POWER® Processor-Based Systems RAS

Introduction to Power Systems™ Reliability, Availability, and Serviceability Covering POWER9™ Processor-based Systems with PowerVM®

With updates for POWER8™ Processor-based Systems

June 12th, 2019

IBM Systems Group
Daniel Henderson
Senior Technical Staff Member, RAS
Trademarks, Copyrights, Notices and Acknowledgements

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at [http://www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

- Active Memory™
- AIX®
- POWER®
- POWER Hypervisor™
- Power Systems™
- Power Systems Software™
- Power®
- POWER®
- POWER6®
- POWER7®
- POWER7+™
- POWER8™
- PowerLinux™
- PowerHA®
- PowerVM®
- System x®
- System z®
- PowerVC™
- POWER Hypervisor™
- Power Architecture™

Additional Trademarks may be identified in the body of this document.

Other company, product, or service names may be trademarks or service marks of others.

**Notices**

The last page of this document contains copyright information, important notices and other information.

**Acknowledgements**

While this whitepaper has a single principal author/editor it is the culmination of the work of a number of different subject matter experts within IBM who contributed ideas, detailed technical information, and the occasional photograph and section of description.

These include the following:

Table of Contents

Trademarks, Copyrights, Notices and Acknowledgements .......... 2
  Trademarks ................................................................................................................. 2
  Notices ......................................................................................................................... 2
  Acknowledgements ....................................................................................................... 2

Table of Contents ............................................................................................................. 3

Introduction ..................................................................................................................... 7
  Whitepaper Organization ............................................................................................. 7

Section 1: Overview of POWER9 Processor-based systems .......... 8
  Comparative Discussion ............................................................................................... 8
    Figure 1: POWER9 Server RAS Highlights Comparison ........................................... 8

IBM Power Systems E980 ............................................................................................ 10
  Introduction .................................................................................................................. 10
    Figure 2: E980 System Structure ............................................................................... 10
  Overview Compared to POWER8 .................................................................................. 10
  Fabric Bus RAS ............................................................................................................ 11
    Figure 3: POWER9 Two Drawer SMP Cable Connections ....................................... 11
  Clocking ........................................................................................................................ 12
  Internal I/O ................................................................................................................... 12
    Figure 4: I/O Connections ......................................................................................... 12
  NVMe Drives .................................................................................................................. 12
  Processor and Memory RAS ......................................................................................... 13
  Infrastructure RAS ....................................................................................................... 13

Power Systems E950 ..................................................................................................... 14
  Introduction .................................................................................................................. 14
    Figure 5: E950 System Illustration (Four Socket System) .......................................... 14
    Figure 6: E950 Memory Riser .................................................................................... 15
  I/O ................................................................................................................................. 15
    Figure 7: E950 I/O Slot Assignments ....................................................................... 15
  PCIe CAPI 2.0 ............................................................................................................... 16
  DASD Options ............................................................................................................... 16
  Infrastructure ................................................................................................................. 16

IBM Power System™ S914, IBM Power System™ S922, IBM Power System™ S924 ... 17
  Introduction .................................................................................................................. 17
  System Features .......................................................................................................... 17
    Figure 8: Simplified View of 2 Socket Scale-Out Server ............................................ 18
  I/O Configurations ........................................................................................................ 18
    Figure 9: Scale-out Systems I/O Slot Assignments for 1 and 2 Socket Systems ....... 18
  Infrastructure and Concurrent Maintenance ................................................................. 19

PCle Gen3 I/O Expansion Drawer .................................................................................... 21
  Figure 10: PCIe Gen3 I/O Drawer RAS Features ......................................................... 21
  Figure 11: PCIe Gen3 I/O Expansion Drawer RAS Feature Matrix ........................... 22

Section 2: Reference for POWER8 Processor Based Systems .......... 23
  Introduction .................................................................................................................. 23
    Figure 12: Comparing Power System RAS Features ................................................. 24
Section 3: General RAS Philosophy and Architecture .......................... 33

Philosophy ......................................................................................... 33
Integrate System Design ..................................................................... 33
Figure 16: IBM Enterprise System Stacks ........................................... 34
Incorporate Experience ...................................................................... 34
Keep Error Detection as a First Priority ............................................ 34
Leverage Technology and Design for Soft Error Management .......... 35
Deploy Strategic Spare Capacity to Self-Heal Hardware ................... 36
Focus on OS Independence ............................................................... 36
Build System Level RAS Rather Than Just Processor and Memory RAS 36

Error Reporting and Handling ............................................................ 37
First Failure Data Capture Architecture ............................................ 37
Figure 17: Errors Classified by Severity and Service Actions Required 38
Processor Runtime Diagnostics ......................................................... 38
Service Diagnostics in the POWER9 Design ...................................... 38
Processor Module Design and Test .................................................... 39
Partitioning with PowerVM .............................................................. 39

Section 4: Subsystems RAS Details ..................................................... 41

Processor RAS Details ...................................................................... 41
Figure 18: POWER6 Processor Design Compared to POWER9 ............ 41
Figure 19: Error Handling Methods Highlights ................................. 42

POWER 9 Memory RAS ..................................................................... 46
Memory RAS Fundamentals ............................................................... 46
Model Comparative Memory RAS ..................................................... 46
Figure 20: Memory Design Points ..................................................... 46
Memory Design Basics ..................................................................... 46
Figure 21: Simplified General Memory Subsystem Layout for 64 Byte Processor Cache Line 47
Figure 22: One Method of Filling a Cache Line Using 2 x8 Industry Standard DIMMs 48
POWER Traditional Design .............................................................. 48
1s and 2s Scale-out Design ............................................................... 48
Figure 23: 1s and 2s Memory Illustrated ............................................ 49
E950 Design Point ............................................................................ 49
DRAM Row Repair .......................................................................... 49
E980 Design Point ............................................................................ 50
Figure 24: POWER8 Memory Subsystem .......................................... 50
Memory RAS Beyond ECC ................................................................. 50
Hypervisor Memory Mirroring ....................................................... 50
  Figure 25: Active Memory Mirroring for the Hypervisor .......... 51
Dynamic Deallocation/Memory Substitution ................................. 51

RAS beyond Processors and Memory ............................................. 52
Introduction .................................................................................. 52
  Serial Failures, Load Capacity and Wear-out .............................. 52
  Common Mode Failures ............................................................. 52
Power and Cooling Redundancy Details .......................................... 53
  Power Supply Redundancy ......................................................... 53
  Voltage Regulation .................................................................. 53
Redundant Clocks .......................................................................... 54
Internal and External Cache Coherent Accelerators .......................... 55
Service Processor and Boot Infrastructure ........................................ 55
Trusted Platform Module (TPM) ...................................................... 55
I/O Subsystem and VIOS™ .............................................................. 55
  Figure 26: End-to-End I/O Redundancy .................................. 56
  PCIe Gen3 Expansion Drawer Redundancy ................................. 56
  Figure 27: Maximum Availability with Attached I/O Drawers ..... 57
Planned Outages ........................................................................... 57
Updating Software Layers ............................................................... 57
Concurrent Repair ......................................................................... 58
Integrated Sparing .......................................................................... 58

