element-seconds up to time t:

$$T_t = \frac{g(t)}{a(t)} \qquad (65)$$

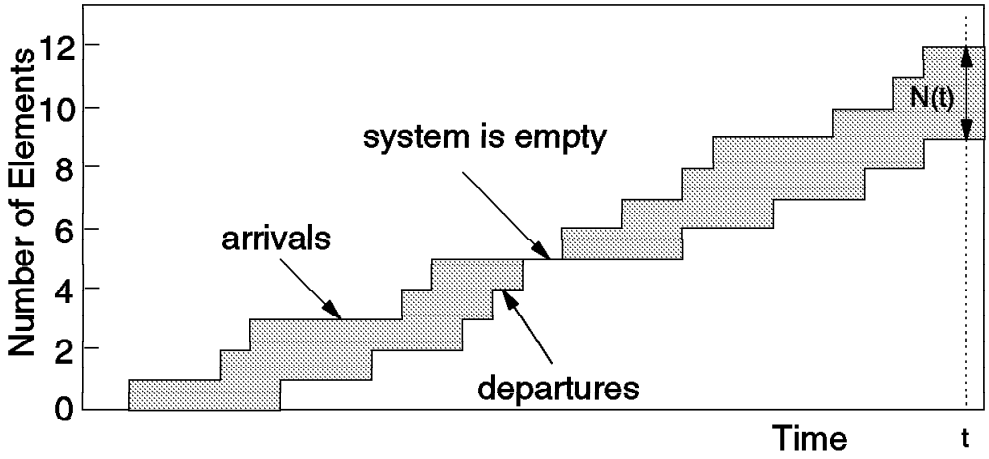

Figure 73. Arrivals and Departures

Lastly, let us calculate the average number of elements in the system during the interval (0,t) by dividing the accumulated number of element-seconds by the total interval length t:

$$N_t = \frac{g(t)}{t} \tag{66}$$

Combining equations (64), (65), and (66), we can write:

$$N_t = \frac{a(t)}{t} T_t = \lambda_t T_t$$

Let us assume that the queueing system is such that there is an equilibrium solution; that is, there is a solution when enough time has elapsed for the system to settle down: Then, if we use λ for the average arrival rate and T for the average system time, we can write:

$$N = \lim_{t \to \infty} N_t = \lim_{t \to \infty} \lambda_t T_t \quad \backslash \quad N = \lambda T \tag{67}$$

This result does not depend upon any specific assumptions regarding the arrival distribution or the service time distribution; nor does it depend upon the number of servers in the system or upon the particular queueing discipline; therefore, the result is applicable to any queueing system.

The broken-line box in Figure 72 on page 184 could apply to the queue, to the server, or to the entire system composed of queue and server; therefore, *N* can be the average number of elements in the queue, being serviced, or in the system; and *T* can be the queue time, the service time, or the response time, respectively.

An intuitive understanding of Little's Theorem is based on the fact that if there are λ arrivals per second and the server capacity is higher than the arrival rate, then there are also λ departures per second.

Utilization and Single-server Systems

Let us consider a general single-server system, G/G/1, and define t as an arbitrarily long time interval; the number of arrivals is very nearly equal to:

$$N_{arrivals} \approx \lambda t \tag{68}$$

Let us define P_0 as the probability that the server is idle at some randomly selected time. We may, therefore, estimate P_0 by dividing the interval of time the server is idle by the total interval of time:

$$P_0 = \frac{t_{idle}}{t}$$

Now, we are able to express the idle time and busy time as functions of the interval time and P_0:

$$t_{idle} = tP_0$$

$$t_{busy} = t - tP_0 = t(1 - P_0)$$

Assuming each element is serviced in $\overline{T}_s$ seconds, the number of departures during the interval t is very nearly equal to:

$$N_{departures} \approx \frac{t_{busy}}{\overline{T}_s} = \frac{t(1 - P_0)}{\overline{T}_s} \tag{69}$$

The system reaches equilibrium when the interval is long enough, and then, the number of arrivals are equal to the number of departures. Equating (68) and (69) we can write:

$$\lambda t \approx \frac{t(1 - P_0)}{\overline{T}_s}$$

$$\therefore \lambda = \lim_{t \to \infty} \frac{t(1 - P_0)}{t\overline{T}_s} = \frac{1 - P_0}{\overline{T}_s}$$

$$\therefore \lambda \overline{T}_s = 1 - P_0$$

Using equation (62) we can then write:

$$r = 1 - P_0 \circ P[\{\text{server busy}\}] \tag{70}$$

Therefore, in any single-server system (G/G/1), the server utilization is the probability that the server is busy. This heuristic result is formally derived for M/M/1 systems in “M/M/1 Model” on page 199.

Stochastic Processes

A stochastic process is a set of random variables indexed by the time parameter. The DASD I/O subsystem, for example, may be modelled as a stochastic process; some of its time-dependent random variables are:

- Number of elements in the system
- Number of elements in queue
- Number of elements in service
- Response time
- Queue time
- Service time
- Idle time
- Busy time
- Number of elements serviced during busy time

The classification of a random process depends upon three quantities: the state space, the index (time) parameter, and the statistical dependencies among the random variables X(t) for different values of the index parameter t.

The set of possible values (or states) that X(t) may take on is called its state space, which may be discrete or continuous. The permitted times at which changes in states may take place may also be classified as discrete or continuous.

The truly distinguishing feature of a stochastic process is the relationship among the random variables of the group. According to this classification, a stochastic process can be classified as:

- Stationary
- Independent
- Markov
- Birth-death
- Semi-Markov
- Renewal
- Random walk

Some of these processes overlap; as, for example, the renewal and the birth-death processes; some are subsets of other processes; as, for example, the renewal process and its superset random walk. The semi-Markov process is the most general process and is a superset of all others.

A set of random variables form a Markov process if the probability that a change to the next state depends only upon the current state and not upon any previous states. In other words, the way the entire history affects the future of the process is completely summarized in the current state of the process. Therefore, the distribution of time that a process may remain in a given state does not depend on the amount of time that the process is already in that state.

This restriction forces the state time to be exponentially distributed in case of continuous time, and geometrically distributed for discrete time transitions.

An important particular case of Markov processes is known as the birth-death process, in which state transitions take place between only neighboring states.

The elementary queueing models, such as M/M/1 and M/M/c, are based on the birth-death process; therefore, we will study this process in detail.

Birth-Death Processes

Birth-death processes are of particular interest in queueing theory as the elementary queueing systems can be modelled as birth-death processes.

The process is said to be in state E_n when the population at that time is of size *n*. A transition from E_n to E_{n+1} denotes a birth within the population, while a transition from state E_n to E_{n-1} denotes a death. In queueing theory notation, births correspond to arrivals and deaths correspond to departures or service completions.

As shown in Figure 74, we consider changes in the size of the population where transitions from state E_n take place to nearest neighbors only.

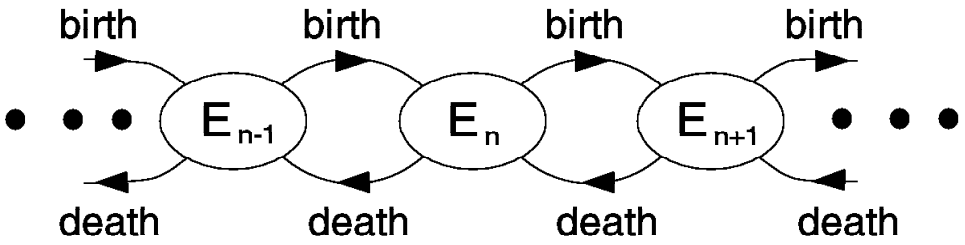

Figure 74. General State Transition Diagram

Regarding the nature of births and deaths, let us introduce the notion of a birth rate λ_n, which describes the rate at which births occurs when the population is of size *n*. Similarly, we define μ_n, which is the rate at which deaths occur when the population is of size *n*.

With these assumptions, we can define the following probabilities, considering an infinitesimal time interval *dt* ($dt \to 0$):

- The probability of one birth in the interval *(t,t+dt)* when in state *n* is $\lambda_n dt$.
- The probability of one death in the interval *(t,t+dt)* when in state *n* is $\mu_n dt$.
- The probability of zero death in the interval *(t,t+dt)* when in state *n* is $1 - \lambda_n dt$.
- The probability of zero death in the interval *(t,t+dt)* when in state *n* is $1 - \mu_n dt$.

From these definitions, you can see that multiple births, multiple deaths, or both a birth and a death in the infinitesimal interval *dt* are prohibited in the sense that each such multiple event is in the order of $(dt)^2$, and therefore, negligible.

We can express the dynamics of this system noticing that it can be found in the state E_n at time *t+dt* in case of occurrence of one of the three following mutually exclusive events:

- There are *n* elements in the population at time *t* and no state change occurs.
- There are *(n-1)* elements in the population at time *t* and one birth occurs in the interval *(t,t+dt)*.
- There are *(n+1)* elements in the population at time *t* and one death occurs in the interval *(t,t+dt)*.

Setting up the System

These three cases are portrayed in Figure 75; the probability for the first case is merely the probability $P_n(t)$ that we were at state E_n at time *t* multiplied by the probability that we stayed at the same state during the time interval.

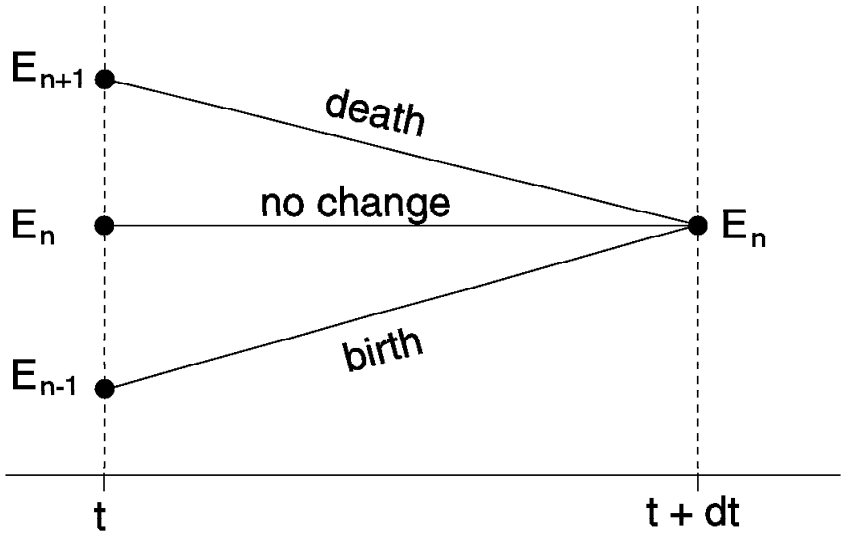

Figure 75. Possible Transitions into a General State

Therefore, the probability that there are *n* elements in the population at the moment *t+dt* (when dt ® 0) corresponds to the probability of the union of these three mutually exclusive events:

- The first event is given by the intersection of three independent events:

  $P_n(t)$ There are *n* elements in the population at the moment *t*

  $1 - \lambda_n dt$ No birth occurs during *dt*

  $1 - \mu_n dt$ No death occurs during *dt*

- The second event is given by the intersection of three independent events:

  $P_{n-1}(t)$ There are *(n-1)* elements in the population at the moment *t*

  $\lambda_{n-1} dt$ One birth occurs during *dt*

  $1 - \mu_{n-1} dt$ No death occurs during *dt*

- The third event is given by the intersection of three independent events:

  $P_{n+1}(t)$ There are *(n+1)* elements in the population at the moment *t*

  $1 - \lambda_{n+1} dt$ No birth occurs during *dt*

  $\mu_{n+1} dt$ One death occurs during *dt*

To calculate the probability of *n* elements in the population at the moment *t+dt*, we have to add these mutually exclusive events, remembering that the probability of each event is the multiplication of the probabilities of the three independent events; the corresponding equation is:

$$P_n(t + dt) = P_n(t)(1 - \lambda_n dt)(1 - \mu_n dt) + P_{n-1}(t)\lambda_{n-1} dt(1 - \mu_{n-1} dt) +$$
$$+ P_{n+1}(t)(1 - \lambda_{n+1} dt)\mu_{n+1} dt$$

Disregarding the terms involving $(dt)^2$, this equation reduces to:

$$\frac{P_n(t + dt) - P_n(t)}{dt} = \lambda_{n-1} P_{n-1}(t) + \mu_{n+1} P_{n+1}(t) - (\lambda_n + \mu_n) P_n(t)$$

The first term is the derivative of $P_n(t)$ and the result is the following differential equation:

$$\frac{dP_n(t)}{dt} = \lambda_{n-1} P_{n-1}(t) + \mu_{n+1} P_{n+1}(t) - (\lambda_n + \mu_n) P_n(t) \qquad n \geq 1 \qquad (71)$$

Obviously, equation (71) only makes sense when $n \geq 1$, since we cannot have a negative number of elements in the population. The probability that there are zero elements in the system at the moment *t+dt* can be calculated in the same manner; it corresponds to the probability of the union of two mutually exclusive events:

- The first event is given by the intersection of two independent events:

  $P_0(t)$ There are no elements in the population at the moment *t*

  $1 - \lambda_0 dt$ No birth occurs during *dt*

- The second event is given by the intersection of three independent events:

  $P_1(t)$ There is one element in the population at the moment *t*

  $1 - \lambda_1 dt$ No birth occurs during *dt*

  $\mu_1 dt$ One death occurs during *dt*

Adding the probabilities of these two mutually exclusive events we get the boundary condition:

$$P_0(t + dt) = P_0(t)(1 - \lambda_0 dt) + P_1(t)(1 - \lambda_0 dt)\mu_1 dt$$

Disregarding the terms involving $(dt)^2$, this equation leads to the following differential equation:

$$\frac{dP_0(t)}{dt} = \mu_1 P_1(t) - \lambda_0 P_0(t) \qquad n = 0 \qquad (72)$$

Finally, the probabilities must be conserved for all *t*, that is, the sum of the probabilities of all events in the space must be equal to one:

$$\sum_{n=0}^{\infty} P_n(t) = 1 \qquad (73)$$

The solution of the system of equations (71), (72), and (73) provides the probability mass function (pmf) for the birth-death process; that is, the probability that the population is of size *n* at a certain time *t*.

Using an Inspection Technique

Another way to develop the birth-death equations is by using an inspection technique combined with a state-transition-rate diagram similar to the one shown in Figure 74 on page 188. It is a simpler technique, but not as rigorous as the previous one.

In a state-transition-rate diagram, any state is represented by an oval surrounding the state number, as shown in Figure 76. The arrows indicate the permitted transitions and the labels give the infinitesimal rates at which these transitions take place; that is, the labels refer to arrival and departure rates and not to probabilities.

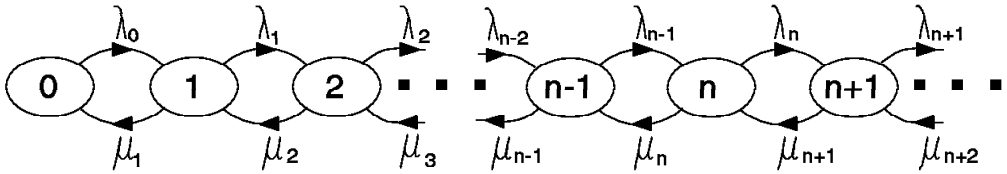

Figure 76. General State Transition Rate Diagram

If you multiply each transition label by the quantity *dt* you obtain the probability of such a transition occurring in the next interval of time whose duration is *dt*. In that case it is also necessary to insert self-loops on each state indicating the probability that the system remains in the given state in the interval of time *dt*.

If you consider state E_n and the special state E_0, you can observe that the rate at which probability flows to these states at time *t* is:

$$\text{Flow rate into } E_n = \lambda_{n-1}P_{n-1}(t) + \mu_{n+1}P_{n+1}(t) \qquad n \geq 1$$

$$\text{Flow rate into } E_0 = \mu_1 P_1(t) \qquad n = 0$$

Similarly, the flow rate out of those states at time *t* is:

$$\text{Flow rate out of } E_n = (\lambda_n + \mu_n)P_n(t) \qquad n \geq 1$$

$$\text{Flow rate out of } E_0 = \lambda_0 P_0(t) \qquad n = 0$$

The inspection technique consists of understanding that the effective probability flow rate into any state is the difference between the flow into and out of that state; therefore, the equations for a general state are:

$$\frac{dP_n(t)}{dt} = \lambda_{n-1}P_{n-1}(t) + \mu_{n+1}P_{n+1}(t) - (\lambda_n + \mu_n)P_n(t) \qquad n \geq 1 \tag{74}$$

$$\frac{dP_0(t)}{dt} = -\lambda_0 P_0(t) + \mu_1 P_1(t) \qquad n = 0$$

Therefore, this simple inspection technique provides the same system of equations given by equations (71) and (72).

Solving the System

We are interested in a general equilibrium solution as opposed to a time-dependent transient solution, and therefore, we have to solve the system of differential equations (74) and evaluate:

$$P_n = \lim_{t \to \infty} P_n(t) \tag{75}$$

There are two ways to accomplish this:

- Finding the general transient solution using Laplace- and z-transforms, and calculating the equilibrium solution by evaluating equation (75).
- Assume beforehand that the limit exists, making the derivatives over time go to zero, and reducing the differential equations to simple algebraic equations.

In both cases, P_n is interpreted as the limiting probability that the system contains *n* elements after settling down (warming up).

It is important to understand that, while P_n is no longer a function of time, it does not mean that the process does not move from state to state in this limiting case. Certainly, the number of elements in the system changes with time, but the long-run probability of finding the system with *n* elements is properly described by P_n, which is independent of time.

Using the second approach and noticing that the probabilities are now time-independent, we can reduce the system of equations (74) to:

$$\begin{aligned} \lambda_{n-1} P_{n-1} + \mu_{n+1} P_{n+1} - (\lambda_n + \mu_n) P_n &= 0 \qquad n \geq 1 \\ \mu_1 P_1 - \lambda_0 P_0 &= 0 \qquad n = 0 \end{aligned} \tag{76}$$

The annoying task of providing a separate equation for *n=0* may be overcome by agreeing once and for all that some birth and death coefficients are identically equal to zero, as well as the probabilities of having a negative number of elements in the population:

$$\begin{aligned} \lambda_{-1} = \lambda_{-2} = \lambda_{-3} = \ldots = 0 \\ \mu_0 = \mu_{-1} = \mu_{-2} = \ldots = 0 \\ P_{-1} = P_{-2} = P_{-3} = \ldots = 0 \end{aligned}$$

The negative-indexed coefficients should be zero because it does not make sense to define negative states; μ_0 should also be zero because there cannot be a death when the population is zero.