Clustering and Cloud Support .......................................................... 59
PowerHA SystemMirror ................................................................. 59
  Live Partition Mobility .............................................................. 59
  Figure 28: LPM Minimum Configuration ................................. 59
Minimum Configuration ................................................................. 60
  Figure 29: I/O Infrastructure Redundancy ................................. 60
  Figure 30: Use of Redundant VIOS ......................................... 61
PowerVC™ and Simplified Remote Restart ...................................... 61
Error Detection in a Failover Environment ....................................... 61

Section 5: Reliability and Availability in the Data Center ............... 63
The R, A and S of RAS .................................................................. 63
  Introduction ............................................................................. 63
  RAS Defined ............................................................................ 63
Reliability Modeling ...................................................................... 63
  Figure 31: Rough Reliability Cheat Sheet* ............................. 64
Different Levels of Reliability ......................................................... 64
Costs and Reliability ..................................................................... 65
  Service Costs .......................................................................... 65
  End User Costs ....................................................................... 65
Measuring Availability .................................................................. 66
  Measuring Availability ............................................................ 66
  Figure 32: Different MTBFs but same 5 9’s of availability ......... 67
  Contributions of Each Element in the Application Stack .......... 67
  Figure 33: Standalone System Availability Considerations ....... 68
Critical Application Simplification ............................................... 69
Measuring Application Availability in a Clustered Environment ....... 69
  Figure 34: Ideal Clustering with Enterprise-Class Hardware Example .... 70
Recovery Time Caution ................................................................. 70
Clustering Infrastructure Impact on Availability ............................. 70
Real World Fail-over Effectiveness Calculations ............................. 71
  Figure 35: More Realistic Model of Clustering with Enterprise-Class Hardware .... 72
Summary............................................................................................................ 75
In Investing in RAS .................................................................................................. 75
Final Word ................................................................................................................ 76
Introduction

Reliability generally refers to the infrequency of system and component failures experienced by a server. Availability, broadly speaking, is how the hardware, firmware, operating systems and application designs handle failures to minimize application outages.

Serviceability generally refers to the ability to efficiently and effectively install and upgrade systems, firmware and applications, as well as to diagnose problems and efficiently repair faulty components when required.

These interrelated concepts of reliability, availability and serviceability are often spoken of as "RAS". Within a server environment all of RAS, but especially application availability, is really an end-to-end proposition. Attention to RAS needs to permeate all the aspects of application deployment. However, a good foundation for server reliability whether in a scale-out or scale-up environment is clearly beneficial.

Systems based on the POWER processors are generally known for their design emphasizing Reliability, Availability and Serviceability capabilities. Previous versions of a RAS whitepaper have been published to discuss general aspects of the hardware reliability and the hardware and firmware aspects of availability and serviceability.

The focus of this whitepaper is POWER9 Processor-based systems using the PowerVM hypervisor. Systems not using PowerVM will not be discussed specifically in this whitepaper. Serviceability aspects are limited to hardware capabilities (such as design for concurrent maintenance.) For details of error log analysis, call-home capabilities, and repair details, the reader is directed to IBM publications such as the POWER9 Knowledge center: https://www.ibm.com/support/knowledgecenter/he/POWER9/p9hdx/POWER9welcome.htm

Whitepaper Organization

This whitepaper is organized into four sections:

Section 1: RAS Overview of POWER9 and POWER8 Processor based systems

An overview of the RAS capabilities of various POWER9 processor based servers using PowerVM and the differences between products. This is a good starting point for readers familiar with the RAS characteristics of earlier generations of POWER processor based systems.

Section 2: POWER8 processor based systems RAS.

Reference information for previous generation POWER8 processor based systems. Supersedes previous POWER8 RAS Whitepaper

Section 3: General Design Philosophy

A general discussion of POWER systems RAS design philosophy, priorities and advantages.

Section 4: Subsystems RAS

A more detailed discussion of each sub-system within a POWER server concentrating on the unique RAS features of processor, memory and other components.

Section 5: Reliability and Availability in the Data Center

Discussion of RAS measurements and expectations, including various ways in which RAS may be described for systems: Mean Time Between Failures, ‘9’s of availability and so forth.
Section 1: Overview of POWER9 Processor-based systems

Comparative Discussion

In February 2018, IBM introduced a set of scale-out servers using a POWER9™ processor technology processor designed for scale-out systems featuring a version of the POWER9 processor designed specifically for the scale-out space.

In August 2018, two additional categories of servers were introduced using a POWER9 processor designed for the scale-up or enterprise market.

The Enterprise or scale-up systems are designed with a POWER9 processor intended for use in the Enterprise 4 processor socket or greater space. It features an improved design for communicating with processors across nodes as well. The processor RAS features include support for the IBM memory buffer chip and advanced memory RAS features including hypervisor memory mirroring.

The POWER9 processor for scale-out systems is optimized for the scale-out environment with different RAS features including support for direct attachment of Industry Standard DIMMs (ISDIMMs).

The figure below is a general comparative highlights overview for the RAS features of these PowerVM-based systems with supporting details found in other sections:

**Figure 1: POWER9 Server RAS Highlights Comparison**

<table>
<thead>
<tr>
<th>Feature</th>
<th>POWER9 1s and 2s Systems^</th>
<th>POWER9 IBM Power System E950</th>
<th>POWER9 IBM Power System E980</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base POWER9™ Processor RAS features including</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• First Failure Data Capture</td>
<td>Yes*</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>• Processor Instruction Retry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• L2/L3 Cache ECC protection with cache line-delete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Power/cooling monitor function integrated into processors’ on chip controllers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• CRC checked processor fabric bus retry with spare data lane</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POWER9 Enterprise RAS Features</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Extended L2/L3 cache line delete</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>• Core Contained Checkstops</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POWER9 Multi-node SMP Fabric RAS</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>PCIe hot-plug with processor integrated PCIe controller</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Memory DIMM ECC supporting x4 Chipkill*</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Uses IBM memory buffer and has spare DRAM module capability with x4 DIMMS*</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>x8 DIMM support with Chipkill correction for marked a DRAM*</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>No</td>
<td>Yes - Feature</td>
<td>Yes - Base</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
<td>---------------</td>
<td>------------</td>
</tr>
<tr>
<td>Custom DIMM support with additional spare DRAM capability*</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Active Memory Mirroring for the Hypervisor</td>
<td>No</td>
<td>Yes - Feature</td>
<td>Yes - Base</td>
</tr>
<tr>
<td>Redundant/spare voltage phases on voltage converters for levels feeding processor and custom memory DIMMs or memory risers.</td>
<td>No</td>
<td>Redundant</td>
<td>Both redundant and spare</td>
</tr>
<tr>
<td>Redundant global processor clocks with concurrent failover</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Redundant service processor and related boot facilities</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Redundant TPM capability</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Multi-node support</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* In scale-out systems Chipkill capability is per rank of a single Industry Standard DIMM (ISDIMM); in IBM Power System E950 Chipkill and spare capability is per rank spanning across an ISDIMM pair; and in the IBM Power System E980, per rank spanning across two ports on a Custom DIMM.

The E950 system also supports DRAM row repair

^ IBM Power System™, S914, IBM Power System™ S922, IBM Power System™ S924
  IBM Power System™, H922 IBM Power System™ S924, IBM Power System™ H924
IBM Power Systems E980

Introduction

The E980 class systems are designed to maximize availability in support of systems with 4, 8, 12 and 16 sockets. The basic design is in many ways similar to the design of the POWER8 870/880 870C/880C described in the POWER8 RAS whitepaper.

There are key differences, however, such as the new fabric bus used to communicate across nodes. The figure below, in a rough fashion only, gives a logical abstraction of the design and highlights some of the changes from POWER8.

Figure 2: E980 System Structure

Overview Compared to POWER8

The main building block of these systems is a 4 processor module CEC drawer supporting connection to up to 32 custom memory DIMMs. The system contains 8 PCIe slots connecting the processors. These slots support PCIe GEN4 as opposed to PCIe GEN3 in the POWER8 Systems.

New for POWER9, internal NVMe drives have been added as storage for limited use situations. These are sold in pairs. Each CEC drawer may have zero, two or four drives.

Each system also contains a System Control Unit drawer. This drawer, powered from the first CEC drawer (and second if present) has dual global flexible service procesor cards as well as system VPD and...
an operator panel. Instead of containing an optional DVD drive, the System Control Drawer has a USB port for connecting an external USB drive.

Aside from the addition of the NVMe drives, the most obvious change is that the system control drawer no longer has system clock cards. This is due to a fundamental design change in the POWER9 processor where the POWER9 processor no longer requires that the processors have clock synchronization across CEC drawers. Therefore each system drawer incorporates two clock cards, eliminating separate global clock cards in the system control drawer.

The rest of this section will discuss the structural differences between these systems and POWER8, then go into the RAS details about the rest of the design.

**Fabric Bus RAS**

The SMP interconnect fabric of a POWER processor is the logic used to communicate between processors and components that share in the SMP cache coherency of the system.

Each fabric bus connecting a processor is comprised of two links. Data is CRC checked and a retry mechanism handles the occasional soft error. If multiple retries are required due to a solid fault that impacts a single lane of data on the link, the hardware can switch to a spare data lane provided for each link.

Because each bus is composed of two links, if one link fails, bus traffic may be retried/rerouted to use the second link only (in a lane reduction or half bandwidth mode.) This mechanism can prevent certain outages that would require a system reboot on a POWER8 system.  

![Figure 3: POWER9 Two Drawer SMP Cable Connections](image)

One fabric bus is composed of two links across two separate cables.

Also significant for the fabric busses that go across nodes, each link uses a separate active SMP cable. This is intended to support concurrent repair of a failed SMP cable, both in tolerating a fault (with reduced bus performance) and allowing repair of the cable without taking an outage.

While the new fabric bus design has a number of RAS advantages as described, it does mean more cables to install for each bus compared to the POWER8 design. The POWER9 design addressed this with a cable installation process using LEDs at the pull-tab cable connections to guide the servicer through a cable plugging sequence,

---

1. Running in half-bandwidth mode reduces performance not only for customer data but also hypervisor and other firmware traffic across the bus. Because of this, there are limitations in the mid-November 2018 and earlier firmware as to which connections and how many busses can be at half-bandwidth without causing a system impact.

2. Supported by at least firmware level 930
**Clocking**

The POWER8 multi-node system required that each processor in the system use a single processor reference clock source. The POWER8 processor could take in two different sources, providing for redundancy, but each source had to be the same for each processor across all 4 CEC drawers.

This meant that the system control drawer contained a pair of “global” clock cards and the outputs of each card had to be cabled to each CEC drawer in the system through a pair of local clock and control cards.

In POWER9 each E980 CEC drawer does not require a separate clock source. Consequently, there are no global system clock cards in the system control drawer. Instead each node has processor clock logic added to the local clock and control cards. In this fashion the clocking continues to be redundantly supplied to each processor while eliminating separate global clock cards and cables.

It should be noted that the local clock and control card also supplies redundant inputs for the processor “multi-function” clock input used for such items as PCIe clocking. Failover to the redundant clock is provided dynamically. Failover will be seen by adapter device drivers as PCIe EEH recoverable events.

**Internal I/O**

As in the POWER8 design, each processor internally uses PHBs to provide x16 lanes for each of two PCIe busses, for a total of 8 PCIe slots. These slots, however, support Gen4 instead of Gen3 PCIe.

In addition to supporting internal adapters, each of the Internal PCIe slots can also use a card that provides optical connectivity to an external I/O drawer: currently the same Gen3 I/O drawer offered with POWER8.

In addition, as previously described processors 2 and 3 use another host bridge to provide connectivity to the NVMe drives.

Processor 1 uses an additional PHB to connect to a new USB card.

This card provides 3 USB connections. One of these is routed through a cable to the System Control Drawer, re-driven to a connector on the Operator panel to allow for attachment of an external DVD drive.

**Figure 4: I/O Connections**

<table>
<thead>
<tr>
<th>Processor</th>
<th>Slots (Gen4 x16)</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proc0</td>
<td>P1-C1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1-C2</td>
<td></td>
</tr>
<tr>
<td>Proc1</td>
<td>P1-C7</td>
<td>USB 3.0 Card</td>
</tr>
<tr>
<td></td>
<td>P1-C8</td>
<td></td>
</tr>
<tr>
<td>Proc2</td>
<td>P1-C3</td>
<td>NVMe Drive 1 and Drive 2</td>
</tr>
<tr>
<td></td>
<td>P1-C4</td>
<td></td>
</tr>
<tr>
<td>Proc3</td>
<td>P1-C5</td>
<td>NVMe Drive 3 and Drive 4</td>
</tr>
<tr>
<td></td>
<td>P1-C6</td>
<td></td>
</tr>
</tbody>
</table>

**NVMe Drives**

POWER8 Enterprise systems CEC drawers had no internal storage. This was primarily due to the expectation that Enterprise customers would make use of a Storage Area Network (SAN) for both user data availability as well as the storage used by VIOS which provided virtualized I/O redundancy.

As an alternative, external DASD drawers were also available in POWER8. However, feedback from a number of customers expressed a preference for using for the VIOS root volume group even when a SAN was deployed for everything else.
To accommodate this need, POWER9 Enterprise systems have the option of internal NVMe drives for such purpose as VIOS rootvg storage. Each CEC drawer in the E980 system has 4 NVMe drive slots. The drives are sold in pairs, with the expectation that the data on each drive will be mirrored.

The drives are connected to processors two and three using PCIe busses from each as shown earlier.

To maximize availability, a drive and its redundant pair should be controlled by separate processors in single drawer systems. In multi-drawer systems different configurations might be used to maintain availability across multiple VIOS.

One such method of maximizing availability would have redundant VIOS partitions and each VIOS partition have mirrored drives not controlled by the same processor.

The NVMe drives support concurrent maintenance. They are connected to the CEC drawer backplane through a separate backplane and riser card that are not concurrently maintained.

**Processor and Memory RAS**

Section three of this paper will describe the processor and memory RAS characteristics of this system.

**Infrastructure RAS**

As illustrated earlier, like the POWER8 system, the POWER9 CEC drawers and system control drawers have a highly redundant infrastructure design.

Regardless of the number of CEC drawers, service processor redundancy is provided. Each CEC drawer (or node) now has redundant processor clocks; additional redundant service infrastructure support; an alternate boot processor module and boot firmware module; and redundant trusted platform module (TPM) cards. The E980 is the only POWER9 system that has these redundant components.

In addition, in the CEC drawers, the Power supplies provide at least n+1 power supply redundancy and support line-cord redundancy. Fans provide at least n+1 redundancy as well. The power supplies and fans also support concurrent maintenance.