Thus, for all values of *n* we may reformulate the system of equations (76) into one single equation:

$$\lambda_{n-1} P_{n-1} + \mu_{n+1} P_{n+1} - (\lambda_n + \mu_n) P_n = 0 \tag{77}$$

One way to solve equation (77) is to rewrite it as follows:

$$\mu_{n+1} P_{n+1} - \lambda_n P_n = \mu_n P_n - \lambda_{n-1} P_{n-1} \tag{78}$$

Let us define a function g_n such that:

$$g_n = \mu_{n+1} P_{n+1} - \lambda_n P_n \quad (79)$$

Therefore, g_n represents the difference of flow between two adjacent states. Examining equations (78) and (79) we can write:

$$g_n = g_{n-1} \quad (80)$$

Clearly, equation (80) implies that g_n is constant with respect to *n* since it is the same for all *n*. The interpretation here is that the difference of flow between two adjacent states is the same for any state.

In particular, for *n=1*:

$$g_1 = g_0$$

Making *n=0* in equation (79) we get:

$$g_0 = \mu_1 P_1 - \lambda_0 P_0$$

And then, the second equation of the system (76) enables us to write:

$$g_0 = 0$$

The interpretation here is that, in equilibrium, the difference of flow between two adjacent states is the same for any state and it is zero.

Since all g_n are equal:

$$g_n = \mu_{n+1} P_{n+1} - \lambda_n P_n = 0$$

$$\therefore \quad P_{n+1} = \frac{\lambda_n}{\mu_{n+1}} P_n \quad (81)$$

Therefore, to find the general expression for the pmf for any queueing theory model based on birth-death processes, we have to solve the following system of equations, given by equations (73) and (81):

$$\begin{cases} P_{n+1} = \dfrac{\lambda_n}{\mu_{n+1}} P_n \\ \sum_{n=0}^{\infty} P_n = 1 \end{cases} \quad (82)$$

Since the summation involves an infinite series, the necessary conditions for the series to converge must be established in each case. The specific assumptions are introduced in system (82) for each model to determine the corresponding equations, as shown in the next sections.

Induction Argument

Another way to solve the system of equations (76) is through induction. Although not as elegant and physically related to the actual system dynamics, the induction argument is quite simple.

We start by evaluating an equation for P_1 using the second equation of system (76):

$$P_1 = \frac{\lambda_0}{\mu_1} P_0 \tag{83}$$

Then, we make an assumption that the equation for P_n is similar to equation (83):

$$P_n = \frac{\lambda_{n-1}}{\mu_n} P_{n-1} \tag{84}$$

The next step is using equation (84) in the first equation of system (76) and see whether a similar expression for P_{n+1} can be derived from there:

$$\lambda_{n-1} P_{n-1} + \mu_{n+1} P_{n+1} - (\lambda_n + \mu_n) P_n = 0$$

$$\therefore \quad \mu_{n+1} P_{n+1} = \mu_n P_n + \lambda_n P_n - \lambda_{n-1} P_{n-1} \tag{85}$$

Replacing the first occurrence of P_n in equation (85) by its assumed value established in equation (84), we have:

$$\begin{aligned} \mu_{n+1} P_{n+1} &= \mu_n \frac{\lambda_{n-1}}{\mu_n} P_{n-1} + \lambda_n P_n - \lambda_{n-1} P_{n-1} \\ &= \lambda_{n-1} P_{n-1} + \lambda_n P_n - \lambda_{n-1} P_{n-1} \\ &= \lambda_n P_n \end{aligned} \tag{86}$$

And finally:

$$P_{n+1} = \frac{\lambda_n}{\mu_{n+1}} P_n \tag{87}$$

Thus, we have:

- An expression for P_1
- Assumed an equivalent expression for P_n
- Proved that an equivalent expression for P_{n+1} is also valid

Therefore, we can validate the expression for P_2 because we know that the expression for P_1 is valid, and therefore, validate the expression for P_3, and keep using the same argument for any value of *n*.

This is a valid proof, but as you can see, you have to guess what the general expression is before proving it is correct.

Poisson and Exponential Distributions

The two most important probability distributions for queueing models are the Poisson distribution and the exponential distribution. Their importance is due to the fact that many real-life situations in fact do obey the appropriate requirements.

This section presents their distribution function, the corresponding mean and variance, and their properties.

Poisson Distribution

The Poisson distribution was defined by S. Poisson and proved to be an approximation of the binomial distribution when the number of trials is large and the probability of success is small. A discrete random variable *X* is a Poisson random variable with parameter $a > 0$ when its probability mass function (pmf) is given by:

$$p(x) = \frac{a^x}{x!} e^{-a}$$

The moments of the Poisson distribution can be determined easily with transforms; the mean and variance are:

$$E[X] = a$$

$$s^2 = a$$

Exponential Distribution

A continuous random variable *X* is exponentially distributed with parameter $a > 0$ when its probability density function (pdf) is given by:

$$f(x) = \begin{cases} a e^{-ax} & x > 0 \\ 0 & x \le 0 \end{cases}$$

The distribution function (PDF) is given by:

$$F(x) = \int_0^x a e^{-ax} = -e^{-ax} \Big|_0^x = 1 - e^{-ax}$$

The moments of the exponential distribution can be determined easily with transforms; the mean, variance, and coefficient of variation are:

$$E[X] = \frac{1}{a}$$

$$s^2 = \frac{1}{a^2}$$

$$c^2 = 1$$

Properties

Let us consider a random sequence of points (or events) scattered along the time axis *t*, satisfying the following properties:

- The position of each point does not depend on the other points.
- The number of points found up to a certain time t_0 in the *t* axis does not affect the probabilities of finding one or more points in an interval counted after t_0 (infinite universe).
- The probability of finding one point in an infinitesimal interval *dt*, measured after *t*, is equal to λdt, where λ is a positive constant; the probability of finding more than one point is negligible.

According to these properties, the probability of finding *n* points in a time interval of length *t* is given by the union of two events:

- *n* arrivals in *t* and zero in *dt*
- *n-1* arrivals in *t* and one in *dt*

Therefore, we can write:

$$\begin{aligned} P_n(t+dt) &= P_n(t)(1-\lambda dt) + P_{n-1}(t)\lambda dt \qquad n>0 \\ P_0(t+dt) &= P_0(t)(1-\lambda dt) \qquad n=0 \end{aligned} \tag{88}$$

We can rewrite equations (88) as a system of differential equations:

$$\begin{aligned} \frac{dP_0(t)}{dt} &= -\lambda P_0(t) \qquad n=0 \\ \frac{dP_n(t)}{dt} &= -\lambda P_n(t) + \lambda P_{n-1}(t) \qquad n>0 \end{aligned} \tag{89}$$

When we solve the system (89), we find the following solutions:

$$P_0(t) = e^{-\lambda t}$$

$$P_1(t) = (\lambda t)e^{-\lambda t}$$

$$P_2(t) = \frac{(\lambda t)^2}{2} e^{-\lambda t}$$

$$P_3(t) = \frac{(\lambda t)^3}{3!} e^{-\lambda t}$$

...

We can conjecture, and prove by induction, that the general formula is:

$$P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \tag{90}$$

which is the Poisson distribution with parameter λ.

You can see that the assumptions made for this process are quite similar to those made for birth-death processes. In fact, if you make $l_i = l$ and all $m_i = 0$ in system (74), you will find the same system of equations (89);

Therefore, the Poisson process is a pure birth process with constant birth rate that is independent of the size of the population; that is, all arrival coefficients are equal to l, which is called the Poisson parameter.

The Poisson process has the following properties:

- Additive property

  The sum of *n* independent Poisson processes with parameter l_i (*i=1,2,...,n*) is a Poisson process with parameter $l = l_1 + l_2 + \cdots l_n$

- Decomposition property

  Suppose that *N(t)* is the number of occurrences of a specified event and that $\{N(t), t \geq 0\}$ is a Poisson process with parameter l. Suppose further that each occurrence of the event has a probability *p* of being recorded, and that the recording of an occurrence is independent of other occurrences and also of *N(t)*. If $M_1(t)$ is the number of occurrences so recorded, then $\{M_1(t), t \geq 0\}$ is also a Poisson process with parameter lp. If $M_2(t)$ is the number of occurrences not recorded, then $\{M_2(t), t \geq 0\}$ is also a Poisson process with parameter $l(1 - p)$, and $M_1(t)$ is independent of $M_2(t)$.

  In other words, a random selection of a Poisson process yields a Poisson process.

- Inter-arrival times

  Let us consider the random variable t_a as the inter-arrival time and call *A(t)* its probability distribution function (PDF). Since the definition for *A(t)* is merely the probability that the time between arrivals is less than or equal to *t*, it must clearly be given by:

  $$A(t) = 1 - P[t_a > t]$$

  But $P[t_a > t]$ is just the probability that no arrivals occur in *(0,t)*, that is, it is equal to $P_0(t)$; making *n=0* in equation (90) we can write:

  $$A(t) = 1 - e^{-lt} \qquad (91)$$

  which is the exponential distribution with parameter l (average $1/l$).

  Therefore, when the birth rate (arrival rate) is a Poisson process with mean l arrivals per second, the continuous random variable that represents the time between arrivals (inter-arrival time) is exponentially distributed with mean of $1/l$ seconds; similarly, if the inter-arrival time is exponentially distributed, the arrival process is a Poisson process.

  Using the same reasoning, if the death rate (service-completion rate) is a Poisson process with a mean of m completions per second, the time between service completions (service time) is an exponentially distributed random variable with an average of $1/m$ seconds.

- PASTA property

  In a queueing system, you can measure two averages for the number of elements: one average over time and another calculated at arrival points (or any other well-defined point). In general, there is no reason to expect these averages to be the same. Think of a D/D/1 queue in which the arrivals are

regular, say once an hour on the hour, and the service times are always 30 minutes. The expected number of elements in the system (over time) is 0.5 while the expected number of elements in the system at arrival points is zero. These two averages are equal only for Poisson processes; this property is called PASTA (Poisson Arrivals See Time Averages).

- Lack-of-memory property

  Assume an exponentially distributed server with a mean of $1/m$ seconds:

  - The probability of servicing one element in more than t_s seconds is:

    $$P[t > t_s] = 1 - (1 - e^{-mt_s}) = e^{-mt_s} \quad (92)$$

  - The probability of servicing one element in more than t_s seconds considering that the element is already being serviced for t_0 seconds is (by definition of conditional probability):

    $$P[t > t_0 + t_s \mid t > t_0] = \frac{P[(t > t_0 + t_s) \cap (t > t_0)]}{P[t > t_0]} =$$

    $$= \frac{P[t > t_0 + t_s]}{P[t > t_0]} = \frac{1 - (1 - e^{-m(t_0 + t_s)})}{1 - (1 - e^{-mt_0})}$$

    $$\therefore \quad P[t > t_0 + t_s \mid t > t_0] = e^{-mt_s} \quad (93)$$

    Equations (92) and (93) show that the probability that an element currently in service is completed at some future time *t* is independent of how long the element has already been in service. In other words, the distribution of the remaining time, given that a certain time has elapsed since the start of service, is identical to the unconditional distribution.

    Since the history of the random variable plays no role in predicting its future, we say that it is memoryless, or has a lack-of-memory property, also known as Markovian or non-aging property. The exponential distribution is the only continuous distribution to possess this property.

- Randomness property

  If an interval of time *(0,t)* contains exactly one occurrence of a Poisson process, then the random variable describing the time of occurrence of this Poisson event has a continuous uniform distribution on the interval *(0,t)*; that is, any sub-interval of *(0,t)* of length d has probability d/t of containing the time of occurrence of the event. Therefore, we usually say that an event of a Poisson process is a purely random event.

Several physical, organic, economic, and social phenomena are Poisson processes. In particular, the number of customers arriving at a service location such as a gas station, information counter, public telephone, or toll station, is a Poisson process, assuming unbiased conditions.

In many cases, the sum of a large number of independent arbitrary processes tends to a Poisson process; this explains why Poisson processes appear so often in nature when the aggregate effect of a large number of elements is under consideration. Therefore, the Poisson process is used to model a large class of stochastic phenomena and is extremely useful in the analysis of queueing systems.

M/M/1 Model

The M/M/1 model is the simplest nontrivial interesting system with Poisson arrivals and exponential service times. We also assume one server and infinite queueing; therefore, the state-transition-rate diagram is as shown in Figure 77.

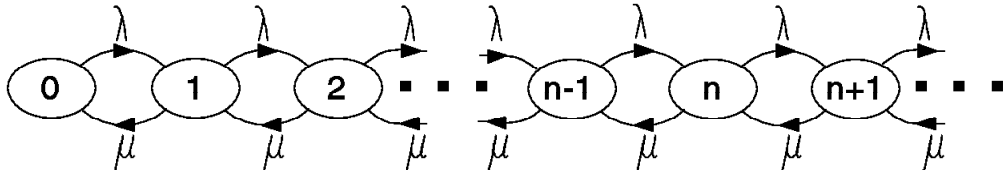

Figure 77. State Transition Rate Diagram for M/M/1

We can model these conditions by selecting the birth-death coefficients as:

$$\begin{aligned} \lambda_n &= \lambda \qquad n \geq 0 \\ \mu_n &= \mu \qquad n \geq 1 \end{aligned}$$

Applying these coefficients to equation (81) and using equation (61) on page 183 for the traffic intensity ($r = \lambda/\mu$), we can write:

$$\begin{aligned} P_1 &= \frac{\lambda}{\mu} P_0 = \rho P_0 \\ P_2 &= \frac{\lambda}{\mu} \rho P_0 = \rho^2 P_0 \\ &\ldots \\ P_n &= \frac{\lambda}{\mu} \rho^{n-1} P_0 = \rho^n P_0 \end{aligned} \tag{94}$$

In order to determine P_0 we use the probability conservation equation from system (82) noticing that the expression for P_n is a geometric series and it converges for $r < 1$, as established in equation (50) on page 174:

$$\sum_{n=0}^{\infty} P_0 r^n = \frac{P_0}{1-r} = 1$$

$$\therefore \quad P_0 = 1 - r \tag{95}$$

Compare equation (95) with equation (70); the formal derivation of this property for the M/M/1 system agrees with the heuristic result we found in “Utilization Factor and Traffic Intensity” on page 183.

Combining equations (94) and (95), we finally arrive at the probability mass function (pmf) for the number of elements in the system:

$$P_n = (1-r)r^n \tag{96}$$

Number of Elements

Using the pmf established in equation (96), we are able to determine the average number of elements in queue, in service, and in the system, as explained in "Mean" on page 169.

Number of Elements in the System

The pmf is the same as in equation (53) discussed in "RPS-miss Time" on page 178; therefore, the average number of elements in the system is given by the same equation (60).

$$\overline{N}_t = \sum_{n=1}^{\infty} nP_n = (1-r)\sum_{n=1}^{\infty} nr^n \qquad (97)$$

$$\therefore \quad \overline{N}_t = \frac{r}{1-r} \qquad (98)$$

Number of Elements in Service

The average number of elements in service can be calculated by noticing that the M/M/1 model has only one server, and therefore:

$N_s = 0$ when there are no elements in the system ($n = 0$)

$N_s = 1$ when there is at least one element in the system ($n \geq 1$)

Therefore, we can write:

$$\overline{N}_s = 0 \cdot P_0 + \sum_{n=1}^{\infty} 1 \cdot P_n = \sum_{n=1}^{\infty} P_n \qquad (99)$$

We can change the starting index in equation (99) to zero, include P_0 in the summation, and then subtract P_0 from the equation:

$$\overline{N}_s = \sum_{n=0}^{\infty} (P_n) - P_0 = 1 - P_0 \qquad (100)$$

Since we did not make any restriction so far, equation (100) is valid for any G/G/1 system; using equation (95), which is also valid for any G/G/1 system:

$$\overline{N}_s = 1 - (1 - r)$$

$$\therefore \quad \overline{N}_s = r \qquad (101)$$

The average number of elements in service is a measure of the server utilization, which is a consequence of the definition of r. This result confirms, for the M/M/1 model, our discussion in "Utilization Factor and Traffic Intensity" on page 183 that in any single-server system the server utilization is the probability that the server is busy.

Number of Elements in Queue

The average number of elements in the queue can be calculated by noticing that:

$N_q = 0$ when there are no elements in the system

$N_q = 0$ when there is one element in the system

$N_q = 1$ when there are two elements in the system

$N_q = 2$ when there are three elements in the system

$\vdots$

$N_q = (n - 1)$ when there are *n* elements in the system

Therefore, we can write:

$$\overline{N}_q = 0 \cdot P_0 + \sum_{n=1}^{\infty} (n - 1) P_n$$

$$\therefore \quad \overline{N}_q = \sum_{n=1}^{\infty} n P_n - \sum_{n=1}^{\infty} P_n \qquad (102)$$

However, equation (102) is the same as the difference between equations (97) and (99) representing the average number of elements in the system and in service, respectively. Replacing the corresponding values, we can write:

$$\overline{N}_q = \overline{N}_t - \overline{N}_s$$

This confirms the system behavior, already defined in “Notation” on page 182, establishing that the total number of elements in the system is equal to the sum of the number of elements in the queue plus the number of elements in service.

Using the values already calculated for $\overline{N}_t$ and $\overline{N}_s$ in equations (98) and (101):

$$\overline{N}_q = \frac{r}{1 - r} - r$$

$$\therefore \quad \overline{N}_q = \frac{r^2}{1 - r} \qquad (103)$$

Equations (98) and (103) show that the system is instable for $r = 1$ because $r < 1$ is the condition for convergence; this effect is further discussed in “Response Time” on page 205.

Average Size of Nonempty Queues

We might also be interested in the average queue size of nonempty queues, which we denote by $\overline{N}_w$. In other words, we want to ignore the cases where the queue is empty and calculate the size of the queue when there is a queue.

Before doing this, let us evaluate the expression for the probability that the number of elements in the system is less than or equal to a certain specified constant; that is, we want to evaluate the PDF for this random variable, as discussed in "Discrete Random Variables" on page 163.

$$P_{n \le i} = \sum_{n=0}^{i} (1-r)r^n = (1-r)\frac{r^{i+1}-1}{r-1}$$

$$\therefore \quad P_{n \le i} = 1 - r^{i+1} \tag{104}$$

Similarly, you can evaluate:

$$P_{n \ge i} = r^i \tag{105}$$

Note that the events represented by equations (104) and (105) are not mutually exclusive; they have the event $P_{n=i}$ in common.

The conditional probability distribution of *n* elements in the system, given that the queue is not empty is:

$$P_w \equiv P[\{n \text{ in system}\} \mid \{n \ge 2\}]$$

Using equation (105) and the laws of conditional probability, we can write:

$$P_w = \frac{P[\{n \text{ in system}\} \cap \{n \ge 2\}]}{P_{n \ge 2}} = \frac{P_n}{r^2} \qquad n \ge 2$$

Using the same reasoning as in "Number of Elements in Queue" on page 201:

$$\overline{N}_w = \sum_{n=2}^{\infty} \frac{(n-1)P_n}{r^2} = \frac{1}{r^2}\left(\sum_{n=2}^{\infty} nP_n - \sum_{n=2}^{\infty} P_n\right)$$

$$\overline{N}_w = \frac{(\overline{N}_t - 1 \cdot P_1) - (1 - P_1 - P_0)}{r^2}$$

$$\therefore \quad \overline{N}_w = \frac{1}{1-r} \tag{106}$$

System Time Distributions

To obtain information concerning the time an arriving element spends in the system, we have to derive the corresponding probability distribution function (PDF).

Up to now, we did not consider the queue discipline in our derivations; however, when considering individual elements, the queue discipline must be specified; in this discussion, we are assuming the queue discipline is FIFO (First-in First-out).

Let us use $W_s(t)$, $W_q(t)$, and $W(t)$ for the PDF of the service time, queue time, and response time, respectively. Because of higher level of mathematics necessary to develop it, we just present the final results here:

The service time is exponentially distributed by definition, and according to equation (91):

$$W_s(t) = 1 - e^{-\mu t} \tag{107}$$

The queue time random variable is, for most of part, a continuous random variable except that there is a nonzero probability that the element finds the system empty and does not have to wait; that is, the element enters service immediately after arriving:

$$\begin{aligned} W_q(0) &= 1 - \rho \qquad t = 0 \\ W_q(t) &= 1 - \rho e^{-(\mu - \lambda)t} \qquad t > 0 \end{aligned} \tag{108}$$

The distribution for the response time is:

$$W(t) = 1 - e^{-(\mu - \lambda)t} \qquad t > 0 \tag{109}$$

Remember that a PDF gives the probability that the random variable is less of equal the argument; that is, for the queue time:

$$W_q(t_0) \equiv P[T_q \leq t_0]$$

Output Process

The input process of an M/M/1 system is Poisson, by definition. The output process depends whether the M/M/1 system is already in steady state.

In an M/M/1 queueing system in steady state, the inter-departure time is also an exponentially distributed random variable with mean $1/\lambda$, where λ is the parameter of the input Poisson process. In other words, the output process is also Poisson with the same parameter as the input Poisson process.

The condition $\lambda < \mu$ is essential for the existence of a steady state, as discussed before. However, the output process is still Poisson (if a large amount time is allowed to elapse) even when $\lambda \geq \mu$ and a steady state does not hold. Therefore, the M/M/1 output process is always Poisson, for a large elapsed time, with parameter $\min(\lambda, \mu)$.

Average System Times

We can calculate the service time, queue time, and response time by evaluating the pdf for each distribution and evaluating the mean for each case, as discussed in “Mean” on page 169. The pdf can be easily calculated as the derivative of the PDFs established in equations (107), (108), and (109).

A simpler way is using equation (67) established in “Little's Law” on page 184, which is valid for any queueing model.

Service Time

Using Little's Law:

$$\overline{T}_s = \frac{\overline{N}_s}{l} = \frac{r}{l} \qquad (110)$$

$$\therefore \quad \overline{T}_s = \frac{1}{m}$$

This result is valid for any model because the server performs an average of m services per unit of time.

Queue Time

Using Little's Law:

$$\overline{T}_q = \frac{\overline{N}_q}{l} = \frac{r^2}{(1-r)l} = \frac{r^2}{(1-r)rm} \qquad (111)$$

$$\therefore \quad \overline{T}_q = \frac{r}{1-r}\overline{T}_s$$

Equation (111) is the celebrated M/M/1 equation for the average queue time.

Queue Time when there is a Queue

This value, called $\overline{T}_w$, is the average time that an element has to wait when a queue is formed; in other words, we ignore those cases when the system is empty. The average queue time can be expressed as the weighted value between these two conditions:

$$0 \cdot P_0 + \overline{T}_w \cdot P_{n \geq 1} = \overline{T}_q$$

$$\therefore \quad \overline{T}_w = \frac{\overline{T}_s}{1-r} \qquad (112)$$

This is the same as the response time (equation (113) in the next section); therefore, elements that must wait will wait one average service time longer than the average element waits.

This value is particular important for queues consisting of people, because people tend to leave the queue after waiting a certain time, depending on their level of impatience.

Response Time

Response time is calculated by adding the queue time and the service time:

$$\overline{T}_r = \overline{T}_s + \overline{T}_q = \overline{T}_s + \frac{r}{1 - r}\overline{T}_s \qquad (113)$$

$$\therefore \quad \overline{T}_r = \frac{\overline{T}_s}{1 - r}$$

Equation (113), which is plotted in Figure 78, shows that the system is instable for $r = 1$; this is not surprising if you recall that $r < 1$ was our condition for convergence of the geometric series involved in the calculations.

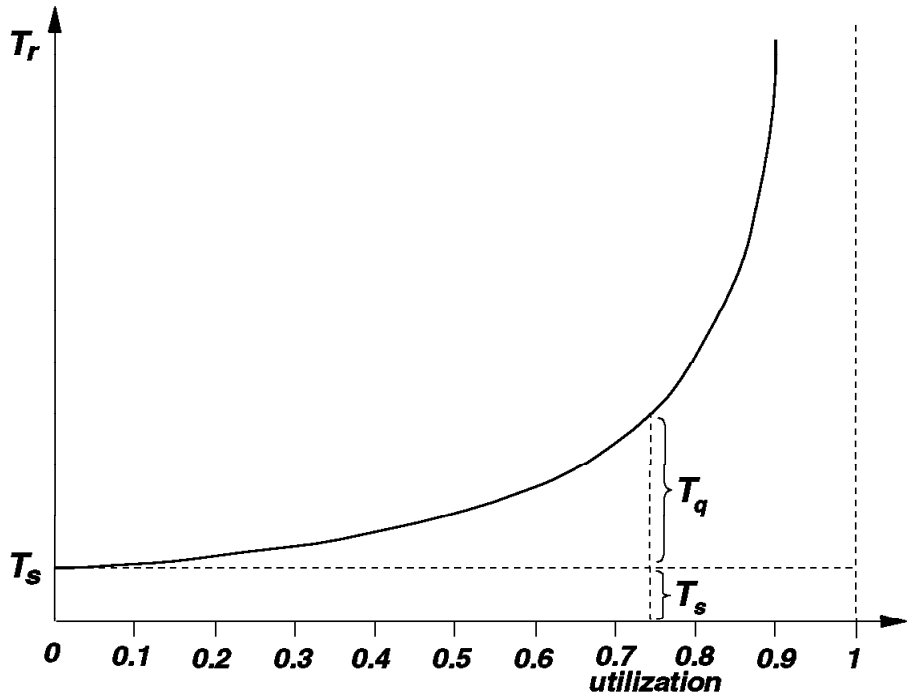

Figure 78. M/M/1 Response Time

What is perhaps surprising is that the behavior of the average response time deteriorates so badly as $r \to 1$. On M/M/1 queueing systems, there is an extreme penalty when you attempt to run the system near its capacity. In fact, this type of behavior is a characteristic of almost all queueing systems.

The intuitive explanation for this behavior is that with random flow there are occasional bursts of traffic that temporarily overwhelm the server. While it is still true that the server is idle on average during $P_0 = 1 - r$ of the time, as established by equation (95), this average idle time is not distributed evenly within small time intervals, but is true only in the long run.

In a steady flow model (D/D/1), the idle time is distributed quite evenly: after every service time, which is always exactly $1/m$ seconds, there is an idle time of exactly $(1/l - 1/m)$ seconds. It is the variability in both the arrival rate and the service time that gives rise to the disastrous behavior as the system nears capacity.

Combined M/M/1 Queues

Consider that two identical M/M/1 queueing systems (S1 and S2) with the same rates λ and μ are in operation side by side, with independent queues. The distribution (pmf) of the number of elements in each system is given by equation (96) on page 199:

$$P_n = (1 - r)r^n$$

The probability of having *n* elements in the combined system (P_n^2) is the union of the following mutually exclusive events (each pair consists of independent events):

- No elements in S1 and *n* elements in S2
- One element in S1 and *(n-1)* elements in S2
- Two elements in S1 and *(n-2)* elements in S2

  ⋮
- *n* elements in S1 and no elements in S2

$$P_n^2 = P_0P_n + P_1P_{n-1} + \cdots + P_{n-1}P_1 + P_nP_0$$

$$= (1 - r)(1 - r)r^n + (1 - r)r(1 - r)r^{n-1} + \cdots + \quad (114)$$

$$+ (1 - r)r^{n-1}(1 - r)r + (1 - r)r^n(1 - r)$$

$$- r)^2r^n + (1 - r)^2r^n + \cdots + (1 - r)^2r^n$$

Since there are n + 1 terms in equation (114) (from 0 to n), we can write:

$$P_n^2 = (n + 1)(1 - r)^2r^n \quad (115)$$

Applying the same reasoning, the pmf for three independent M/M/1 systems can be evaluated as:

$$P_n^3 = \frac{(n + 1)(n + 2)}{2}(1 - r)^3r^n \quad (116)$$

You can use equation (115) and (116) to calculate the average number of elements in service, in queue, and in the system; however, these averages of the combined system are the sum of the average of each system. Therefore, the average number of elements in queue in *c* M/M/1 systems is:

$$\overline{N}_q^c = \frac{cr^2}{1 - r} \quad (117)$$

The average system times are the same as for the single M/M/1 system because the average number of elements and the average arrival rate are multiplied by the same factor, cancelling each other when you apply Little's Law.

These results are valid when the M/M/1 systems have independent queues; that is, an element must choose one system at random (without regard to the queue length in front of each system) and must not switch queues.

M/M/1 Example 1

This example analyzes the case of people arriving at a public telephone according to a Poisson distribution; the average interval between arrivals (inter-arrival time) is 10 minutes and the duration of the calls is exponentially distributed with an average of three minutes.

According to the definition of the parameters:

- The rate of arrivals (λ) is the inverse of the inter-arrival time.
- The rate of services (μ) is the inverse of the service time:

Therefore, for this example, the parameters are:

$$\lambda = \frac{1}{10}$$

$$\mu = \frac{1}{3}$$

$$r = \frac{\lambda}{\mu} = \frac{\frac{1}{10}}{\frac{1}{3}} = 0.3$$

- What is the probability that a person has to wait to make a call?

  This is clearly the probability of the telephone being used by another person:

  $$P_{n>0} = r = 0.3 \circ 30\%$$

- What is the percentage of time the telephone will be in use ?

  This is the probability of the telephone being busy, which is equal to its utilization:

  $$\%\ \text{time busy} = 1 - P_0 = r = 30\%$$

- What is the average number of people in the queue?

  $$\overline{N}_q = \frac{r^2}{1-r} = \frac{(0.3)^2}{0.7} = 0.13$$

- What is the average waiting time?

  $$\overline{T}_q = \frac{r}{(1-r)\mu} = \frac{0.3}{0.7 \times \frac{1}{3}} = 1.3 \text{ minutes}$$

- What is the average time to complete one call?

  This is the response time:

  $$\overline{T}_r = \frac{1}{(1-r)\mu} = \frac{\overline{T}_q}{r} = \frac{1.3}{0.3} = 4.3 \text{ minutes}$$

- What is the probability of having two or more persons in the system?

  According to equation (105):

  $$P_{n \geq 2} = r^2 = 0.09 \approx 9\%$$

- What is the probability that a person has to wait more than two minutes to make his call?

  We have to use the queue time distribution, equation (108), described in "System Time Distributions" on page 203:

  $$P[T_q > t_0] = 1 - P[T_q \leq t_0] = 1 - W_q(t_0) = re^{-(m-l)t_0}$$

  $$\therefore \quad P[T_q > 2] = 0.3e^{-(\frac{1}{3} - \frac{1}{10})2} = 0.188$$

  There is a probability of 18.8% that a person has to wait more than two minutes to make a call.

- What is the probability that a person has to wait between two and five minutes to make his call?

  In this case we use a property of the PDF, described by equation (40) in "Continuous Random Variables" on page 166:

  $$P[2 \leq T_q \leq 5] = W_q(5) - W_q(2)$$

  $$= [1 - 0.3e^{-(\frac{1}{3} - \frac{1}{10})5}] - [1 - 0.3e^{-(\frac{1}{3} - \frac{1}{10})2}]$$

  $$= 0.3e^{-(\frac{1}{3} - \frac{1}{10})2} - 0.3e^{-(\frac{1}{3} - \frac{1}{10})5}$$

  $$= 0.188 - 0.093 = 0.095$$

  There is a probability of 9.5% that a person has to wait between 2 and 5 minutes to make a call.

- The company is willing to install another telephone in that spot when the average waiting time (queue time) is three minutes or more. What should be the limiting average interval between arrivals (inter-arrival time) ?

  $$\overline{T}_q = \frac{r}{1 - r}\overline{T}_s = \frac{r}{(1 - r)m} = \frac{\frac{l}{m}}{(1 - \frac{l}{m})m} = \frac{l}{m(m - l)}$$

  Solving for l:

  $$l = \frac{m^2\overline{T}_q}{1 + m\overline{T}_q} = \frac{(\frac{1}{3})^2 \times 3}{1 + \frac{1}{3} \times 3} = \frac{1}{6}$$

  The company should install another telephone when the arrival rate increases to 1/6 persons per minute (10 persons per hour); that is, when the average time between arrivals reaches six minutes.

M/M/1 Example 2

A company wants to hire a mechanic to maintain a large number of machines that break down in a Poisson process at an average of three per hour, causing a loss of $150.00 per machine per hour.

The company will select one of two mechanics, Jack or Joe, both fixing machines with exponentially distributed times:

- Jack charges $30.00 per hour of work and is able to fix four machines per hour on average.
- Joe charges $300.00 per hour of work and is able to fix six machines per hour on average.

Which one should be hired?

The queueing system under consideration consists of broken machines; therefore, the loss due to these broken machines is the average number of elements (machines) in the system multiplied by the loss caused by each machine, plus the cost of fixing them.

The cost due to machine breakage (C_b) is:

$$C_b = 150\overline{N}_t = 150\frac{r}{1 - r}$$

Since $r = \frac{l}{m}$, we can write:

$$C_b = 150\frac{\frac{l}{m}}{1 - \frac{l}{m}} = \frac{150l}{m - l}$$

$$C_b = \frac{150 \cdot 3}{m - 3} = \frac{450}{m - 3}$$

Therefore, Jack's cost per hour is:

$$C_{Jack} = 30 + \frac{450}{4 - 3} = 480$$

Joe's cost per hour is:

$$C_{Joe} = 300 + \frac{450}{6 - 3} = 450$$

Therefore, although not intuitive, Joe should be hired.

If you plot the cost due to machine breakage (C_b) as a function of m (the rate of fixing machines), you can easily see that the cost goes down sharply from $m = 3$ to $m = 6$, but it does not change as much after $m = 6$.

Busy and Idle Periods

In a single-server system, an arrival element is immediately taken into service if the server is free, but joins a queue if the server is busy. The system can be considered to be in two states, idle or busy, depending on whether the server is idle or busy.

A busy period starts as soon as an element arrives at a free server and ends at the instant the server next becomes free for the first time. An idle period starts when the server becomes free and ends at the instant of the next arrival (residual inter-arrival time).

If we denote $E[I]$ as the average amount of time the server is idle and $E[B]$ as the average amount of time the server is busy, the long-run proportion of time that the server is idle and busy is then:

$$P_0 = \frac{E[I]}{E[I] + E[B]} \qquad 1 - P_0 = \frac{E[B]}{E[I] + E[B]}$$

$$\therefore \quad \frac{E[B]}{E[I]} = \frac{1 - P_0}{P_0} = \frac{r}{1 - r} \tag{118}$$

Equations (118) are valid for any single-server system; in particular, if the arrival process is Poisson with arrival rate λ, then it follows from its lack-of-memory property that an idle period is exponentially distributed with a mean of $1/\lambda$, and for any M/G/1 system we can write:

$$E[I] = \frac{1}{\lambda}$$

$$E[B] = \frac{r \quad 1}{1 - r \ (\ \lambda\ )} = \frac{\overline{T}_s}{1 - r} \tag{119}$$

Let $E[N]$ denote the average number of elements served during a busy period. For every $E[N]$ arrivals during a busy period, only the first arriving element finds the system empty, with probability $a_0 = 1/E[N]$. Since Poisson arrivals have the PASTA property (see “Poisson and Exponential Distributions” on page 195):

$$a_0 = \frac{1}{E[N]} = P_0 = 1 - r$$

$$\therefore \quad E[N] = \frac{1}{1 - r} \tag{120}$$

You can get the same result by considering that the server remains continuously busy during a busy period; that is, the number of services completed is the rate of service multiplied by the busy period:

$$E[N] = E[B]m = \frac{\overline{T}_s}{1 - r^{m}} = \frac{1}{1 - r} \tag{121}$$

M/M/c Model

This is a more complex model consisting of multiple servers with Poisson arrivals, exponentially distributed service times, unlimited storage room, and a constant arrival rate. Considering that we have *c* servers, and the birth-death coefficients are:

$$\mu_n = n\mu \qquad n \leq c$$
$$\mu_n = c\mu \qquad n \geq c$$

The probability mass function (pmf) P_n must be separated into two parts, since the dependence of μ upon *n* is in two parts. Therefore, we have to calculate one pmf that is applicable when the number of elements is less than the number of servers, and another pmf applicable when the number of elements is equal to or greater than the number of servers.