I/O adapters support concurrent maintenance as well and can be made redundant using facilities such as VIOS as previously discussed. The same concurrent maintenance capabilities for the I/O drawer and I/O cables are unchanged from POWER8.

The system control drawer gets power from up to two CEC nodes, providing redundancy. The fans in the system control drawer also support concurrent maintenance.

The Operator Panel design is new compared to POWER8. The Operator Panel is composed of two parts: a base part, and a separate LED display. It also supports a USB connection. The full operator panel assembly is still concurrently maintainable.

The FSP card has been somewhat re-designed; the most visible change being that the cables used to connect the FSP to the CEC drawer are no longer on the FSP cards themselves.

This was done for serviceability reasons. An FSP card installation now incorporates LEDs as part of a visual validation step. Customer functionality of the FSP has not really changed, and the batteries for the real-time clock used by each FSP is also still concurrently maintainable.
Power Systems E950

Introduction

The E950 base system design is flexible to support two or four processors with multiple options for internal storage.

From a RAS standpoint the system design is somewhat different from the POWER8 comparable system. Two items stand out in that regard: greater integration eliminating a separate I/O planar and a new design for supporting industry standard DIMMS while maintaining the advantages of the IBM memory buffer.

The figure below gives a rough idea of the system configuration and capabilities regarding redundancy and concurrent maintenance.

**Figure 5: E950 System Illustration (Four Socket System)**

The memory Risers in the illustration above can be better illustrated by the following:
Figure 6: E950 Memory Riser

The circled portion of the memory riser and DIMMS would be replaced by a single custom enterprise DIMM (CDIMM) in the E980 design.

In addition to not being as densely packaged as the CDIMM design, there are some differences in RAS capabilities with the E850 memory which will be discussed in the next section on processor and memory RAS.

I/O

Figure 5 indicates that there are multiple I/O adapter slots provided. Some are directly connected to the processor and others connected through a PCIe switch. There is a connection for a USB 3.0 adapter.

Generally, I/O adapters are packaged in cassettes and are concurrently maintainable (given the right software configuration.)

Like the E980, two processors provide connections to the NVMe drives (as illustrated). In addition, the system provides connectivity for SAS DASD devices. These are connected using SAS adapters that can be plugged into slots C12 and C9.

Figure 7: E950 I/O Slot Assignments

<table>
<thead>
<tr>
<th>Slot (if any)</th>
<th>Type</th>
<th>From</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>GEN4 x16</td>
<td>Proc 3</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0,</td>
</tr>
<tr>
<td>C3</td>
<td>GEN4 x16</td>
<td>Proc 3</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0</td>
</tr>
<tr>
<td>C4</td>
<td>GEN4 x16</td>
<td>Proc 2</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0</td>
</tr>
<tr>
<td>C5</td>
<td>GEN4 x16</td>
<td>Proc 2</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0</td>
</tr>
<tr>
<td>Slot</td>
<td>Interface</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>GEN3 x8</td>
<td>Proc 1 through a Switch</td>
<td>Reserved for Ethernet Adapter</td>
</tr>
<tr>
<td></td>
<td>GEN3 x4</td>
<td>Proc 1 through a Switch</td>
<td>NVMe Drive 2</td>
</tr>
<tr>
<td></td>
<td>GEN3 x4</td>
<td>Proc 1 through a Switch</td>
<td>NVMe Drive 4</td>
</tr>
<tr>
<td>C7</td>
<td>GEN4 x16</td>
<td>Proc 1</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0</td>
</tr>
<tr>
<td>C8</td>
<td>GEN4 x16</td>
<td>Proc 1</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0</td>
</tr>
<tr>
<td>C9</td>
<td>GEN4 x8</td>
<td>Proc 1</td>
<td>PCIe Adapters, SAS Adapter for Internal SAS</td>
</tr>
<tr>
<td>C10</td>
<td>GEN4 x16</td>
<td>Proc 0</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0</td>
</tr>
<tr>
<td>C11</td>
<td>GEN4 x16</td>
<td>Proc 0</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0</td>
</tr>
<tr>
<td></td>
<td>GEN3 x4</td>
<td>Proc 0 through a Switch</td>
<td>NVMe Drive 1</td>
</tr>
<tr>
<td></td>
<td>GEN3 x4</td>
<td>Proc 0 through a Switch</td>
<td>NVMe Drive 3</td>
</tr>
<tr>
<td>C13</td>
<td>GEN3 x1</td>
<td>Proc 0 through a switch</td>
<td>Internal USB 3.0 Card (Dedicated)</td>
</tr>
<tr>
<td>C10</td>
<td>GEN4 x16</td>
<td>Proc 0</td>
<td>PCIe Adapters, Cable card for I/O expansion, CAPI 2.0</td>
</tr>
<tr>
<td>C12</td>
<td>GEN4 x8</td>
<td>Proc 0</td>
<td>PCIe Adapters, SAS Adapter for Internal SAS</td>
</tr>
</tbody>
</table>

**PCIe CAPI 2.0**

The main illustration for the E950 does not show any PCIe CAPI 2.0 capabilities. The figure above does indicate which slots support PCIe CAPI 2.0 adapters. It should be noted that CAPI adapters may not be concurrently maintainable.

**DASD Options**

Like the E980, internal NVMe drives are provided for VIOS rootvg purposes. The Internal SAS drives can be used for partition and other data.

There are two backplane options available for the SAS drives. The base option allows for connection by one SAS adapter, or use of two SAS adapters with a split-backplane function.

**Infrastructure**

N+1 power supply redundancy and n+1 fan rotor redundancy are supported. Voltage regulator outputs for voltages going to processors and to the memory riser are provided at n+1 level meaning there is one more output phase provided than is needed.

New for the POWER9, the operator panel has a split design with a separate hot-pluggable LCD display.
IBM Power System™ S914, IBM Power System™ S922, IBM Power System™ S924

Introduction

The systems described in this section are meant-for scale-out applications. The POWER9 processor in the POWER9 family used in these systems has features specifically for the 1s and 2s scale-out servers with some changes from the processor used in the scale-up systems described previously. In addition to not supporting multiple nodes, the most visible change is support for direct-attachment of industry standard DIMMS rather than interfacing to memory through a buffer module.

The basic building block of a processor contains 2 cores each supporting 4 SMT threads. These two cores share an L2 and an L3 cache. On these scale-out servers these cores may be “fused” together to support a single 8 SMT thread fused core, or not fused to support up to 24 4 SMT thread cores.

System Features

System RAS feature highlights for the IBM Power System S914, IBM Power System S922, IBM Power System H922, IBM Power System S924 and IBM Power System H924 servers are designed to run with the IBM POWER Hypervisor™ and support AIX®, Linux™ and IBM i operating systems.

System feature highlights include:

- Power supply and fan cooling redundancy supporting concurrent maintenance
- Concurrent Maintenance for PCI adapters and internal disk-drives on the DASD backplane
- Most System service can be performed without the need to take the server off the sliding rails
- Direct Attach Industry Standard DIMMS as previously mentioned

Systems contain either one or two processor sockets with different tower and packaging options as well as I/O and DASD capabilities.
Figure 8: Simplified View of 2 Socket Scale-Out Server

Note: There are differences in I/O connectivity between different models and there are multiple DASD backplane options.