Probability Mass Function for n < c

When n < c there is an increasing rate of service completions as we move from state to state, as shown in Figure 79.

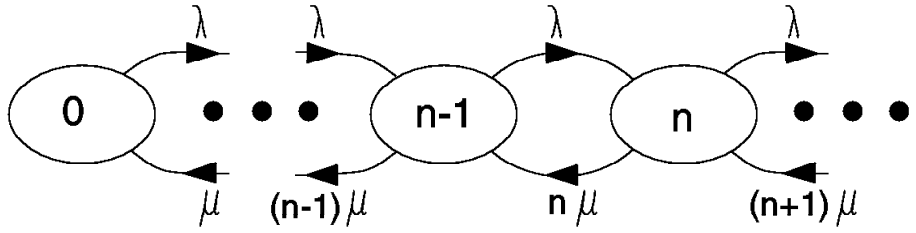

Figure 79. State Transition Rate Diagram for n < c

Applying these coefficients to equation (81) and using the definition of the traffic intensity ($u = \lambda/\mu$), we can write:

$$P_1 = \frac{\lambda}{\mu} P_0 = uP_0$$

$$P_2 = \frac{\lambda}{2\mu} uP_0 = \frac{u^2}{2} P_0$$

$$P_3 = \frac{\lambda}{3\mu} \frac{u^2}{2} P_0 = \frac{u^3}{3 \cdot 2}$$

$$P_4 = \frac{\lambda}{4\mu} \frac{u^3}{3 \cdot 2} P_0 = \frac{u^4}{4 \cdot 3 \cdot 2}$$

...

$$P_n = \frac{u^n}{n!} P_0 \qquad n \leq c \qquad (122)$$

Probability Mass Function for n ³ c

Each state has an increasing rate of service completions up to the point where n = c where the rate reaches saturation, as shown in Figure 80.

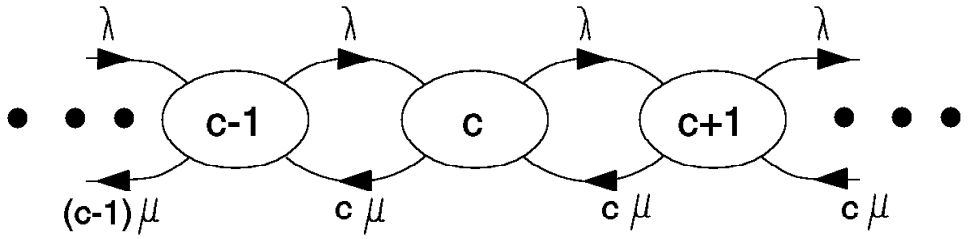

Figure 80. State Transition Rate Diagram for n ³ c

According to our convention, state number *c* represents the state where the number of elements is equal to the number of servers. Therefore, c - 1 is the last state of the group of states with variable service-completion coefficients.

Applying these coefficients to equation (81) and using the definition of the traffic intensity (u = l/m), we can write:

$$P_c = \frac{l}{c\mu} P_{c-1} = \frac{l}{c\mu} \frac{u^{c-1}}{(c-1)!} P_0 = \frac{u^c}{c!} P_0$$

$$P_{c+1} = \frac{l}{c\mu} P_c = \frac{l}{c\mu} \frac{u^c}{c!} P_0 = \frac{u^{c+1}}{c!\, c} P_0$$

$$P_{c+2} = \frac{l}{c\mu} P_{m+1} = \frac{l}{c\mu} \frac{u^{c+1}}{c!\, c} P_0 = \frac{u^{c+2}}{c!\, c^2} P_0$$

...

$$P_{c+i} = \frac{l}{c\mu} P_{m+i-1} = \frac{l}{c\mu} \frac{u^{c+i-1}}{c!\, c^i} P_0 = \frac{u^{c+i}}{c!\, c^i} P_0$$

In order to write the pmf as an expression of a general state number, let us make n = c + i, and use the definition u = cr:

$$P_n = \frac{u^n}{c!\, c^{n-c}} P_0$$

$$P_n = \frac{(cr)^n}{c!\, c^{n-c}} P_0 = \frac{r^n c^n}{c!\, c^{n-c}} P_0$$

$$\therefore \quad P_n = \frac{r^n c^c}{c!} P_0 \qquad n \geq c \tag{123}$$

Erlang-C Formula

Equations (122) and (123) represent the pmf for the M/M/c queue. However, since both equations are expressed in terms of P_0, we have to determine this last unknown constant.

As always, in order to determine P_0 we use the probability conservation equation from system (82). Since we have two pmfs that are applicable to different ranges of *n*, we separate the summation for each range and use the functions established in equations (122) and (123):

$$\sum_{n=0}^{\infty} P_n = \sum_{n=0}^{c-1} \frac{u^n}{n!} P_0 + \sum_{n=c}^{\infty} \frac{r^n c^c}{c!} P_0 = 1 \tag{124}$$

$$\therefore \quad P_0 = \frac{1}{\sum_{n=0}^{c-1} \left(\frac{u^n}{n!} \right) + \sum_{n=c}^{\infty} \frac{c^c}{c!} r^n}$$

The second summation is a geometric series that converge when $r < 1$, or when $u < c$. The first term of the series is r^c and the common ratio is r; therefore, we can write:

$$P_0 = \frac{1}{\sum_{n=0}^{c-1} \left(\frac{u^n}{n!} \right) + \frac{c^c}{c!} \frac{r^c}{1-r}}$$

$$\therefore \quad P_0 = \frac{1}{\sum_{n=0}^{c-1} \frac{u^n}{n!} + \frac{u^c}{c!(1-r)}} \tag{125}$$

We saw in “Utilization Factor and Traffic Intensity” on page 183 that, in any single-server system, the probability that the server is busy is equal to the server utilization.

A multiple-server system is busy when all servers are busy, which is equal to the probability of having at least *c* elements in the system. The utilization factor r represents the percentage of busy servers, as discussed in “Number of Elements in Service” on page 215. On the other hand, the system is free when the number of elements in the system is less than the number of servers ($n \leq c - 1$), because one arriving element can be serviced without queueing.

If you look at equation (124) you can observe that the two terms represent the probability of system free and system busy, respectively.

Therefore, the probability of system busy is either the second term of equation (124) or the complement of the first term; it is easier to use the second term, because it has already been evaluated before for P_0.

$$P_{n \geq c} = \sum_{n=c}^{\infty} \frac{r^n c^c}{c!} P_0 = \frac{u^c}{c!\,(1-r)} P_0 \qquad (126)$$

Using equation (122) or equation (123), the probability of having exactly *c* elements in the system is:

$$P_c = \frac{u^c}{c!} P_0 \qquad (127)$$

We can simplify equation (126) using equation (127):

$$P_{n \geq c} = \frac{P_c}{1-r} \qquad (128)$$

However, it is useful to express equation (126) in terms of the traffic intensity and number of servers. Using equation (125) and replacing the calculated value for P_0, we can write:

$$P_{n \geq c} = \frac{\dfrac{u^c}{c!\,(1-r)}}{\dfrac{u^c}{c!\,(1-r)} + \sum_{n=0}^{c-1} \dfrac{u^n}{n!}}$$

We can simplify it, multiplying each term by $(1-r)$:

$$P_{n \geq c} = \frac{\dfrac{u^c}{c!}}{\dfrac{u^c}{c!} + (1-r)\sum_{n=0}^{c-1} \dfrac{u^n}{n!}} \qquad (129)$$

Equation (129) represents the probability that an arriving element has to wait because the system is busy; this equation is known as Erlang-C formula in USA, and is also denoted by C(c,u). In Europe, it is known as Erlang's second formula, and is denoted by $E_{2,c}(u)$.

You can use equation (129) to verify equations (9), (10), and (11), recalling that r is the average channel utilization.

The calculations involved in the Erlang-C formula are quite tedious and error prone, particularly for $c > 2$ since you have to evaluate finite sums term by term. Tables are available in some books, but a small REXX program can easily provide these values.

Number of Elements in Service

Using the same reasoning as in M/M/1 model, the average number of elements in service is:

- $N_s = n$ when there are n elements in the system and $n < c$
- $N_s = c$ when there are n elements in the system and $n \geq c$

$$\overline{N}_s = \sum_{n=0}^{c-1} nP_n + \sum_{n=c}^{\infty} cP_n = \left[\sum_{n=0}^{c-1} \left(\frac{nu^n}{n!} \right) + \frac{cu^c}{c!(1-r)} \right] P_0$$

$$= u \left[\sum_{n=1}^{c-1} \left(\frac{u^{n-1}}{(n-1)!} \right) + \frac{u^{c-1}}{(c-1)!(1-r)} \right] P_0$$

$$= u \left[\sum_{n=0}^{c-2} \left(\frac{u^n}{n!} \right) + \frac{\{(1-r)+r\}u^{c-1}}{(c-1)!(1-r)} \right] P_0$$

$$= u \left[\sum_{n=0}^{c-2} \left(\frac{u^n}{n!} \right) + \frac{u^{c-1}}{(c-1)!} + \frac{ru^{c-1}}{(c-1)!(1-r)} \right] P_0$$

$$= u \left[\sum_{n=0}^{c-1} \left(\frac{u^n}{n!} \right) + \frac{u^c}{c!(1-r)} \right] P_0$$

$$= u \left[P_0 \right]^{-1} P_0$$

$$\therefore \quad \overline{N}_s = u \tag{130}$$

The expected (average) number of idle servers is then:

$$E[\{\text{idle servers}\}] = c - u = c(1-r)$$

Since $r = u/c$, r has one of the following interpretations:

- Percentage of busy servers
- Percentage of time each server is busy
- Average utilization of servers
- Average number of busy servers.
- Probability that any server is busy

Number of Elements in Queue

In an M/M/c system, we have:

- $N_q = 0$ when $n \leq c$
- $N_q = (n - c)$ when $n > c$

$$\overline{N}_q = \sum_{n=c}^{\infty} (n - c) P_n$$

$$= \sum_{n=c}^{\infty} (n - c) \frac{u^n}{c! c^{n-c}} P_0$$

$$= \frac{P_0 u^c}{c!} \sum_{n=c}^{\infty} (n - c) \frac{u^{n-c}}{c^{n-c}}$$

$$= \frac{P_0 u^c}{c!} \sum_{n=c}^{\infty} (n - c) r^{n-c}$$

Let us change the index of the summation making $m = n - c$:

$$\overline{N}_q = \frac{P_0 u^c}{c!} \sum_{m=0}^{\infty} m r^m \qquad (131)$$

The summation in equation (131) is similar to equation (58), and we can use the same result found in “Using Derivatives” on page 180:

$$\overline{N}_q = \frac{P_0 u^c}{c!} \frac{r}{(1 - r)^2}$$

$$= \frac{r u^c}{c! (1 - r)^2} P_0$$

Using equation (126), we can write:

$$\overline{N}_q = \frac{r}{1 - r} P_{n \geq c} \qquad (132)$$

Equation (132) reduces to equation (103) when the number of servers is equal to one since $P_{n \geq 1} = r$ for an M/M/1 system.

Number of Elements in the System

The number of elements in the system can be calculated merely by adding equations (130) and (132):

$$\overline{N}_t = \overline{N}_s + \overline{N}_q$$

Using equations (130) and (132), we can write:

$$\overline{N}_t = u + \frac{r}{1 - r} P_{n \geq c} \tag{133}$$

System Times

In an M/M/c system, every server has an exponentially distributed service time with a mean service rate of μ. The average response time that an element can expect from this system is still the average time spend in the queue plus the average time spent in service.

If all servers are busy ($n \geq c$), the time between service completions is the minimum time taken by any server to complete one service. This random variable is also exponentially distributed with a mean service rate of $c\mu$. In this case, you can imagine the server facility performing like a single exponentially distributed server with a mean service rate of $c\mu$.

Service Time

$$\overline{T}_s = \frac{1}{\mu} \tag{134}$$

Queue Time

Using equations (63), (132), and Little's Law:

$$\overline{T}_q = \frac{r}{\lambda(1 - r)} P_{n \geq c} = \frac{\overline{T}_s}{c(1 - r)} P_{n \geq c} \tag{135}$$

Queue Time when there is a Queue

Similar to the M/M/1 model, you can weight the average queue time:

$$0 \cdot P_{n < c} + \overline{T}_w \cdot P_{n \geq c} = \overline{T}_q$$

Using equation (135):

$$\overline{T}_w = \frac{\overline{T}_s}{c(1 - r)} \tag{136}$$

Response Time

Adding equations (134) and (135), we can write:

$$\overline{T}_r = \overline{T}_s \left[1 + \frac{P_{n \geq c}}{c(1 - r)} \right] \tag{137}$$

One M/M/c Queue or c M/M/1 Queues?

Let us consider *c* number of M/M/1 queues, as discussed in "Combined M/M/1 Queues" on page 206, each with rates λ and μ, and a standard M/M/c queue with rates $c\lambda$ and μ. Which model, the M/M/c system or the combined M/M/1 systems, provides the shortest average queue time ?

Since the service capacity is the same for the M/M/c and for the combined M/M/1 queues, and the average arrival rate is the same for each server, the system with smallest queue time is the one with the shortest average queue size.

Let us rewrite equation (129), multiplying each term by c! and expanding the summation:

$$P_{n \geq c} = \frac{u^c}{u^c + c!(1-r)\left[1 + u + \frac{u^2}{2} + \frac{u^3}{3!} + \cdots + \frac{u^{c-1}}{(c-1)!}\right]} \qquad (138)$$

Assuming that $c \geq 2$, the summation in equation (138) has at least two terms; let us then replace all its terms, except the last one, by a positive constant, say *A*:

$$P_{n \geq c} = \frac{u^c}{u^c + c!(1-r)\left[A + \frac{u^{c-1}}{(c-1)!}\right]}$$

$$= \frac{u^c}{u^c + c!(1-r)A + c!\frac{u^{c-1}}{(c-1)!} - c!\frac{u}{c}\frac{u^{c-1}}{(c-1)!}} \qquad (139)$$

$$= \frac{u^c}{u^c + c!(1-r)A + cu^{c-1} - u^c}$$

$$= \frac{u^c}{cu^{c-1} + c!(1-r)A}$$

Dividing all terms of the last form of equation (139) by u^{c-1}, we can write:

$$P_{n \geq c} = \frac{u}{c + \frac{c!(1-r)A}{u^{c-1}}} \qquad (140)$$

Since $r < 1$ by definition, we can replace the second term of the denominator of equation (140) by another positive constant, say *B*:

$$P_{n \geq c} = \frac{u}{c + B} \qquad c \geq 2 \qquad (141)$$

Equation (141) shows that in all M/M/c queues ($c \geq 2$), $P_{n \geq c}$, which is the probability of queueing, is always less than r.

Let us now divide equation (117) on page 206 by equation (132), and call *mm1* and *mmc* the number of elements in queue for the M/M/1 systems and M/M/c system, respectively. Note that r is the same for all systems because the traffic intensity for each server is the same.

$$\frac{mm1}{mmc} = \frac{c_r^2 \quad 1 - r}{1 - rP_{n \geq c}}$$

$$\therefore \quad \frac{mm1}{mmc} = \frac{u}{P_{n \geq c}} \qquad (142)$$

Replacing $P_{n \geq c}$ in equation (142) by its evaluated expression in equation (141):

$$\frac{mm1}{mmc} = u\frac{c + B}{u} = c + B$$

$$\therefore \quad mm1 = (c + B)\, mmc \qquad c \geq 2 \qquad (143)$$

Equation (143) shows that the combined M/M/1 queue is at least twice the M/M/c queue; therefore, the M/M/c queue with a cl arrival rate is more efficient than *c* M/M/1 systems with a l arrival rate and independent queues.

Table 16 shows the expected queue size for different values of r and number of servers. The M/M/1 queue size shown is for one system, and has to be multiplied by *c* when comparing it with the corresponding M/M/c queue size. For example, for r = 30%, the combined M/M/1 queue size is 4.33 times (2*0.12857/0.05934) the M/M/2 queue size.

Table 16. M/M/c Against Combined M/M/1 Systems

Average Server Utilization	Queue Size			
	M/M/1	M/M/2	M/M/3	M/M/4
10	0.01111	0.00202	0.00041	0.00009
20	0.05000	0.01667	0.00616	0.00240
30	0.12857	0.05934	0.03001	0.01588
40	0.26667	0.15238	0.09412	0.06407
50	0.50000	0.33333	0.23684	0.17391
60	0.90000	0.67500	0.53212	0.43056
70	1.63333	1.34510	1.14880	1.00019
80	3.20000	2.84444	2.58876	2.38573
90	8.10000	7.67368	7.35355	7.08978

The value of the constant *B* in equation (143) decreases as r increases; therefore, the largest difference between the M/M/1 queue and the M/M/c queue is observed at low utilizations. This may be intuitively understood when you consider that the M/M/c queue optimizes the servers idle time; at low utilizations, there are more M/M/1 servers idle while queues are allowed to form at the other servers.

M/M/c Example 1

Consider the same example of people arriving at a public telephone analyzed for the M/M/1 queue on page 207; let's assume that the telephone company has installed a second telephone to cope with a halved inter-arrival time of five minutes; the duration of the calls is still exponentially distributed with the same average of three minutes.

- What is the probability that a person has to wait to make a call?

  This is the Erlang-C formula for an M/M/2 system:

$$u = \frac{1/5}{1/3} = 0.6 \qquad r = \frac{u}{2} = 0.3$$

$$\therefore \quad P_{n \geq 2} = \frac{2r^2}{1 + r} = \frac{2 \cdot (0.3)^2}{1 + 0.3} = 0.14 \approx 14\%$$

- What is the average number of people in the queue?

$$\overline{N}_q = \frac{r}{1 - r} P_{n \geq 2} = \frac{0.3}{0.7} \cdot 0.14 = 0.06$$

- What is the average waiting time?

$$\overline{T}_q = \frac{P_{n \geq 2}}{c_m(1 - r)} = \frac{0.14}{2 \cdot 1/3 \cdot 0.7} = 0.3 \text{ minutes}$$

- What is the average time to complete one call (response time)?

$$\overline{T}_r = \overline{T}_s + \overline{T}_q = 3.3 \text{ minutes}$$

- The company is willing to still install another telephone in that spot when the average waiting time (queue time) is three minutes or more. What should be the limiting average interval between arrivals (inter-arrival time)?

$$\overline{T}_q = \frac{r}{\lambda(1 - r)} P_{n \geq 2} = \frac{\frac{\lambda}{c_m}}{\lambda\left(1 - \frac{\lambda}{c_m}\right)} P_{n \geq 2}$$

$$\therefore \quad \overline{T}_q = \frac{P_{n \geq 2}}{c_m - \lambda}$$

  Solving for λ:

$$\lambda = \frac{c_m \overline{T}_q - (P_{n \geq 2})}{\overline{T}_q} = \frac{2 \cdot \frac{1}{3} \cdot 3 - 0.14}{3} = 0.62$$

  The company should install another telephone when the arrival rate increases to 0.62 persons per minute; that is, when the average time between arrivals reaches 1.6 minutes.

M/M/c Example 2

A 4-path storage control can handle a typical I/O request in 2.5 milliseconds, with the service time exponentially distributed. How many disks, following a Poisson process with an arrival rate of 15 I/O operations per second, can this storage control handle, assuming the following design criteria for the subsystem:

- The probability that an I/O request has to wait must not exceed 10%.
- The average waiting time for those requests that must wait should not exceed one millisecond

We can model this system as an M/M/4 system, varying the combined I/O arrival rate, and calculating the corresponding system values; the solution must meet the following translation of the design criteria into technical queueing theory notation:

$$P_{n \geq 4} \leq 0.1$$

$$\overline{T}_w \leq 1 \text{ millisecond}$$

The problem can be solved by the REXX program shown in Figure 81; the results are presented in Table 17.

```
/* Number of disks in storage control */
/* Arguments: c - number of channels  */

 arg c
 if c = '' then c = 4
 Ts  = 2.5  /* milliseconds */

 do lambda = 420 to 1440 by 60
    u  = lambda*Ts/1000
    A = 1
    do n = 1 to c-1
       A = A + u**n/fat(n)
    end n
    B  = u**c/(fat(c)*(1-u/c))
    Pnc = B/(A + B)
    Tq  = Ts*Pnc/(c-u)
    Tw  = Ts/(c-u)
    say format(lambda,4) format(A,2,4) format(B,2,4),
       format(Pnc,2,4)  format(Tq,2,4) format(Tw,2,4); say
 end lambda
exit

fat: procedure
 arg i
 r = 1
 do j = 1 to i
    r = r * j
 end j
return r
```

Figure 81. Number of Disk Devices

Table 17. Number of Disk Devices

I/O rate	$A = \sum_{n=0}^{3} u^n/n!$	$B = \frac{u^4}{4!\,(1 - u/4)}$	$P_{n\,3\,4} = \frac{B}{(A + B)}$	T_q	T_w
420	2.7942	0.0687	0.0240	0.0203	0.8475
480	3.2080	0.1234	0.0370	0.0331	0.8929
540	3.6713	0.2089	0.0538	0.0508	0.9434
600	**4.1875**	**0.3375**	**0.0746**	**0.0746**	**1.0000**
660	4.7599	0.5257	0.0995	0.1058	1.0638
720	5.3920	1.7953	0.1285	0.1461	1.1364
780	6.0871	1.1755	0.1619	0.1974	1.2195
840	6.8485	1.7060	0.1994	0.2624	1.3158
900	7.6797	2.4408	0.2412	0.3445	1.4286
960	8.5840	3.4560	0.2870	0.4485	1.5625
1020	9.5648	4.8601	0.3369	0.5809	1.7241
1080	10.626	6.8133	0.3907	0.7513	1.9231
1140	11.769	9.5616	0.4482	0.9745	2.1739
1200	13.000	13.500	0.5094	1.2736	2.5000
1260	14.321	19.305	0.5741	1.6886	2.9412
1320	15.735	28.236	0.6422	2.2934	3.5714
1380	17.245	42.930	0.7134	3.2428	4.5455
1449	18.856	69.984	0.7878	4.9235	6.2500

As you can see from Table 17, 600 I/O operations per second satisfies both design conditions. Since each disk contributes with 15 I/O operations per second, the number of disks on this storage control should be limited to 40.

However, in order to account for imbalances, you should reduce this number by a uniformity factor (let's use 0.8) resulting in a maximum of 32 devices in the storage control.

Table 17 also shows the impact on performance if the I/O rate was higher than anticipated: both T_q and T_w increase faster than the I/O rate.

The limiting I/O rate that would cause system instability corresponds to a traffic intensity equal to the number of servers:

$$C = \frac{\lambda}{m} \qquad \therefore \quad \lambda = Cm$$

$$\therefore \quad \lambda = \frac{4 \cdot 1000}{2.5} = 1600$$

Therefore, this M/M/4 I/O subsystem must have an I/O rate less than 1600 I/O operations per second; otherwise, the queue would grow without limit.

M/M/1/K Model

For the simple M/M/1 queue, the assumption is that the system can accommodate any number of elements. In the M/M/1/K model, we assume that the system can accommodate a finite number of elements, including the one being served, if any.

Elements arrive in accordance with a Poisson process with a rate of λ, but an element joins the system only when it finds fewer than *K* elements in the system; an element arriving when there are *K* elements in the system is refused entry and departs without service. Service time is exponential with a rate of μ.

Though the arrival process in the M/M/1/K system is Poisson, the input process for this truncated system is not truly Poisson. The reason is that an arriving customer cannot find the system in state *K*, since it would then be turned away.

Poisson arrivals have a unique property in that such arrivals behave like random arrivals; that is, an observer from a Poisson arrival stream finds the same system state distribution as a random observer. In other words, the expected number of elements in the system over time is the same as the expected number at arrival points. This unique property is called PASTA, as described in "Poisson and Exponential Distributions" on page 195.

In fact, whenever arrivals occur from a finite source or there is a limited waiting storage, the input to the service facility is called quasi-random input, but can still be handled by the birth-death model, which is a superset of the Poisson process. In this case, the probabilities must be normalized, and the concept of effective input rate must be introduced, as you will see later in this chapter.

The M/M/1/K system behaves as an ordinary M/M/1 queue so long as the number of elements in the system is fewer than *K*, as shown in Figure 82; however, from state *K*, only a departure is possible.

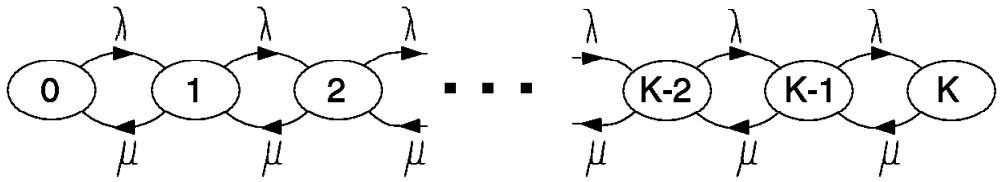

Figure 82. State Transition Rate Diagram for M/M/1/K

We can model these conditions appropriately with our birth-death process, choosing the following coefficients:

$$\lambda_n = \begin{cases} \lambda & n < K \\ 0 & n \geq K \end{cases}$$

$$\mu_n = \begin{cases} \mu & n \leq K \\ 0 & n > K \end{cases}$$

Applying these coefficients to equation (81) and using the definition $r = \lambda/\mu$, as established by equation (61) on page 183, we can write:

$$P_1 = \frac{\lambda}{\mu} P_0 = rP_0$$

$$P_2 = \frac{\lambda}{\mu} rP_0 = r^2 P_0$$

Therefore, the general equation for P_n is:

$$P_n = \begin{cases} \frac{\lambda}{\mu} r^{n-1} P_0 = r^n P_0 & 0 \le n \le K \\ 0 & n > K \end{cases}$$

In order to determine P_0, we use the probability conservation equation from the system of equations (82):

$$\sum_{n=0}^{K} P_0 r^n = 1 \qquad (145)$$

The summation in equation (145) is a geometric series when $r \neq 1$, and it is a straight sum of ones when $r = 1$; therefore, we can write:

$$\frac{P_0(r^{K+1} - 1)}{r - 1} = 1 \qquad r \neq 1$$

$$P_0(K+1) = 1 \qquad r = 1$$

$$\therefore \quad P_0 = \begin{cases} \frac{1 - r}{1 - r^{K+1}} & r \neq 1 \\ \frac{1}{K+1} & r = 1 \end{cases} \qquad (146)$$

Combining equations (144) and (146), we find the probability mass function (pmf) for the number of elements in the system:

$$P_n = \begin{cases} \frac{(1 - r)r^n}{1 - r^{K+1}} & r \neq 1 \\ \frac{1}{K+1} & r = 1 \end{cases} \qquad (147)$$

The finite summation in equation (145) has a value for all values of r; thus, the system is stable for all values of r. Assuming $r < 1$ and taking the limit as $K \to \infty$, we get the corresponding pmf for the M/M/1 model.

Effective Input Rate

Since some elements are refused entry to the system, it is interesting to calculate the rate of elements joining the system, or the effective input rate (l_a).

This input rate is calculated by multiplying the Poisson arrival rate (l) by the probability that the system is not full; that is, the number of elements in the system is fewer than *K*.

$$l_a = l(1 - P_K)$$

$$\therefore \quad l_a = \begin{cases} l\left[\dfrac{1 - (1 - r)r^K}{1 - r^{K+1}}\right] & r \neq 1 \\ l\left(1 - \dfrac{1}{K+1}\right) & r = 1 \end{cases}$$

$$\therefore \quad l_a = \begin{cases} \dfrac{l(1 - r^K)}{1 - r^{K+1}} & r \neq 1 \\ \dfrac{lK}{K+1} & r = 1 \end{cases} \qquad (148)$$

Little's Law does not depend on the distribution or queue discipline, as discussed in “Little's Law” on page 184, but it must consider the rate of input into the system. Therefore, the effective input rate l_a, rather than the Poisson rate l, should be used to calculate response time, queue time, and service time.

Element-lost Rate

The element-lost rate (l_b) is the complement of the effective input rate. Therefore, you calculate either by subtracting equations (148) from l or by multiplying l by the probability that the system contains *K* elements:

$$l_b = l - l(1 - P_K) = lP_K$$

$$\therefore \quad l_b = \begin{cases} \dfrac{l(1 - r)r^K}{1 - r^{K+1}} & r \neq 1 \\ \dfrac{l}{K+1} & r = 1 \end{cases} \qquad (149)$$

Equation (149) represents the rate at which elements are refused entry to the system, and therefore, are lost. If K = 1, the system loses any arriving elements when it is busy. This model is a proper description for telephone lines, and it is known as blocked-calls-cleared system with a single server.

Number of Elements in the System

The average number of elements in the system can be calculated using the definition of mean, as discussed in “Mean” on page 169:

$$\overline{N}_t = \sum_{n=0}^{K} kP_n$$

You can use the techniques described in “Using Derivatives” on page 180 and “Arithmetic Series” on page 173 to calculate the mean when $r \neq 1$ and $r = 1$, respectively. The results are:

$$\overline{N}_t = \begin{cases} \dfrac{r}{1-r}\left[\dfrac{(K+1)r^{K+1}}{1-r^{K+1}} - 1\right] & r \neq 1 \\ \dfrac{K}{2} & r = 1 \end{cases} \qquad (150)$$

Number of Elements in Service

Since this is a single-server system, the general equation (100) applies here. Combining equations (100) and (146) we can write:

$$\overline{N}_s = \begin{cases} \dfrac{r(1-r^K)}{1-r^{K+1}} & r \neq 1 \\ \dfrac{K}{K+1} & r = 1 \end{cases} \qquad (151)$$

Equation (151) also represents the average server utilization.

Number of Elements in Queue

Subtracting the number of elements in service from the elements in the system, we can write:

$$\overline{N}_q = \begin{cases} \dfrac{r^2}{1-r}\left[\dfrac{(K+r)r^{K+1}}{1-r^{K+1}} - 1\right] & r \neq 1 \\ \dfrac{K(K-1)}{2(K+1)} & r = 1 \end{cases} \qquad (152)$$

System Times

You can use Little's Law and equations (150), (151), and (152) to calculate the average response time, service time, and queue time, respectively:

$$\overline{T}_s = \frac{1}{m} \qquad \overline{T}_q = \frac{\overline{N}_q}{l_a} \qquad \overline{T}_r = \frac{\overline{N}_t}{l_a} \qquad (153)$$

M/M/1/K Example 1

Consider a cyclic model with a fixed number *K* of jobs circulating endlessly between two servers I and II, as shown in Figure 83.

The two servers have independent exponential service time distributions with parameters l and m, respectively, and the queue discipline is FIFO.

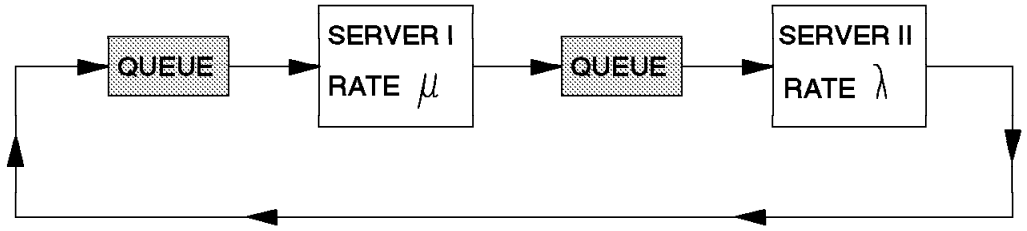

Figure 83. Two-Stage Cyclic Queue

As long as the number of jobs at server I is less than *K*, jobs will arrive there from server II, the inter-arrival times being distributed as the service time of server II; that is, exponential with parameter l.

When all jobs are at server I, no further arrival can take place; thus, this system can be modelled as M/M/1/K. If we denote $P[n, K-n]$ as the probability of *n* jobs either in the queue or in service with server I and *K-n* jobs with server II, this probability is given by $P[n]$ as in equation (147).

The utilization factor of servers I and II are given by:

$$r_1 = 1 - P_0$$

$$r_2 = 1 - P_k$$

$$\frac{r_1}{r_2} = r$$

The expected numbers N_1 and N_2 in the subsystems are:

$$E[N_1] = \sum_{n=0}^{K} nP_n$$

$$E[N_2] = K - E[N_1]$$

The average time taken for a job to go through both servers and make a complete cycle is:

$$\frac{K}{r_1 m} = \frac{K}{r_2 l}$$

This model can be used for batch systems where the processor can be taken as server I and the I/O subsystem as server II.

M/M/1/K Example 2

A barber runs a 1-man shop without appointments; customers arrive according to a Poisson distribution at a rate of six customers per hour and are willing to wait as long as they can occupy one of the four waiting seats available in the shop. Haircutting time is exponentially distributed with an average of nine minutes.

This is an M/M/1/5 system (four waiting seats plus one) with traffic intensity:

$$r = \frac{l}{m} = \frac{6/60}{1/9} = 0.9$$

- What is the percent of time a customer does not have to wait?

  A customer does not wait when there is no one in the shop:

$$\therefore \quad P_0 = \frac{1 - 0.9}{1 - (0.9)^6} = \frac{0.1}{0.468} = 0.214 \approx 21.4\%$$

- How long does one customer wait for his haircut?

  According to equations (152) and (153):

$$N_q = \frac{(0.9)^2}{1 - 0.9} - \frac{(5 + 0.9)(0.9)^6}{1 - (0.9)^6} = 1.41$$

$$\therefore \quad T_q = 1.41\frac{1 - (0.9)^6}{6(1 - (0.9)^5)} = 0.19$$

  The average waiting time is 0.19 hours or 11.4 minutes.

- How many customers per hour is the barber losing?

  Using equation (149):

$$l_b = \frac{6(1 - 0.9)(0.9)^5}{1 - (0.9)^6} = 0.75$$

- If his marginal profit is $5.00 per haircut and he works 45 hours per week, should he move to a next-door shop with six waiting seats if the rental would cost him $10.00 more a week?

  The number of customers lost in the new M/M/1/7 system is (assuming the same conditions):

$$l_b = \frac{6(1 - 0.9)(0.9)^7}{1 - (0.9)^8} = 0.50$$

  Using the answer from the previous item, he would get an additional $0.75 - 0.50 = 0.25$ customers per hour; therefore, his additional earnings would be:

$$0.25 \times 5 \times 45 = 56.25$$

  He should move because he would make an additional $56.25 per week.

M/M/1//M Model

Here we consider the case were we no longer have a Poisson input process with an infinite population, but rather have a finite population of possible elements.

When an element is outside the system, it is in an arriving state and the time it takes it to arrive is an exponentially distributed random variable with a mean of $1/\lambda$. Since all elements act independently of each other, when there are *n* elements in the system (queue plus service), then there are M - n elements in the arriving state. Therefore, the total average arrival rate in that state is $(M - n)\lambda$, as shown in Figure 84.