**I/O Configurations**

Like the E950, internal I/O slots have a mixture of slots directly attached to processors and those connected through a switch.

Figure 9: Scale-out Systems I/O Slot Assignments for 1 and 2 Socket Systems

<table>
<thead>
<tr>
<th>Slot (if any)</th>
<th>Type</th>
<th>In 1 Processor Socket Systems</th>
<th>In 2 Processor Socket Systems</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>Gen4 x8 with x16 connector</td>
<td>Not Present</td>
<td>Proc 1</td>
<td></td>
</tr>
<tr>
<td>Slot</td>
<td>Type</td>
<td>Source 1</td>
<td>Source 2</td>
<td>Remarks</td>
</tr>
<tr>
<td>------</td>
<td>--------------------</td>
<td>----------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>C3</td>
<td>Gen4 x16</td>
<td>Not Present</td>
<td>Proc 1</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>Gen4 x16</td>
<td>Not Present</td>
<td>Proc 1</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>Gen3 x8 or USB Connector</td>
<td>From Proc 0 through switch 0</td>
<td>From Proc 0 through switch 0</td>
<td>On 1S is a PCIe Adapter connector on 2S provides a USB function (integrated)</td>
</tr>
<tr>
<td>C6</td>
<td>Gen3 x8 with x16 connector</td>
<td>From Proc 0 through switch 0</td>
<td>Proc 0 through a switch</td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>Gen3 x8</td>
<td>From Proc 0 through switch 0</td>
<td>Proc 0 through a switch</td>
<td></td>
</tr>
<tr>
<td>C8</td>
<td>Gen4 x8 with x16 connector</td>
<td>From Proc 0 through switch 0</td>
<td>Proc 0</td>
<td></td>
</tr>
<tr>
<td>C9</td>
<td>Gen4 x16</td>
<td>From Proc 0</td>
<td>Proc 0</td>
<td></td>
</tr>
<tr>
<td>C10</td>
<td>Gen3 x8 (1s) or SAS connector (2S)</td>
<td>From Proc 0 through switch 1</td>
<td>From Proc 1 through switch 1</td>
<td>PCIe Adapters for 1s and provides a SAS function (integrated) on 2S</td>
</tr>
<tr>
<td>C11</td>
<td>Gen3 x8</td>
<td>From Proc 0 through switch 1</td>
<td>From Proc 1 through switch 1</td>
<td>PCIe Adapters. Is default slot for LAN on 2S systems</td>
</tr>
<tr>
<td>C12</td>
<td>Gen3 x8 with x16 connector</td>
<td>From Proc 0 through switch 1</td>
<td>From Proc 1 through switch 1</td>
<td>PCIe Adapters</td>
</tr>
<tr>
<td>C49</td>
<td>Gen3 x8</td>
<td>From Proc 0 through switch 1</td>
<td>Proc 0 through a switch</td>
<td>M.2 NVMe Drive Carrier 2 or 2nd SAS Adapter Card</td>
</tr>
<tr>
<td>C50</td>
<td>Gen3 x8</td>
<td>From Proc 0 through switch 0</td>
<td>Proc 0 through a switch</td>
<td>M.2 NVMe Drive Carrier 1 or 1st SAS Adapter Card</td>
</tr>
<tr>
<td>Integrated</td>
<td>Gen4 x1</td>
<td>Proc 0</td>
<td>Proc 0</td>
<td>USB 3.0 4dimm(Integrated)</td>
</tr>
</tbody>
</table>

Other documentation should be referred to for details on I/O options including card profiles and power requirements.

As a general rule, for maximum redundancy it is important to understand that for certain faults within a processor, all of the I/O underneath the processor may not be accessible. Likewise, when a PCIe switch is also incorporated into the path, certain switch faults can impact all the I/O underneath a switch.

In the above illustration specific slots are dedicated for SAS adapters. The C49 and C50 slots can be used either for an NVME M.2 adapter or for SAS adapters. When used for SAS adapters these slots have connections between them for communication in a dual SAS adapter environment.

For the internal SAS adapters there are options to use two SAS adapters with a single backplane or with a split DASD backplane. Using two I/O adapters provides maximum redundancy.

Note also that certain slots do support a cable card used to connect the system to an external I/O drawer. The I/O drawer RAS is discussed later.

**Infrastructure and Concurrent Maintenance**

Different models/configurations will have different fan options. In general, power supply and fan rotor redundancy will be at least n+1 meaning that the system will continue to operate when one element fails.

The capability of concurrently replacing a power supply or fan element is also provided.

Additionally, generally speaking I/O adapters can be concurrently maintained from a system standpoint. The need to halt the I/O on the component being maintained and/or make use of I/O redundancy is still required. The SAS drives can also be maintained in a similar way.
Due to system requirements the SAS adapters, GPU cards, CAPI cards, and the NVMe M.2 carrier cards, and the M.2 flash modules on the carrier cards are not concurrently maintainable.

In these systems the base component of the Operator Panel is standard on all configurations and is not concurrently maintainable. Use of the hot-pluggable LCD display component is optional for some configurations. Service requires there to be at least 1 LCD display per rack for rack-mounted systems.
PCIe Gen3 I/O Expansion Drawer

PCIe Gen3 I/O Expansion Drawers can be used in systems to increase I/O capacity.

Figure 10: PCIe Gen3 I/O Drawer RAS Features

*Optical Connections to CEC Drawer*

These I/O drawers are attached using a connecting card called a PCIe3 cable adapter that plugs in to a PCIe slot of the main server. In **POWER9** these cable cards have been redesigned in certain areas to improve error handling. These improvements include new routing for clock logic within the cable card as well as additional recovery for faults during IPL.

Each I/O drawer contains up to two I/O drawer modules. An I/O module uses 16 PCIe lanes controlled from a processor in a system. Currently supported is an I/O module that uses a PCIe switch to supply six PCIe slots.

Two different active optical cable are used to connect a PCIe3 cable adapter to the equivalent card in the I/O drawer module. While these cables are not redundant, as of FW830 firmware or later; the loss of one cable will simply reduce the I/O bandwidth (number of available lanes available to the I/O module) by 50%.

Infrastructure RAS features for the I/O drawer include redundant power supplies, fans, and DC outputs of voltage regulators (phases).

The impact of the failure of an I/O drawer component can be summarized for most cases by the table below.
Figure 11: PCIe Gen3 I/O Expansion Drawer RAS Feature Matrix