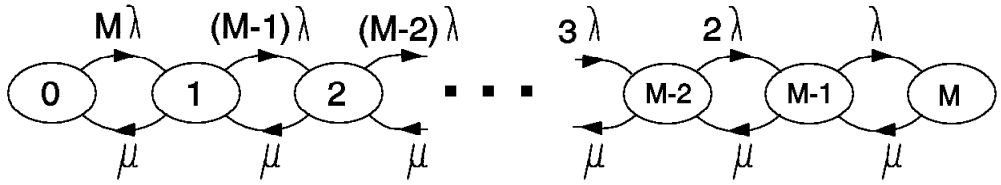

Figure 84. State Transition Rate Diagram for M/M/1//M

We can model these conditions appropriately with our birth-death process choosing the following coefficients:

$$\lambda_n = \begin{cases} (M - n)\lambda & 0 \le n \le M \\ 0 & n > M \end{cases}$$

Applying these coefficients to equation (81) and using the definition $r = \lambda/\mu$, as established by equation (61) on page 183, we can write:

$$P_1 = M\frac{\lambda}{\mu}P_0 = r\,M\,P_0$$

$$P_2 = (M - 1)\frac{\lambda}{\mu}r\,M\,P_0 = r^2(M - 1)M\,P_0$$

$$P_3 = (M - 2)\frac{\lambda}{\mu}r^2(M - 1)M\,P_0 = r^3(M - 2)(M - 1)M\,P_0$$

$$P_n = \frac{\lambda}{\mu}r^{n-1}(M - n + 1)\ldots(M - 1)M\,P_0 = r^n\frac{M!}{(M - n)!}P_0 \qquad 0 \le n \le M$$

As always, in order to determine P_0 we use the probability conservation equation from system (82).

$$\sum_{n=0}^{M} r^n\frac{M!}{(M - n)!}P_0 = 1 \quad \therefore \quad P_0 = \frac{1}{\sum_{n=0}^{M} r^n\frac{M!}{(M - n)!}} \tag{154}$$

Number of Elements in the System

According to equation (81), we can write:

$$P_{n+1} = r(M - n)P_n \quad \backslash \quad r\,n\,P_n = r\,M\,P_n - P_{n+1}$$

We can perform a summation on all terms:

$$r \sum_{n=0}^{M} nP_n = r\,M \sum_{n=0}^{M} P_n - \sum_{n=0}^{M} P_{n+1} \qquad (155)$$

The first summation in equation (155) is the number of elements in the system; the second summation is equal to one, by definition. The third summation can have its index range changed to n = 1 to M, observing that $P_{M+1} = 0$:

$$r\overline{N}_t = r\,M - \sum_{1}^{M} P_n = r\,M - (1 - P_0)$$

$$\backslash \quad \overline{N}_t = M - \frac{1 - P_0}{r} \qquad (156)$$

Number of Elements in Service

Using the same reasoning used when calculating the average for this random variable for the M/M/1 model on page 200, we find that:

$$\overline{N}_s = 1 - P_0 \qquad (157)$$

Number of Elements in Queue

We can calculate this average by difference:

$$\overline{N}_q = \overline{N}_t - \overline{N}_s = M - \frac{1 - P_0}{r} - (1 - P_0)$$

$$\backslash \quad \overline{N}_q = M - \left(1 + \frac{1}{r}\right)(1 - P_0) \qquad (158)$$

Number of Elements Out of the System

Since this is a finite population, it makes sense to determine the average number of elements that are not in the system (N_x):

$$\overline{N}_x = M - \overline{N}_t$$

$$\backslash \quad \overline{N}_x = \frac{1 - P_0}{r} \qquad (159)$$

Relative Number of Elements

If you divide the average number of elements in the system, in service, in the queue, and out of the system, by the size of the population you obtain the percentage of elements in each of these states. This is also the average percentage of time that each element spends in each of these states.

Percentage of Time in System

$$\frac{\overline{N}_t}{M} = 1 - \frac{1 - P_0}{Mr}$$

Percentage of Time in Service

$$\frac{\overline{N}_s}{M} = \frac{1 - P_0}{M}$$

Percentage of Time in Queue

$$\frac{\overline{N}_q}{M} = 1 - (1 + \frac{1}{r}) \frac{1 - P_0}{M}$$

Percentage of Time out of System

$$\frac{\overline{N}_x}{M} = \frac{1 - P_0}{Mr}$$

Effective Input Rate

Because we have a finite population, the input rate depends on how many elements are already in the system. To calculate the average or effective input rate, note that if the system has *n* elements in it, there are *M-n* possible arrivals elements outside the system, each with a mean arrival rate of λ; therefore, the mean input rate to the system is $(M - n)\lambda$. Summing over all possible *n* after weighting each term by its appropriate probability, we find that:

$$\lambda_a = \sum_{n=0}^{M} (M - n)\lambda P_n$$

$$= \lambda \left(M \sum_{n=0}^{M} P_n - \sum_{n=0}^{M} n P_n \right)$$

$$\therefore \quad \lambda_a = \lambda(M - \overline{N}_t) = \lambda \overline{N}_x$$

System Times

You can use Little's Law and the effective input rate established in equation (160) to write:

$$\overline{T}_s = \frac{1}{\mu} \qquad \overline{T}_q = \frac{\overline{N}_q}{\lambda \overline{N}_x} \qquad \overline{T}_r = \frac{\overline{N}_t}{\lambda \overline{N}_x}$$

General Application for Finite Population Models

Because the size of the population is finite, the M/M/1//M system is always stable. You can see that no restrictions were imposed when finding the equations for the number of elements. In fact, these type of queueing system are self-regulating because, when the system is busy and many elements in the queue, the rate at which additional elements arrive is correspondingly reduced, thus reducing further congestion of the system.

The M/M/1//M model and the corresponding multiple-server model M/M/c//M are often used to model machines under maintenance. The population consists of a certain number of operational machines and the queueing system consists of machines that need repair. Here we have a finite number of machines arriving at the queueing system when they break down.

The percentage of time a machine spends out of the system ($\overline{N}_x/M$), which means the machine is operational, should be close to 100%.

The tedious calculations corresponding to the equations of this model (and the M/M/c model) are very error prone. Tables containing values of P_0, N_x/M, and N_t/M for various values of r and *M* are published in some books, but a small REXX program can easily provide these values.

M/M/1//M Example 1

A group of machines need periodic adjustments in order to produce components within an allowed tolerance. The time between adjustments is exponentially distributed with an average interval between adjustments of six hours.

If the cost of an out-of-service machine (C_w) is $200.00 per hour, and the cost of a mechanic (C_s) is $60.00 per hour, let us calculate the optimum number of mechanics for a group of 50 machines, assuming that the service time of each mechanic is exponentially distributed with an average of 36 minutes.

This problem is properly addressed by the M/M/c//M model; however, you can use the M/M/1//M model if you assume that each mechanic works with a sub-group of machines without sharing the work with the others. In this case, you have to find the optimum size of the population (*M*) to be assigned to each mechanic. Once the assignment is made, the mechanic for a sub-group is the only one who can adjust the machines of that sub-group; he cannot give or receive help from other mechanics.

The traffic intensity in this example is:

$$r = \frac{36}{6 \cdot 60} = 0.1$$

The total cost per group of machines per hour is the number of broken machines times the cost of a break-down plus the cost of one mechanic. Assuming that each mechanic takes care of *M* machines and that the number of components produced is proportional to the time the machines are operational, the cost of production of each component per hour is:

$$\text{Cost} = \frac{\overline{N}_t C_w + C_s}{\overline{N}_x}$$

Figure 85 is a REXX program that calculates the results shown Table 18.

```
/* Cost of group of machines for a M/M/1//M system    */
/* Arguments: max - maximum number of groups analyzed  */
/*          rho - traffic intensity                    */
 arg max rho
 if max = '' then max = 8
 if rho = '' then rho = 0.1
 Cw = 200; Cs = 60

 do M = 2 to max
    sum = 1
    do n = 1 to M
       sum = sum + fat(M,M-n+1)*rho**n
    end n
    P0 = 1/sum
    Ns = 1 - P0
    Nx = (1-P0)/rho
    Nt = M - Nx
    Cost = (Nt*Cw + Cs)/Nx
    say 'M =' format(M,2) format(P0,3,5) format(Ns,3,5),
        format(Nx,3,5) format(Nt,3,5) format(Cost,4,5); say
 end M
exit

fat: procedure
 arg i,f
 r = 1
 do j = f until j = i
    r = r * j
 end j
return r
```

Figure 85. Cost of Group of Machines (Single Server)

Table 18 shows that five machines per mechanic yields the lowest cost and, therefore, we need 10 mechanics.

Table 18. Percentages and Cost (Single Server)

M	P_0	$\overline{N}_s$	$\overline{N}_x$	$\overline{N}_t$	$\frac{\overline{N}_t C_w + C_s}{\overline{N}_x}$
2	0.8197	0.1803	1.8033	0.1967	55.09
3	0.7321	0.2679	2.5794	0.3206	46.33
4	0.6467	0.3533	3.5333	0.4666	43.39
5	**0.5640**	**0.4361**	**4.3605**	**0.6395**	**43.09**
6	0.4845	0.5155	5.1549	0.8452	44.43
7	0.4090	0.5910	5.9096	1.0904	47.06
8	0.3383	0.6617	6.6168	1.3832	50.87

M/M/1//M Example 2

A group of 10 machines break down according to a Poisson process with one mechanic assigned to repair them. The average length of time for which a machine remains in working order is 30 hours and the time required to repair one machine has an exponential distribution with mean three hours.

Figure 86 is a sample REXX program that answers the following questions related to above problem (routine FAT is the same as in example 1):

- What is the probability that five or more machines are out of order at the same time?

$$P_{n \geq 5} = \sum_{n=5}^{10} P_n = 0.0914 \approx 9.14\%$$

- What is the average number of machines in working order?

$$\overline{N}_x = 8.06$$

- What is the average period of time a machine is out of order?

$$\overline{T}_r = 7.73 \text{ hours}$$

- What is the average fraction of time the mechanic is busy?

  The mechanic is busy when there is at least one machine under repair:

$$1 - P_0 = 0.755 \approx 75.5\%$$

```
/* M/M/1//M example - n machines, one mechanic   */
/* Parameters: number of machines               */

 arg M
 if M  = '' then M = 10
 lambda = 1/32; mu = 1/3
 rho = lambda/mu
 k = 5  /* want probability or 5 or more machines */

 P0m1 = 1
 do n = 1 to M
    P0m1 = P0m1 + fat(M,M-n+1)*rho**n
 end n
 Pge5 = 0
 do n = k to M
    Pge5 = Pge5 + fat(M,M-n+1)*rho**n
 end n

 P0   = 1/P0m1
 Pge5 = Pge5*P0
 Busy = 1 - P0
 Nx   = (1-P0)/rho
 Nt   = M - Nx
 Tr   = Nt/(lambda*Nx)
 say  M '  ' Pge5 '   '  Nx '    ' Tr '     ' Busy
exit
```

Figure 86. One Mechanic

M/M/c//M Model

This model is similar to the M/M/1//M model, but with many servers. Assuming that c < M almost everything discussed in "M/M/1//M Model" on page 229 also applies here.

The state transition diagram is a combination of both M/M/c and M/M/1//M models, as shown in Figure 87

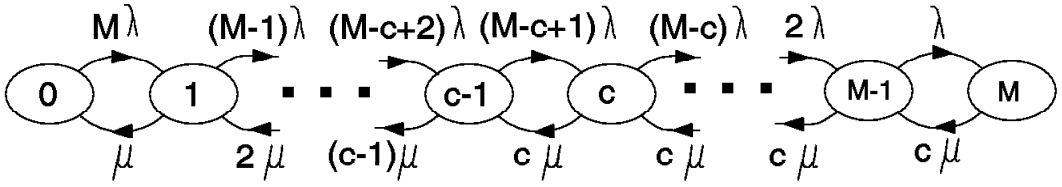

Figure 87. State Transition Rate Diagram for M/M/c//M

We can model these conditions appropriately with our birth-death process, choosing the following coefficients:

$$\lambda_n = \begin{cases} (M - n)\lambda & 0 \le n < M \\ 0 & n \ge M \end{cases}$$

$$\mu_n = \begin{cases} n\mu & 1 \le n < c \\ c\mu & n \ge c \end{cases}$$

Applying the coefficients to equation (81) and using the definition of traffic intensity ($u = \lambda/\mu$), we can prove that:

$$P_n = \begin{cases} \binom{M}{n} u^n P_0 & 0 \le n < c \\ \binom{M}{n} \frac{n!}{c^{n-c} c!} u^n P_0 & c \le n \le M \end{cases}$$

$$P_0 = \left[\sum_{n=0}^{c-1} \binom{M}{n} u^n + \sum_{n=c}^{M} \binom{M}{n} \frac{n!}{c^{n-c} c!} u^n \right]^{-1}$$

Where the binomial coefficient is defined in the usual way:

$$\binom{M}{n} = \frac{M!}{n!(M-n)!}$$

Number of Elements in the System

Using the definition of the mean of a random variable:

$$\overline{N}_t = P_0 \left[\sum_{n=0}^{c-1} n \binom{M}{n} u^n + \sum_{n=c}^{M} n \binom{M}{n} \frac{n!}{c^{n-c} c!} u^n \right]$$

Number of Elements in Service

The average number of elements in service is:

$$\overline{N}_s = \sum_{0}^{c-1} n P_n + \sum_{n=c}^{M} c P_n$$

$$= \sum_{0}^{c-1} n P_n + c \left(1 - \sum_{n=0}^{c-1} P_n \right)$$

$$= c - \sum_{n=0}^{c-1} (c-n) P_n$$

$$\overline{N}_s = c - P_0 \sum_{n=0}^{c-1} (c-n) \binom{M}{n} u^n$$

Number of Elements in Queue

The average number of elements in the queue is:

$$\overline{N}_q = \overline{N}_t - c + P_0 \sum_{n=0}^{c-1} (c-n) \binom{M}{n} u^n$$

Number of Elements Out of the System

As in the M/M/1//M model, the number of elements out of the system is:

$$\overline{N}_x = M - \overline{N}_t$$

Effective Input Rate

The effective rate is the same as established for the M/M/1//M model:

$$\lambda_a = \lambda (M - \overline{N}_t)$$

M/M/c//M Example

As an example, let us solve the same problem presented under the M/M/1//M queue on page 232, considering now that any mechanic can repair any broken machine. Here we consider all machines as a single group; that is, we model the system as an M/M/c/50 queue, and find the number of servers *c* that minimizes the cost.

The total cost per hour is the number of broken machines times the cost of a break-down plus the cost of *c* mechanics.

$$\text{Cost} = \frac{\overline{N}_t\, C_w + c\, C_s}{\overline{N}_x}$$

Table 19 shows the results for several number of mechanics; you can see that seven mechanics yields the lowest cost.

Table 19. Percentages and Cost (Multiple Servers)

c	P_0	$\overline{N}_s$	$\overline{N}_x$	$\overline{N}_t$	$\frac{\overline{N}_t C_w + c C_s}{\overline{N}_x}$
3	0.0000	2.9989	29.988	20.012	139.47
4	0.0018	3.8904	38.850	11.150	83.58
5	0.0054	4.3489	43.272	6.7284	38.03
6	0.0074	4.5097	44.726	5.2742	31.63
7	**0.0082**	**4.5694**	**45.205**	**4.7954**	**30.51**
8	0.0084	4.5959	45.370	4.6299	30.98
9	0.0085	4.6106	45.427	4.5728	32.02
10	0.0085	4.6212	45.446	4.5538	33.24
11	0.0085	4.6304	45.452	4.5478	34.53

The optimum solution is an M/M/7/50 system, which requires seven mechanics with a total cost of $30.51 per hour.

If you compare this result with the combined M/M/1//5 case, presented in Table 18 on page 233, you can see that:

- In this M/M/c//50 example, *c=1* means that there would be one mechanic repairing all 50 machines; this is completely different from the M/M/1//M example whose solution is a combination of 10 independent M/M/1//5 queues.
- The average number of machines in working condition is 45.205 in the multiple-server model as opposed to 43.605 (4.3605 ´ 10) in the single-server model.
- The lowest cost in each M/M/1//5 system is $43.09 per group of machines per hour; therefore the total cost for the all 10 groups is $430.90 per hour.

Figure 88 is a REXX program that calculates the results shown in Table 19.

```
/* Cost of group of machines for a M/M/c//M system    */
/* Arguments: max - maximum number of servers analyzed */
/*          u   - traffic intensity                   */
/*          M   - number of machines                  */
 arg max u M
 if max = '' then max = 11
 if u   = '' then u   = 0.1
 if M   = '' then M   = 50
 Cw = 200; Cs = 60

 do c = 2 to max
    sump = 1; suml = 0; sums = 1
    do n = 1 to c-1
       sum = bin(M,n)*u**n
       sump = sump + sum
       suml = suml + n*sum
       sums = sums + (c-n)*sum
    end n
    fatc = fat(c)
    do n = c to M
       sum = bin(M,n)*fat(n)/(fatc*c**(n-c))*u**n
       sump = sump + sum
       suml = suml + n*sum
    end n

    P0 = 1/sump
    Ns = c - P0*sums
    Nt = P0*suml
    Nx = M - Nt
    Cost = (Nt*Cw + c*Cs)/Nx
    say 'c =' format(c,2) format(P0,3,5) format(Ns,3,5),
        format(Nx,3,5) format(Nt,3,5) format(Cost,4,5); say
 end c
exit

/* evaluate factorial of argument */
fat: procedure
 arg i
 r = 1
 do j = 1 to i
    r = r * j
 end j
return r

/* evaluate binomial coefficient  */
bin: procedure
 arg i,f
 r = 1
 do j = i to i-f+1 by -1
    r = r * j
 end j
return r/(fat(f))
```

Figure 88. Cost of Group of Machines (Multiple Servers)

M/G/1 Model

The M/G/1 queue is a single-server system with Poisson arrivals and arbitrary service-time distribution. This system is modelled by non-Markovian processes and since it requires higher level of mathematical knowledge, we just present the main results here.

The M/G/1 equations differ from the M/M/1 equations by the factor $(1 + c_s^2)/2$, where c_s^2 is the coefficient of variation of the service time distribution (see "Coefficient of Variation" on page 172 for a definition). If you make $c_s^2 = 1$, which is the coefficient of variation for the exponential distribution, all M/G/1 equations reduce to the M/M/1 equations.

Number of Elements in Service

The average number of elements in service is equal the server utilization:

$$\overline{N}_s = r$$

Number of Elements in Queue

The average number of elements in the queue is:

$$\overline{N}_q = \frac{r^2 \quad 1 + c_s^2}{1 - r \quad 2}$$

Number of Elements in the System

This is known as the Pollaczek-Khintchine mean-value formula:

$$\overline{N}_t = r + \frac{r^2 \quad 1 + c_s^2}{1 - r \quad 2}$$

System Times

Using Little's Law, we can calculate:

Service Time

$$\overline{T}_s = \frac{1}{m}$$

Queue Time

$$\overline{T}_q = \frac{r T_s \quad 1 + c_s^2}{1 - r \quad 2}$$

Response Time

$$\overline{T}_r = \overline{T}_s + \frac{r T_s \quad 1 + c_s^2}{1 - r \quad 2}$$

Table 20 shows the response time factors for different coefficients of variation and server utilizations. To get the response time, multiply each value in the table by the service time. Figure 89 is a graphical representation of the same values.

You can see that, at low server utilization, the response time is about the same for all coefficient of variations.