<table>
<thead>
<tr>
<th>Faulty Component</th>
<th>Impact of Failure</th>
<th>Impact of Repair</th>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O adapter in an I/O slot</td>
<td>Loss of function of the I/O adapter</td>
<td>I/O adapter can be repaired while rest of the system continues to operate.</td>
<td>Multipathing/ I/O adapter redundancy, where implemented, can be used to prevent application outages</td>
</tr>
<tr>
<td>First Fault on a Data lane (in the optics between the PCIe3 cable adapter card in the system and the I/O module)</td>
<td>None: Spare Used</td>
<td>No Repair Needed: Integrated Sparing Feature</td>
<td></td>
</tr>
<tr>
<td>A Second Data Lane fault or other failure of one active optics cable</td>
<td>System continues to run but the number of active lanes available to I/O module will be reduced</td>
<td>Associated I/O module must be taken down for repair; rest of the system can remain active.</td>
<td></td>
</tr>
<tr>
<td>Other Failure of PCIe3 cable adapter card In System or I/O module</td>
<td>Loss of access to all the I/O of the connected I/O Module</td>
<td>Associated I/O module must be taken down for repair; rest of the system can remain active.</td>
<td>Systems with an HMC</td>
</tr>
<tr>
<td>One Fan</td>
<td>System continues to run with remaining fan</td>
<td>Concurrently repairable</td>
<td></td>
</tr>
<tr>
<td>One Power supply</td>
<td>System continues to run with remaining fan</td>
<td>Concurrently repairable</td>
<td></td>
</tr>
<tr>
<td>Voltage Regulator Module associated with an I/O module</td>
<td>System continues to run for a phase failure transition to n mode. Other faults will impact all the I/O in the Module</td>
<td>Associated I/O module cannot be active during repair; rest of the system can remain active.</td>
<td>Systems with an HMC</td>
</tr>
<tr>
<td>Chassis Management Card (CMC)</td>
<td>No Impact to running system, but once powered off, the I/O drawer cannot be re-integrated until CMC is repaired.</td>
<td>I/O drawer must be powered off to repair (loss of use of all I/O in the drawer)</td>
<td>Systems with an HMC</td>
</tr>
<tr>
<td>Midplane</td>
<td>Depends on source of failure, may take down entire I/O drawer</td>
<td>I/O drawer must be powered off to repair (loss of use of all I/O in the drawer)</td>
<td>Systems with an HMC</td>
</tr>
</tbody>
</table>
Section 2: Reference for POWER8 Processor Based Systems

In response to requests for information concerning the previous generation of POWER8 processor systems, this section updates information from the prior POWER8 RAS whitepaper.

Introduction

POWER8 processor-based systems using the IBM flexible service processor can be grouped into several categories with similar RAS features aligning with the maximum number of processor sockets supported in each system.

The first group includes 1 and 2 socket systems (1s and 2s) that are designed for a scale-out environment. Other groups include 4 socket systems and systems with 4 socket building-blocks that can be interconnected to create systems with 8, 16, 24, 32 or more processor sockets.

From a RAS standpoint, all systems are built on a strong POWER processor base. The preceding subsection summarized the RAS characteristics of these features and the POWER8 processor/memory improvements.

Enterprise-level RAS is more involved than just processor and memory RAS. The structure of the systems using these components is also critical in avoiding application outages.

Power Systems RAS characteristics are designed to be suitable to the size of the system and intended use. Figure 12 illustrates some of different Power Systems offered and their associated RAS characteristics.
### Figure 12: Comparing Power System RAS Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>POWER7 1s,2s Servers and IBM Power Systems 750/760</th>
<th>POWER8 1s and 2s Systems</th>
<th>POWER8 IBM Power Systems E850/E850c</th>
<th>POWER8 IBM Power Systems E870/E870c E880/E880c</th>
</tr>
</thead>
<tbody>
<tr>
<td>POWER7+ Processor soft error protection, recovery and self-healing</td>
<td>Yes</td>
<td>Yes *</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>New POWER8 Processor features including:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Integrated PCIe controller</td>
<td>No</td>
<td>Yes *</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>- Integrated Power/cooling monitor function using on chip controller</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Memory buffer replay</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCIe hot-plug</td>
<td>750/760</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PCIe controllers integrated into processor</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Enterprise Memory with custom DIMMS, Chipkill and spare DRAM capabilities, memory buffer soft error handling features</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Active Memory Mirroring for the Hypervisor</td>
<td>No</td>
<td>No</td>
<td>Supported as an Option</td>
<td>Yes in base configuration</td>
</tr>
<tr>
<td>Can take advantage of Capacity on Demand</td>
<td>No</td>
<td>No</td>
<td>Available Option</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic Substitution of Unused memory for predictive memory faults</td>
<td>No</td>
<td>Yes+*</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Triple Redundant Ambient Temperature Sensors on Op Panel</td>
<td>No</td>
<td>Yes+</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Redundant and spare voltage converter phases supplied to processors and to CDIMMS</td>
<td>No</td>
<td>No</td>
<td>Redundant or Spare</td>
<td>Both redundant and spare</td>
</tr>
<tr>
<td>Redundant global processor clocks</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Redundant service processor</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can re-IPL with one node even when an entire node must be de-configured due to a fault</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes on multiple node systems</td>
</tr>
<tr>
<td>Supports Enterprise System Pools</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*1s and 2s systems in this table include:
AIX® IBM and Linux™: IBM Power S812, IBM Power S822 and IBM Power S824
Linux Only: IBM Power S812L, IBM Power S822L and IBM Power S824L

See Figure 14 for information about POWER7 S812LC and S812LC

*Not all features are supported with PowerKVM™
+Support depends on firmware level installed in system

**POWER8 processor-based 1s and 2s Systems IBM POWER System S814, S822, S824**

The one processor module socket (1s) IBM Power System S814 server, and two processor module socket (2s) IBM Power System S822 and IBM Power System S824 servers are designed to run with the IBM POWER™ Hypervisor and support AIX®, Linux™ and IBM i operating systems.
Those familiar with POWER7 servers will see that these servers are aimed at replacing POWER7 processor-based 1s and 2s servers. Typically, 1s and 2s systems may be thought of as “scale-out” systems designed to run a set of enterprise applications in concert with other similar systems in some form of clustered environment.

Responding to the increased performance and capacity, these 1s and 2s systems were designed with enhancements to both Processor and System platform level RAS characteristics compared to predecessor POWER7 and POWER7+ processor-based systems.

Outside of the processor itself, perhaps the two most noticeable system enhancements are:

- The ability to remove/replace PCIe adapters without the need to shut down the system or terminate partitions
- The use of what was previously considered to be Enterprise Memory using custom DIMMs with an IBM memory-buffer chip on-board each DIMM and featuring spare DRAM modules

The first feature reduces planned outages for repair of I/O adapters. The second is intended to reduce planned outages for repair of DIMMs as well as unplanned outages due to DIMM faults.

**POWER8 processor-based 1s and 2s Systems IBM POWER System S812L, S822L and S824L**

From a hardware standpoint The IBM Power System S812L, S822L and IBM Power System S824L are similar to the servers described in the previous section, though they are designed to run Linux exclusively.

As an alternative to the POWER Hypervisor and PowerVM® these systems can be configured for open virtualization using the IBM PowerKVM™ Power Kernel-Based Virtual Machine to provide a hypervisor function. PowerKVM in turn uses the OpenPower Abstraction Layer (OPAL) for certain hardware abstraction functions.

Systems running PowerKVM do not have identical virtualization features as those running PowerVM. Generally speaking processor and memory error detection, fault isolation and certain self-healing features are handled entirely within the hardware and dedicated service processor and are available in Power Systems regardless of the hypervisor deployed.

Certain functions that require notification of the hypervisor but do not require interaction with operating systems are also implemented in systems running with PowerKVM as well. Such features include processor instruction retry.

Some functions are not currently supported in the PowerKVM environment may be added over time. These currently include L2/L3 cache line self-healing features.

Other functions such as Alternate Processor Recovery depend heavily on the virtualization features of the POWER Hypervisor and are unlikely to become part of the PowerKVM environment.

These differences are highlighted in Figure 13: Comparing Power System S812L and S822L to S812LC and S822LC.

It should also be noted that different hypervisors and operating systems themselves may have differing capabilities concerning avoiding code faults, security, patching, boot time, update strategies and so forth. These can be important but are not discussed in this whitepaper.

**POWER8 processor-based IBM Power System S812LC and S822LC**

**Introduction**

These systems make use of the OpenPower Abstraction Layer (OPAL) software to provide essential abstraction of the hardware either directly to an operating system or through the PowerKVM hypervisor.

These systems differ from other Power Systems in that they are intended to be managed by the Intelligent Platform Management Interface (IPMI) providing an alternative approach to system management.
Because of this, they do not make use of the IBM flexible service processor for handling errors. Instead they make use of a Baseboard Management Controller (BMC).

Figure 13 gives an overview of how these systems compare to IBM Power System S812L and S822L configured to run with the PowerKVM hypervisor.

![Figure 13: Comparing Power System S812L and S822L to S812LC and S822LC](legend: ● supported, ▬ not supported)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processor/Memory</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base Processor Features Including:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ L1 Cache Set Delete</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>▪ L2/L3 Cache ECC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ Processor Fabric Bus ECC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ Memory Bus CRC with Retry/Sparing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor Instruction Retry</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Advanced Processor Self-Healing and fault handling:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ L2/L3 Cache Line Delete</td>
<td>–</td>
<td>–</td>
<td>●</td>
</tr>
<tr>
<td>▪ L3 Cache Column eRepair</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ Alternate Processor Recovery</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enterprise Memory with Spare DRAMS</td>
<td>–</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td><strong>Power/Cooling</strong></td>
<td>S822LC – Configuration Dependent</td>
<td>S812LC -Standard</td>
<td></td>
</tr>
<tr>
<td>Redundant Power Supplies</td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>OCC error handling w/ power safe mode</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Redundant Fans/Rotors</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Hot Swap Fans</td>
<td>S822LC</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Hot Swap DASD / Media</td>
<td>● (DASD only)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td><strong>I/O</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dual disk controllers (split backplane)</td>
<td>–</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Hot Swap PCI Adapter</td>
<td>–</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Concurrent Op Panel Repair</td>
<td>N/A</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Cable Management</td>
<td>Arm with slide rail S822LC (8335 only)</td>
<td>Arm</td>
<td>Arm</td>
</tr>
<tr>
<td><strong>Service Infrastructure</strong></td>
<td>BMC</td>
<td>FSP</td>
<td>FSP</td>
</tr>
<tr>
<td>Service Processor Type</td>
<td>Limited Lightpath</td>
<td>Partial Lightpath</td>
<td>Partial Lightpath</td>
</tr>
<tr>
<td>Service Indicators</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Error Handling

In implementing the described error reporting and management interface, these servers make use of a Baseboard Management Controller (BMC) based service processor.

User level error reporting is done through the IPMI interface to the BMC including thermal and power sensor reports.

The BMC service processor has different capabilities than the IBM designed Flexible Service Processor (FSP) used in other Power Systems.

For example, the FSP can monitor the processor and memory sub-system for recoverable errors that occur during run-time. The same service processor is also capable of doing analysis of faults at IPL-time or after a system checkstop.

The BMC-based service processor structure does not allow for the service processor to fully perform those activities.

Therefore, to maintain essential Power System First Failure Data Capture capabilities for the processor and memory, much of the run-time diagnostic code that runs on the FSP in other Power Systems has been written as an application that can run as code on a host processor.

This code can monitor recoverable errors and make service related callouts when an error threshold is reached. Compared to the FSP based system, however, this host-based code will generally not make requests for resource de-configuration on recoverable errors.

Some unrecoverable errors that might be handled by the hypervisor, such as uncorrectable errors in user data can still be signaled to the hypervisor.

Unrecoverable errors not handled by the hardware or hypervisor will result in a platform checkstop. On a platform checkstop, the service processor will gather fault register information. On a subsequent system reboot, that fault register information is analyzed by code similar to the run-time diagnostics code, running on the processor module the system booted from.

If uncorrectable errors are found in hardware during the boot process, it might be possible for a system design to run through a continuous process of booting, encountering an unrecoverable error, then unsuccessfully attempt boot again.
To prevent such repeated crashes, or systems failing to IPL due to uncorrectable resources, resources with uncorrectable errors detected during the diagnostic process at boot-time can be deconfigured (guarded) to allow the system to IPL with remaining resources, where possible.

IPL can only occur from the first processor in a multi-socket system, however.

In addition, the BMC-based service processor is also essential to boot. The BMC design allows for a normal and a “golden” boot image to be maintained. If difficulties occur during normal IPL, the system can fall back to the “golden” boot image to recover from certain code corruptions as well as to handle issues with guarded resources.
**POWER8 processor-based IBM POWER E850 and E850c Systems**

*Introduction*

The IBM Power System E850 and E850c have 4 processor sockets, with a minimum of two processor sockets each capable of supporting a POWER8 dual chip modules (DCM). These DCMs are the same type as used in 1s and 2s systems. However, because of the increased capacity this model was designed with additional enterprise-class availability characteristics.

The systems are capable of taking advantage of such enterprise features as Capacity on Demand and RAS features which leverage this capability are included where spare capacity is available. In addition, systems have the option of mirroring the memory used by the hypervisor.

*System Structure*

Figure 14 below gives a reasonable though not exact or complete representation of the components of a fully populated E850/E850c server.

The figure illustrates the unique redundancy characteristics of the system. Power supplies are configured with an N+1 redundant capability.

Fan rotors within the system also maintain at least an N+1 redundancy, meaning the failure of any one fan rotor, absent any other fault, does not cause a system outage.

Fans used to cool the CEC planar and components associated with it are also concurrently maintainable.

There are additional fans in the lower portion of the system used to cool components associated with the RAID controller/storage section. These fans cannot be repaired concurrently. Therefore, the system is supplied with sufficient fan rotors to ensure N+1 redundancy, and in addition two additional fan rotors are provided. The additional fan rotors are considered as integrated spares that do not require replacing.

This configuration allows the system to run with two of these fan rotors failed, without requiring a repair, and the system still maintains n+1 fan rotor redundancy.

Given the amount of sparing built in, there is a very low expectation of needing to take a system down to repair any of these fans. Should such a repair be necessary, however, the entire set of 8 fans would be replaced in a single repair action.

As in other systems, the power supplies take AC power inputs and convert to a single 12 volt dc level. Voltage regulator modules (or VRMs) are then used to convert to the various voltage levels that different components need (1.2 volts, 3.3 volts, etc.)

As the general rule the design for such a VRM can be described as consisting of some common non-redundant logic plus one or more converters, channels, or phases. The IBM Power System E850/E850c provide one more such phase than is needed for any voltage output that is distributed to critical elements such as a processor or memory DIMM. This is sometimes referred to as having a redundant phase.

The module(s) supplying the processor core and cache elements are replaceable separate from the CEC planar. If one of the phases within the module fails, a call for repair will be made, allowing for redundancy of these components. For other cases, such as the module cards supplying memory DIMMS, the extra phase is used as an integrated spare. On the first phase failure in such a VRM, the system continues operation without requiring a repair, thus avoiding outage time associated with the repair.