Table 20. M/G/1 Response Time

Server Utilization	Coefficient of Variation and Response Time				
	0	0.5	1.0	1.5	2.0
0.10	1.06	1.08	1.11	1.14	1.16
0.20	1.13	1.19	1.25	1.31	1.38
0.30	1.21	1.32	1.43	1.54	1.64
0.40	1.33	1.50	1.66	1.83	2.00
0.50	1.50	1.75	2.00	2.25	2.50
0.60	1.75	2.13	2.50	2.88	3.25
0.70	2.17	2.75	3.33	3.92	4.50
0.80	3.00	4.00	5.00	6.00	7.00
0.90	5.50	7.75	10.0	12.3	14.5

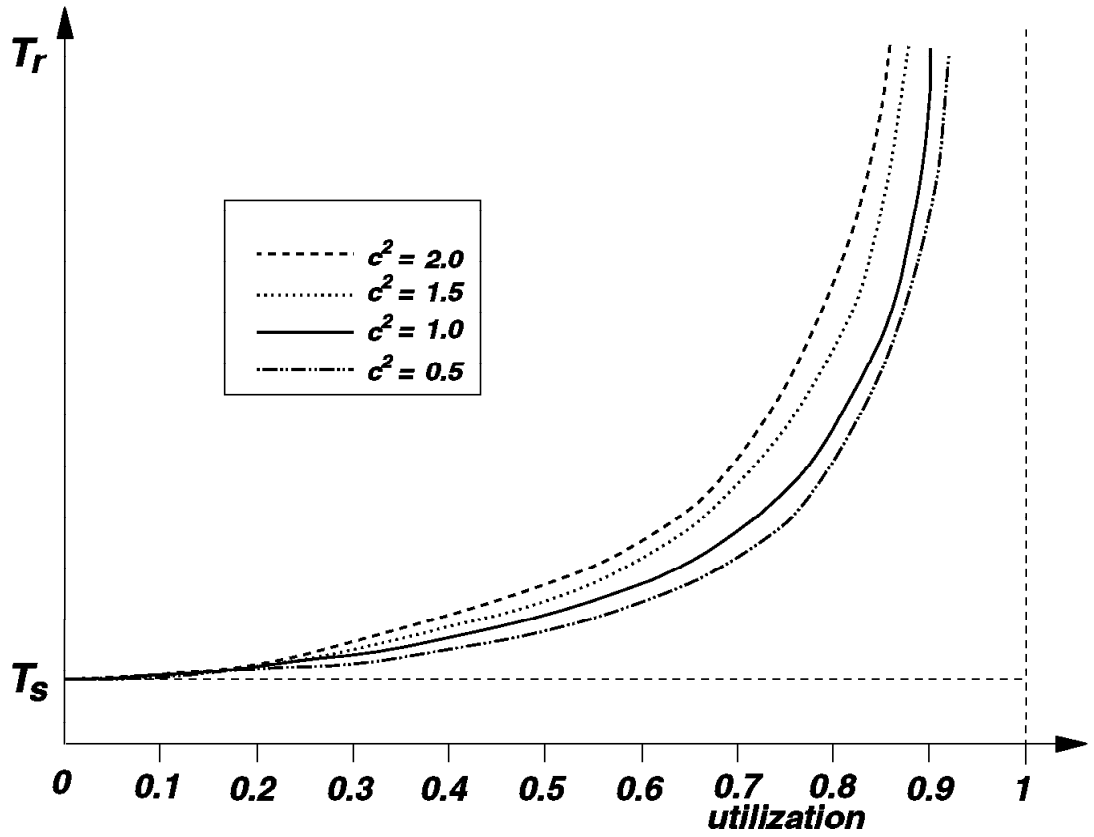

Figure 89. M/G/1 Response Time

M/G/1 Example

A company operates a large number of machines that need lubrication according to a Poisson process with a mean of eight machines per hour. There are three types of machines, each type requiring a different lubrication time; half of the machines require two minutes, one-third requires three minutes, and one-sixth requires six minutes. Lubrication is a straight forward procedure that always require the same time for each type of machine.

Evaluate the average number of machines in queue and the average queue time for this M/G/1 system, and compare them with an M/M/1 system with the same arrival rate and average lubrication time.

- M/G/1

  The mean and variance of the service time are calculated as described in "Parameters of Random Variables" on page 169:

$$\overline{T}_s = 2\frac{1}{2} + 3\frac{1}{3} + 6\frac{1}{6} = 3$$

$$s^2 = (2-3)^2\frac{1}{2} + (3-3)^2\frac{1}{3} + (6-3)^2\frac{1}{6} = 2$$

$$c_s^2 = \frac{2}{3^2} = 0.222$$

$$r = \frac{8 \cdot 3}{60} = 0.4$$

$$\overline{N}_q = \frac{0.16}{0.6} \cdot \frac{1.222}{2} = 0.163$$

$$\overline{T}_q = \frac{0.4 \cdot 3}{0.6} \cdot \frac{1.222}{2} = 1.22$$

- M/M/1

  If you use the same arrival rate and average service time for the M/M/1 system, the utilization is also the same:

$$\overline{N}_q = \frac{0.16}{0.6} = 0.267$$

$$\overline{T}_q = \frac{0.4 \cdot 3}{0.6} = 2.0$$

Therefore, the M/G/1 system is better than the corresponding M/M/1 system in this case. Since the M/G/1 equations for the number of elements in the queue and queue time are the corresponding M/M/1 equations multiplied by the factor $(1 + c_s^2)/2$, the M/G/1 system is better whenever the coefficient of variation is less than one; this is easily verified from Figure 89.

The best result for the M/G/1 model corresponds to a zero coefficient of variation. These systems have constant (deterministic) service times and are know as M/D/1 systems. Since the coefficient of variation measures the dispersion of the values, small coefficients of variation correspond to distributions where the values are clustered around the mean, providing more consistent response times.

Chapter 10. Simulation

It often turns out that it is not possible to develop analytical models for queueing systems. This can be due to the characteristics of the input or service mechanisms, the complexity of the system design, the nature of the queue discipline, or combinations of these factors. In these cases, it may be necessary to resort to simulation.

While simulation may offer a solution for many analytically intractable models, it is not in itself a panacea. There are a considerable number of pitfalls you may encounter in using simulation. Since simulation is comparable to analysis by experimentation, you have all the usual problems associated with running experiments in order to make inferences concerning the modelled systems, and you must be concerned with such things as run length, number of replications, and statistical significance.

Another drawback to simulation occurs when you are interested in optimum design of a queueing system such as the determination of the optimum number of channels for a system where conflicting system costs are known. If it is necessary to study the system using simulation, you must rely on the techniques for searching experimental output. These techniques are often not as neat as the mathematics of optimization for analytical functions. Frequently you will merely try a few alternatives and simply choose the best among them, as we have done with the examples in “M/M/1//M Model” on page 229 and in “M/M/c//M Model” on page 235. It might well be that none of the alternatives tried is optimum nor even near optimum. How close you get to the optimum in a simulation study often depends on how clever you are in considering the alternatives to be investigated. Since there are no rules for selecting the best alternatives, simulation has often been referred to as an art.

Nevertheless, simulation can be an extremely important tool and is often the only procedure that can be used in analyzing many of the complex queueing systems encountered in practice.

Simulation Elements

A simulation model is normally considered as consisting of data generation and bookkeeping.

Data generation involves the production of representative inter-arrival times and service times where needed throughout the queueing system. Generally, this involves producing representative observations from the relevant probability distributions, and it is this aspect to which the term *Monte Carlo* has been applied. A Monte Carlo simulation is one in which it is necessary to generate at least one stream of random observations from some specified probability distribution.

The bookkeeping phase of a simulation deals with updating the system when new events (arrivals and departures) occur, monitoring the system states as they change, and keeping track of the quantities of interest, such as idle time and busy time, so that system parameters, such as queue time and response time, can be calculated.

Data Generation

Let us consider the problem of generating a sequence of numbers that could be considered as being typical observations from a given probability distribution. Conceptually, you could take any distribution, approximate it by a finite number of discrete points, place into a large bowl chips designated with the value of the random variable in amounts proportional to the probabilities, draw chips randomly, and record the values.

For example, if you want to simulate the 2-dice throwing experiment described in "Discrete Random Variables" on page 163, you would put into the large bowl one chip marked with number 2, two chips marked with number 3, three chips marked with number 4, and so on until the last chip (one chip marked with number 12). Now, a random draw of a chip from the bowl simulates the 2-dice throwing experiment because the probability of getting a five, for example, is the same as in the real experiment.

As you can see, this may be a time-consuming process and at best approximate, the degree of approximation depending on the discrete representation of the probability distribution and how close to purely random the selection process can be made.

To avoid some of these problems, we concentrate on a procedure that generates mathematically a stream of random observations from any given distribution. In fact, these observations are pseudo-random because they are in reality a deterministic sequence generated by a completely specified mathematical procedure and can be reproduced as desired. However, the observations act as random and pass any statistical test on randomness and representativeness.

The method described below is sometimes referred to as the inverse or probability distribution transformation method. It can best be described graphically by considering a plot of the probability distribution function (PDF) from which we want to generate random variables.

Assume you want to generate values for the 2-dice experiment described in "Discrete Random Variables" on page 163. The procedure consists in first generating a uniform random variable, say r_i, in the interval *(0,1)*, corresponding to the values that the PDF may assume (a PDF is a nondecreasing function with a range between zero and one). To obtain x_i; that is, a typical observation of the experiment, you simply enter the ordinate with r_i and project over and down, as shown in Figure 90; x_i is the resulting value from the abscissa.

You can see from the figure that the probability of getting a seven using this procedure is the probability of selecting a uniformly distributed random number between 15/36 and 21/36, which is 6/36; this is exactly the probability of getting a seven in the 2-dice experiment represented by the PDF.

The problem can be easily solved if the distribution function can be inverted analytically. Unfortunately analytical inversion is not possible for most distributions, and other techniques must be used.

Discrete distributions can be solved by adding the probabilities until they reach the random number within a desired approximation. Continuous distributions can be solved using numerical inversion techniques or a discrete approximation.

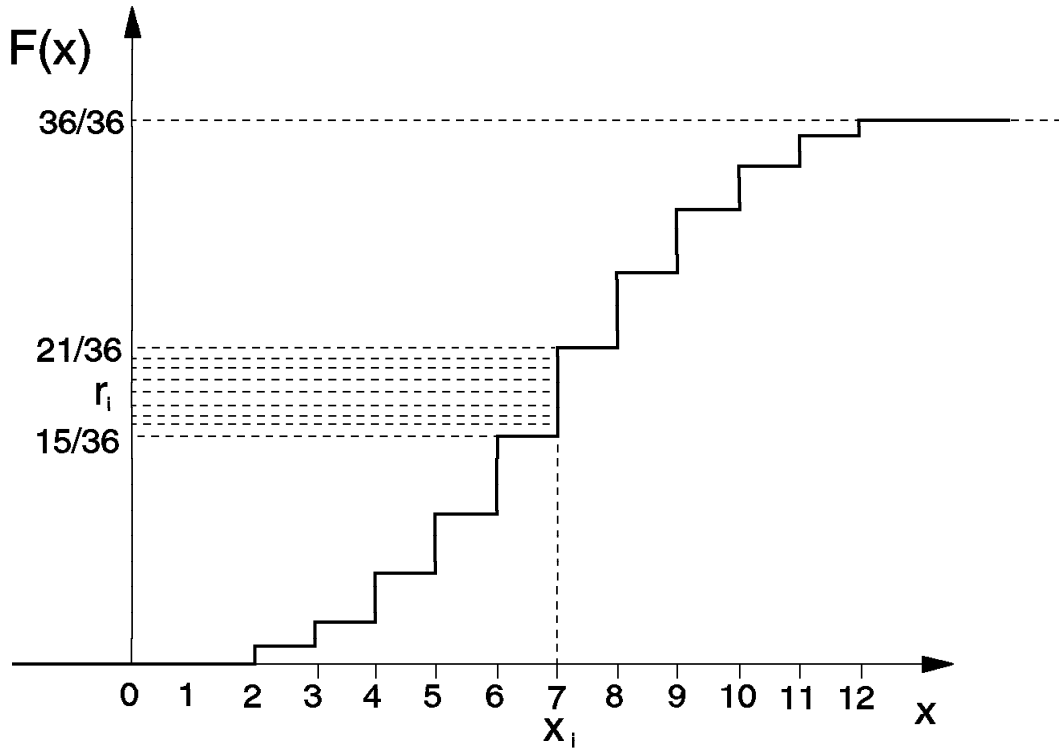

Figure 90. Inversion Technique for Generating Random Variables

Let us consider the case of the exponential distribution with parameter λ (see "Poisson and Exponential Distributions" on page 195); its PDF is given by:

$$F(t) = 1 - e^{-\lambda t}$$

Entering the ordinate with a uniform random number *r* in the interval *(0,1)*, and finding the resulting *t* after the projection amounts to solving the following equation for *t*:

$$r = 1 - e^{-\lambda t}$$

$$e^{-\lambda t} = 1 - r$$

Since *r* is uniform in *(0,1)*, it is immaterial whether we use *r* or *1-r* as our random number:

$$e^{-\lambda t} = r$$

Applying natural logarithms on both sides finally gives:

$$t = \frac{-\ln(r)}{\lambda} \qquad (161)$$

If you supply the arrival rate for a Poisson process, equation (161) provides a value for the inter-arrival time for each value of the uniform random variable *r*.

Bookkeeping

Bookkeeping is the process of updating the system when events occur, recording items of interest, and calculating system parameters. There are two general methods of accomplishing this bookkeeping:

- Time-oriented
- Event-oriented

Time-oriented bookkeeping involves setting a basic time unit and including in the program a timer that advances each basic unit, updating the system states after each time unit.

Event-oriented bookkeeping updates the system state only when events (arrivals or departures) occur. Since there is not necessarily an event every basic time unit, the timer is increased by a variable amount corresponding to the time difference between consecutive events.

Generally, event-oriented bookkeeping is more efficient than time-oriented bookkeeping; therefore, the examples presented in this section use event-oriented bookkeeping.

Manual Simulation

In order to understand the process and to fix some concepts let us simulate manually the behavior of a queueing system with the following inter-arrival and service times:

Inter-arrival time - 9 6 4 7 9 5 8 4 10 6 12 6 8 9 5 7 8

Service time 3 7 9 9 10 5 7 5 5 3 6 3 5 4 9 9 8 6

The results of the simulation is shown in Table 21; you can use the REXX program shown in Figure 91 to verify your calculations.

The simulation starts with clock zero (Table 21, column 1) when the first element arrives, indicated by an *A* in column 2. The clock for subsequent arrivals is the clock of the previous arrival added to the next entry in the list of inter-arrival times. A departure is indicated by a *D* in column 2, and it happens when the element being served leaves the system (column 5 of a previous element). The number between parentheses in column 2 indicates the order of the element when it enters the system.

An element enters service (column 3) either immediately, if there is no queue, or when the element being served leaves the system. The service time (column 4) is given by the next entry in the list of service times.

Columns 8 and 9 represent the number of elements after the time indicated by the clock (column 1); that is, after an arrival or a departure.

Table 21 can be used to calculate several parameters related to the model. For example, to calculate the average queue time, you add all queue times in column 6 and divide by the number of elements in the simulation (18).

To calculate the average number of elements in the system, it is necessary to multiply the value in column 9 by the elapsed time between the present and next event, sum the results of all rows, and divide by the total elapsed time.

Table 21. Event-Oriented Bookkeeping

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Clock	**Event**	**Time into Service**	**Service Time**	**Time out of System**	**Queue Time**	**Response Time**	**Number in Queue after Clock**	**Number in System after Clock**
0	A(01)	0	3	3	0	3	0	1
3	D(01)	-	-	-	-	-	0	0
9	A(02)	9	7	16	0	7	0	1
15	A(03)	16	9	25	1	10	1	2
16	D(02)	-	-	-	-	-	0	1
19	A(04)	25	9	34	6	15	1	2
25	D(03)	-	-	-	-	-	0	1
26	A(05)	34	10	44	8	18	1	2
34	D(04)	-	-	-	-	-	0	1
35	A(06)	44	5	49	9	14	1	2
40	A(07)	49	7	56	9	16	2	3
44	D(05)	-	-	-	-	-	1	2
48	A(08)	56	5	61	8	13	2	3
49	D(06)	-	-	-	-	-	1	2
52	A(09)	61	5	66	9	14	2	3
56	D(07)	-	-	-	-	-	1	2
61	D(08)	-	-	-	-	-	0	1
62	A(10)	66	3	69	4	7	1	2
66	D(09)	-	-	-	-	-	0	1
68	A(11)	69	6	75	1	7	1	2
69	D(10)	-	-	-	-	-	0	1
75	D(11)	-	-	-	-	-	0	0
80	A(12)	80	3	83	0	3	0	1
83	D(12)	-	-	-	-	-	0	0
86	A(13)	86	5	91	0	5	0	1
91	D(13)	-	-	-	-	-	0	0
94	A(14)	94	4	98	0	4	0	1
98	D(14)	-	-	-	-	-	0	0
103	A(15)	103	9	112	0	9	0	1
108	A(16)	112	9	121	4	13	1	2
112	D(15)	-	-	-	-	-	0	1
115	A(17)	121	8	129	6	14	1	2
121	D(16)	-	-	-	-	-	0	1
123	A(18)	129	6	135	6	12	1	2
129	D(17)	-	-	-	-	-	0	1
135	D(18)	-	-	-	-	-	0	0

Similarly, you can calculate the average number of elements in service and in the queue. The average size of nonempty queues can be calculated in the same way, ignoring the entries that contain a zero in column 8.

An easier way to calculate the average number of elements is by applying Little's Law. If you want the number of elements in the queue, for example, add all queue times (column 6) and divide by the total elapsed time.

Figure 91 is a sample REXX program that will verify your runs; just code two arrays of inter-arrival and service times (variables *arrivals* and *services*). Note that the clock starts at zero and the first inter-arrival time corresponds to the second service time; therefore, the last inter-arrival time is a dummy entry and should have a high value to force the elements still in service to depart.

```
/* Simulation of a Single Server System - FIFO discipline             */
/* Produces an event-oriented bookkeeping table for single-servers    */
/* Inter-arrival and service times are tabulated                      */
/* Last inter-arrival is dummy; code a high value to force departures  */

 arrivals = '9 6 4 7  9 5 8 4 10 6 12 6 8 9 5 7 8 99' /* inter-arrivals */
 services = '3 7 9 9 10 5 7 5  5 3 6  3 5 4 9 9 8  6' /* service times  */
 parse value 0 with tos 1 tna 1 snt 1 snq wos /* set all=0 and wos=''   */

 do k = 1 to words(services)          /* simulation size     */
   clk = tna                         /* clock = arrival     */
   if tos > tna then do               /* if server is busy:  */
     tis = tos                       /* time into service   */
     snq = snq + 1                   /* number in queue     */
   end
   else do                           /* if server is idle:  */
     tis = tna                       /* time into service   */
     snq = 0                         /* number in queue     */
   end
   snt = snt + 1                     /* number in system    */
   sts = word(services,k)             /* get service time    */
   tos = tis + sts                   /* time out of service */
   wos = wos tos right(k,2,0)         /* departure times     */
   stq = tis - clk                   /* queue time          */
   str = tos - clk                   /* response time       */
   sta = word(arrivals,k)             /* get inter-arrival   */
   tna = tna + sta                   /* time of next arrival */

   say format(clk,5) '  A('right(k,2,0)')' format(tis,5),
      format(sts,5) format(tos,5) format(stq,5),
      format(str,5) format(snq,5) format(snt,5)

   do words(wos)/2                    /* check all departures */
     if word(wos,1) > tna then leave   /* arrival is earlier  */
     parse var wos clk eln wos         /* element is gone     */
     snt = snt - 1                    /* number in system    */
     if snq > 0 then snq = snq - 1     /* number in queue     */

     say format(clk,5) '  D('eln')    -    -    -    -    -',
        format(snq,5) format(snt,5)
   end
 end
exit
```

Figure 91. Sample REXX for Event-Oriented Bookkeeping

Single-Server System Simulation

The REXX program shown in Figure 92 is an example of a complete simulation of an M/M/1 system. In particular, this example is suited for DASD because the arrival rate is expressed as requests per second (IORATE) and the service time is expressed in milliseconds. The program is generic and can be used to simulate any single-server system (G/G/1) provided that you replace one or both routines that generate the exponentially distributed inter-arrival times (GENARR) and service times (GENSER) with the appropriate distribution to be simulated.

The following techniques are used to calculate the system parameters (the symbols in the bullets are used in the results presented in Table 22):

- Average arrival rate (λ)

  The average arrival rate (asar) is the number of simulated elements (runl) divided by the total elapsed time (tna); it is multiplied by 1000 to correct it to the proper unit (elements per second).

- Service rate (μ)

  The service rate (asmu) is the total number of elements (runl) divided by the cumulative service time (cts).

- Utilization (ρ)

  The simulated utilization (srho) is the cumulative service time (cts) divided by the total elapsed time (tna). The cumulative service time (cts) is also equal to the total elapsed time (tna) minus the cumulative idle time (cit).

- Average service time ($\overline{T}_s$)

  The average service time (asts) is the cumulative service time (cts) divided by the number of simulated elements (runl).

- Average queue time ($\overline{T}_q$)

  The average queue time (astq) is the cumulative queue time (ctq) divided by the number of simulated elements (runl).

- Average response time ($\overline{T}_r$)

  The average response time (astr) is the sum of the average service time (asts) and the average queue time (astq).

- Average queue time when there is a queue ($\overline{T}_w$)

  The average queue time when there is a queue (astw) is the cumulative queue time (ctq) divided by the number of times the queue is not empty (cnz).

- Average idle time ($E[I]$)

  The average idle time (asit) is the cumulative idle time (cit) divided by the number of idle periods (cip). The end of an idle period is detected when the time into service (tis) of an arriving element is equal to its arrival time (tna); the corresponding idle period is the arrival time minus the previous clock.

- Average busy time ($E[B]$)

  The average busy time (asbt) is the cumulative service time (cts) divided by the number of busy periods (cbp). The end of a busy period is detected when the number of elements in the system after a departure (snt) is zero. Also, the number of busy periods (cbp) is equal to the number of idle periods (cip) plus one.

```
/* Simulation of an M/M/1 Queue - DASD I/O subsystem.                */
/* Arguments: mmar = model disk iorate in IOs/second   (def.:20)    */
/*          mmts = model service time in milliseconds (def.:12)     */
/*          seed = seed for random numbers          (def.:52837)  */
/*          runl = number of simulated elements      (def.:100000) */

 arg mmar mmts '(' runl seed             /* get arguments          */
 if mmar = '' then mmar = 20             /* arrival rate: IOs/sec   */
 if mmts = '' then mmts = 12             /* average service time ms */
 if seed = '' then seed = 52837          /* random number gen seed  */
 if runl = '' then runl = 100000         /* simulation run length   */
 lambda = mmar/1000                     /* IOs/millisecond        */
 rho = lambda*mmts                      /* model utilization      */
 mu = 1/mmts                           /* service rate           */
 parse value 0 with,                    /* set all variables to    */
    tna 1 tos 1 cer 1 cnz 1 cts 1 ctq 1,  /* zero, except wos which  */
    cip 1 cqt 1 cbp 1 cit 1 snq 1 snt wos /* is set to empty ('')    */
 elp = time('E')                        /* zero program clock     */
 r = random(1,99999,seed)                /* initialize stream      */

/*               Start of Simulation                              */

 do k = 1 to runl                       /* simulation size        */
   if tos > tna then do                  /* server is busy         */
      tis = tos                         /* time into service      */
      snq = snq + 1                     /* number in queue        */
   end
   else do                             /* server is idle         */
      tis = tna                         /* time into service      */
      snq = 0                           /* number in queue        */
      if k > 1 then do                  /* idle time after clock 0 */
         cit = cit + tna - clk          /* cumulative idle time    */
         cip = cip + 1                  /* number of idle periods  */
      end
   end
   clk = tna                           /* clock = arrival        */
   sts = genser(mu)                     /* generate service time   */
   tos = tis + sts                      /* time out of system     */
   wos = wos tos                        /* departure times        */
   stq = tis - clk                      /* simulated queue time    */
   str = tos - clk                      /* simulated response time */
   sta = genarr(lambda)                  /* inter-arrival time     */
   tna = tna + sta                      /* time of next arrival    */
   if snq > 0 then cnz = cnz + 1         /* # times queue nonzero   */
   if snq = 1 then qck = clk             /* queueing period ini.clk */
   if snt = 0 then cbp = cbp + 1         /* number of busy periods  */
   cer = cer + snt                      /* # elements at arrivals  */
   snt = snt + 1                       /* number in system       */
   cts = cts + sts                      /* cumulative service time */
   ctq = ctq + stq                      /* cumulative queue time   */

   do words(wos)                        /* analyze departures     */
      if word(wos,1) > tna then leave    /* an arrival is earlier   */
      parse var wos clk wos              /* first departure        */
      snt = snt - 1                     /* number in system       */
      if snq = 1 then cqt = cqt + clk-qck /* cumulative queue period */
      if snq > 0 then snq = snq - 1      /* number in queue        */
   end
 end k
```

Figure 92 (Part 1 of 2). Single-server Simulation Program

```
/*              Simulated System Parameters                     */

 tna  = tna - sta                       /* drop last clk increment */
 asar = 1000*runl/tna                    /* average arrival rate    */
 asmu = runl/cts                        /* average service rate    */
 srho = cts/tna                         /* utilization (rho)       */
 asts = cts/runl                        /* average service time    */
 astq = ctq/runl                        /* average queue time      */
 astr = asts + astq                     /* average response time   */
 astw = ctq/cnz                         /* aver. n-zero queue time */
 asit = cit/cip                         /* average idle time       */
 asbt = cts/cbp                         /* average busy time       */
 asns = cts/tna                         /* average # in service    */
 asnq = ctq/tna                         /* average # in queue      */
 asnt = asns + asnq                     /* average # in system     */
 past = cer/runl                        /* = asnt, but at arrivals */
 asnw = ctq/cqt                         /* average # n-zero queue  */
 asnb = runl/cbp                        /* av.# served busy period */

 say 'simulation :' format(asar,4,3) format(asmu,4,3) format(srho,4,3)
 say '           ' format(asts,4,3) format(astq,4,3) format(astr,4,3),
                   format(astw,4,3) format(asit,4,3) format(asbt,4,3)
 say '           ' format(asns,4,3) format(asnq,4,3) format(asnt,4,3),
                   format(past,4,3) format(asnw,4,3) format(asnb,4,3)

/*                   M/M/1 Parameters                           */

 mmts = 1/mu                           /* M/M/1 service time      */
 mmtq = rho/(1-rho)*mmts                /* M/M/1 queue time        */
 mmtr = mmtq + mmts                     /* M/M/1 response time     */
 mmit = 1/lambda                        /* M/M/1 idle time         */
 mmbt = mmts/(1-rho)                    /* M/M/1 busy time         */
 mmtw = mmtq/rho                        /* M/M/1 n-zero queue time */
 mmns = rho                            /* M/M/1 # in service      */
 mmnq = rho**2/(1-rho)                  /* M/M/1 # in queue        */
 mmnt = mmns + mmnq                     /* M/M/1 # in system       */
 mmnb = 1/(1-rho)                       /* M/M/1 # in busy period  */
 mmnw = 1/(1-rho)                       /* M/M/1 # in n-zero queue */
 mrun = tna/(1000*60)                    /* simulated elapsed time  */

 say 'M/M/1 model:' format(mmar,4,3) format(mu,4,3)   format(rho,4,3)
 say '           ' format(mmts,4,3) format(mmtq,4,3) format(mmtr,4,3),
                   format(mmtw,4,3) format(mmit,4,3) format(mmbt,4,3)
 say '           ' format(mmns,4,3) format(mmnq,4,3) format(mmnt,4,3),
                   format(mmnt,4,3) format(mmnw,4,3) format(mmnb,4,3)
 say; say 'Simulated elapsed time:' format(mrun,4,1) 'minutes;',
         'real time:' format(time('E'),4,1) 'seconds'
exit

/* Generate exponentially distributed service time                 */

genser:  /* GENSER executes same code as GENARR */

/* Generate exponentially distributed inter-arrival time           */

genarr: procedure
 arg theta
 r = random(1,99999)/100000             /* random between (0,1)    */
return -ln(r)/theta                     /* exponential observation */
```

Figure 92 (Part 2 of 2). Single-server Simulation Program

- Average number of elements in service ($\overline{N}_s$)

  The average number of elements in service (asns) is the cumulative service time (cts) divided by the total elapsed time (tna); this is an application of Little's Law.

- Average number of elements in the queue ($\overline{N}_q$)

  The average number of elements in the queue (asnq) is the cumulative queue time (ctq) divided by the total elapsed time (tna); this is an application of Little's Law.

- Average number of elements in system ($\overline{N}_t^1$) and ($\overline{N}_t^2$)

  The average number of elements in the system (asnt) is the sum of the average number of elements in service (asts) and the average number of elements in the queue (asnq). We also calculate the average number in the system at arrival times (past) by dividing the cumulative number of elements after an arrival (cer) by the simulated number of elements (runl); for a Poisson input, these two averages are equal (PASTA property).

- Average size of nonempty queues ($\overline{N}_w$)

  The average size of nonempty queues (astw) is the cumulative queue time (ctq) divided by the cumulative nonempty queue period (cqt). Since the queue times overlap, the technique used to account for the cumulative nonempty queue period is to note the clock (qck) when the queue size (snq) is exactly one after an arrival, and detecting when the queue becomes zero after a departure. Figure 93 shows three nonempty queue periods corresponding to the system described in Table 21 on page 247.

- Average number of elements served in a busy period (E[N])

  The average number of elements served in a busy period (asnb) is the number of simulated elements (runl) divided by the number of busy periods (cbp).

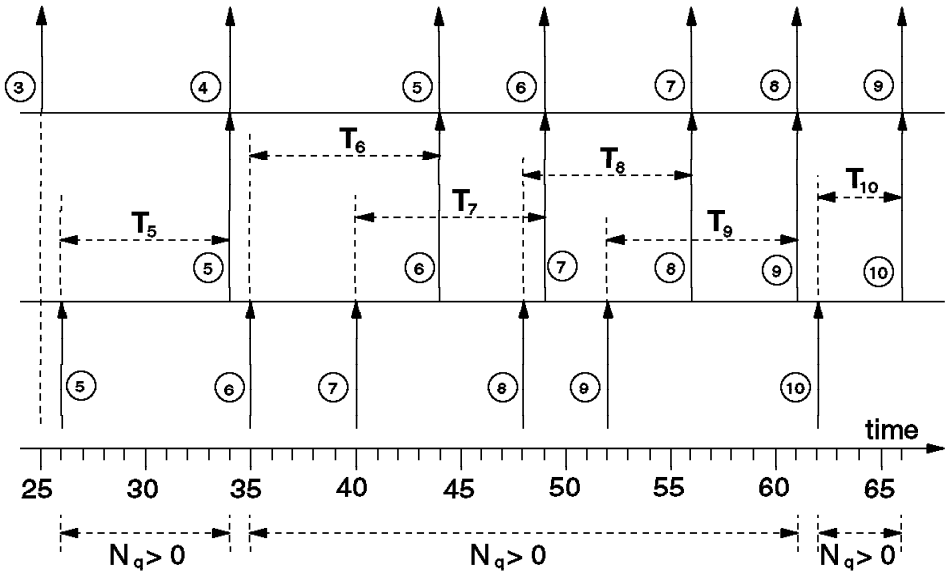

Figure 93. Nonempty Queue Periods

One important point in simulations is deciding when to start and stop the system in order to eliminate transient effects and have a representative sample of measures. One solution is to ignore what happens in the earlier portion of the run after a certain number of elements have been simulated. You could break the run into several intervals and check when the measures in consecutive intervals are close enough.

You should also perform multiple runs (called replications) with a different stream of random numbers (changing the seed of the RANDOM function) and taking the arithmetic average of the replications. If you use the same seed, the random stream is always the same, and therefore, it is a good way to test the simulation model. Selecting a different seed almost certainly produces a different random stream.

We ran the REXX program shown in Figure 92 to simulate 100,000 elements; the output of the run with the default seed (52837) for the RANDOM function is:

```
simulation :   20.071    0.083    0.241
             11.992    3.809   15.802   15.780   49.872   15.809
              0.241    0.076    0.317    0.318    1.319    1.318
M/M/1 model:   20.000    0.083    0.240
             12.000    3.789   15.789   15.789   50.000   15.789
              0.240    0.076    0.316    0.316    1.316    1.316

Simulated elapsed time:  83.0 minutes;  real time:  1.2 minutes
```

The main system parameters are calculated for multiple runs and compared with the M/M/1 model results in Table 22 (all runs simulate 100,000 elements).

The seed for each run is listed in the table so if you run the REXX program using the same seeds, you should get exactly the same results.

Table 22. Simulation Results for M/M/1 Model

Parameters	Run #1	Run #2	Run #3	Run #4	Average	M/M/1
Seed	52837	83616	73523	96158	-	-
l	20.071	19.963	19.949	20.018	20.000	20.000
m	0.083	0.084	0.083	0.083	0.083	0.083
r	0.239	0.241	0.239	0.240	0.240	0.240
$\overline{T}_s$	11.992	11.963	12.020	11.999	11.994	12.000
$\overline{T}_q$	3.809	3.726	3.730	3.808	3.768	3.789
$\overline{T}_r$	15.802	15.687	15.751	15.806	15.762	15.789
$\overline{T}_w$	15.780	15.687	15.621	15.821	15.727	15.789
$E[I]$	49.872	49.994	50.064	49.986	49.979	50.000
$E[B]$	15.809	15.704	15.791	15.802	15.777	15.789
$\overline{N}_s$	0.241	0.239	0.240	0.240	0.240	0.240
$\overline{N}_q$	0.076	0.074	0.074	0.076	0.075	0.076
$\overline{N}_t^1$	0.317	0.313	0.314	0.316	0.315	0.316
$\overline{N}_t^2$	0.318	0.311	0.312	0.316	0.314	0.316
$\overline{N}_w$	1.319	1.305	1.300	1.317	1.307	1.316
$E[N]$	1.318	1.312	1.314	1.317	1.315	1.316

Natural Logarithms

Generating exponentially distributed random variables involves the calculation of natural logarithms of uniformly distributed random numbers, as shown by equation (161) on page 245. The LN function in the RETURN statement of routines GENSER and GENARR is a call to a natural logarithm function.

The standard REXX language implementation does not provide such an intrinsic function. One solution is to code a REXX function to evaluate the natural logarithm. An example of such a function is shown in Figure 94. However, this is quite slow and a simulation run would require a long time. A better solution (actually implemented to run the example shown in Figure 92 on page 250) is to code the logarithm function in Assembler Language and install it as a REXX function.

```
/* Calculate natural logarithm of any number greater than zero */

ln: procedure
 arg x
 if x <= 0 then return '!'
 ln10 = 2.30258509       /* natural logarithm of 10 */
 select
   when x < 1 then do w = -1 by -1 until x >= 1
     x = x*10
   end
   when x >= 10 then do w = 1 until x < 10
     x = x/10
   end
   otherwise w = 0
 end
 z = 0
 y = (x-1)/(x+1)
 do k = 1
    z1 = z + 2/(2*k-1)*y**(2*k-1)
    if z1 = z then return z + w*ln10
    z = z1
 end k
```

Figure 94. Natural Logarithm Function

Other Models

M/M/1/K systems can be simulated by changing the program to discard any arrival when the number of elements in the system (snt) is equal to *K*. In these cases, the program should generate and discard arrivals until the arrival time becomes equal to or greater than the departure time of the element in service.

M/M/1//M systems can be simulated by changing the call to the GENARR routine, passing a different arrival rate according to the number of elements already in the system.

G/G/1 systems can be simulated by coding GENARR and GENSER routines using the corresponding distributions you want to simulate.

Index

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

P

Q

R

S

T

U

V

W

X

ITSO Technical Bulletin Evaluation RED000

VM/ESA Storage Management with Tuning Guidelines

Publication No. GG24-3934-01

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:
   Do you provide billable services for 20% or more of your time? Yes____ No____
   Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____
   If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

Name _____ Address _____

Company or Organization _____

Phone No. _____

IBMLÒ

Cut or Fold Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold Along Line

IBMLÒ

Printed in U.S.A.

GG24-3934-01