As previously noted, when a component such as a processor module or a custom memory DIMM is supplied voltage levels with redundant phases, redundancy typically ends at the processor or CDIMM. On some devices, such as for a memory DIMM the can be some sort of voltage converter to further divide a voltage level for purposes such as providing a reference voltage or signal termination. Other low-voltage regulation may be used for initial power-on initialization of main regulators. Such uses are typically not as demanding on the device as the output phases previously discussed and these devices are not included in any discussion or illustration of voltage regulation.
Comparing the IBM Power System E850/E850C to IBM Power System E870/E870c and IBM Power System E880

There are still some significant differences between the IBM Power System E850/E850C and the E870/E870C/E880/E880C systems that can impact availability.

The latter systems have a redundant clock and service processor infrastructure which the E850/E850C lacks.
As will be discussed in the next sub-section, the IBM Power E870/E870c and E880/E880c systems can expand beyond 4 sockets by adding additional 4 socket system node drawers. When one of these systems is configured with more than one such drawer, there is the added RAS advantage that a portion of the system can IPL after a fault has been isolated to a drawer even for severe faults that impact an entire drawer.

This would not be true of an E850/E850C system.

**POWER8 processor-based IBM Power System E870/E870c and IBM Power System E880**

**Introduction**

The basic building-block of an IBM Power E870/E870c or IBM Power E880/E880c systems is a 4 process-socket system node drawer which fits into a 19” rack. Each node drawer also includes 32 DIMM sockets and 8 PCIe I/O adapter slots. Announced systems support 1 or 2 such system nodes.

In addition to the system nodes, each Power E870/E870c or E880/E880c system includes a separate system control unit also installed into a 19” rack and connected by cables to each system node. The system control node contains redundant global service processors, clock cards, system VPD and other such system-wide resources.

Figure 15 gives a rough logical abstraction of such a system with an emphasis on illustrating the system control structure and redundancy within the design.

**Figure 15: Logical View of Power System E870/E80 System Structure**

The figure illustrates a high level of redundancy of the system control structure (service processor, clocking, and power control) as well as power and cooling.

Most redundant components reside on redundant cards or other field replaceable units that are powered and addressed independently of their redundant pairs. This includes the redundant global and local...
service processor functions and clocks, the power interface that provides power to from the system nodes to the System control unit, as well as system fans.

There is only one system VPD card and one Op Panel, but the system VPD is stored redundantly on the system VPD card, and there are three thermal sensors on the Op panel ensuring no loss of capability due to the malfunction of any single sensor.

At the CEC-level there are voltage regulator modules (VRMs) packaged in Voltage Regulator Module (VRM) cards that are individually pluggable. Each such VRM can be described as having some common elements plus at least three components (known as converters, channels or phases) working in tandem.

The voltage regulator cards for the E870/E870c E880/E880c have two additional runtime output phases beyond what is needed for operation. So, by definition, the design allows for phase redundancy and in addition contains an additional spare phase. Using the spare phase prevents the need to replace a VRM card due to a failure of a single phase.

New in POWER8, the battery used to maintain calendar time when a Flexible Service Processor (FSP) has no power is now maintained on a separate component. This allows the battery to be replaced concurrently separate from the service processor.

In the design, fans and power supplies are specified to operate in N+1 redundancy mode meaning that the failure of a single power supply, or single fan by itself will not cause an outage. In practice, the actual redundancy provided may be greater than what is specified. This may depend on the system and configuration. For example, there may be circumstances where the system can run with only two of four power supplies active, but depending on which power supplies are faulty, may not be able to survive the loss of one AC input in a dual AC supply data-center configuration.

Finally, in multi-node systems, firmware can isolate failing node(s) in such a way that would allow portions of the system to IPL even when entire node(s) need to be deconfigured. For example, in a two node system, if a fabric bus has an uncorrectable fault between the two nodes, the system can fall back to IPL as a one node system. Or on a system with nodes 1-4, if a severe processor chip fault occurs in node three, the system can fallback to IPL as a two-node system.
Section 3: General RAS Philosophy and Architecture

Philosophy

In the previous section three different classes of servers were described with different levels of RAS capabilities. While each server had unique attributes, the section highlighted several common attributes. This section will outline some of the design philosophies and characteristics that influence the Power Systems designs. The following section will detail more specifically how the RAS design is carried out in each sub-system of the server.

Integrate System Design

These IBM Power Systems all use a processor architected, designed and manufactured by IBM. IBM systems contain other hardware content also designed and manufactured by IBM including memory buffer components, service processors and so forth.

Additional components not designed or manufactured by IBM are chosen and specified by IBM to meet system requirements. These are procured for use by IBM using a rigorous procurement process intended to deliver reliability and design quality expectations.

The systems that IBM design are manufactured by IBM to IBM’s quality standards.

The systems incorporate software layers (firmware) for error detection/fault isolation and support as well as virtualization in a multi-partitioned environment.

These include IBM designed and developed service firmware. IBM’s PowerVM hypervisor is also IBM designed and supported.

In addition, IBM offers two operating systems developed by IBM: AIX and IBM i. Both operating systems come from a code base with a rich history of design for reliable operation.

IBM also provides middle-ware and application software that are widely used such as IBM WebSphere® and DB2® pureScale™ as well as software used for multi-system clustering, such as various IBM PowerHA™ SystemMirror™ offerings.

These components are designed with application availability in mind, including the software layers, which are also capable of taking advantage of hardware features such as storage keys that enhance software reliability.
The figure above indicates how IBM design and influence may flow through the different layers of a representative enterprise system as compared to other designs that might not have the same level of control. Where IBM provides the primary design, manufacture and support for these elements, IBM can be responsible for integrating all of the components into a coherently performing system and verifying the stack during design verification testing.

In the end-user environment, IBM likewise becomes responsible for resolving problems that may occur relative to design, performance, failing components and so forth, regardless of which elements are involved.

**Incorporate Experience**

Being responsible for much of the system, IBM puts in place a rigorous structure to identify issues that may occur in deployed systems and identify solutions for any pervasive issue. Having support for the design and manufacture of many of these components, IBM is best positioned to fix the root cause of problems, whether changes in design, manufacturing, service strategy, firmware or other code is needed.

The detailed knowledge of previous system performance has a major influence on future systems design. This knowledge lets IBM invest in improving the discovered limitations of previous generations. Beyond that, it also shows the value of existing RAS features. This knowledge justifies investing in what is important and allows for adjustment to the design when certain techniques are shown to be no longer of much importance in later technologies or where other mechanisms can be used to achieve the same ends with less hardware overhead.

**Keep Error Detection as a First Priority**

It is not possible to detect or isolate every possible fault or combination of faults that a server might see. It is important to invest in error detection, however especially for workloads where getting the right answer is paramount.

Within the POWER9 processor and memory sub-system, this necessarily means systematically investing in error detection. This includes the obvious such as checking for data in memory and caches and validating that data transferred across busses is correct. It goes well beyond this to include techniques for checking state machine transitions, residue checking for certain operations and protocol checking to not
only make sure that the bits transmitted are correct, but also that the data went when and where it was expected and so on.

When it is detected that a fault has occurred, the primary intent of the RAS design is to prevent reliance on this data. Most of the rest of the RAS characteristics discussed in this section describe ways in which disruption due to bad data can be eliminated or minimized. However, there will be cases where avoiding reliance on bad data or calculations means terminating operation.

It should be pointed out error detection may seem like a well understood and expected goal. However, it is not always the goal of every possible sub-system design. Hypothetically, for instance, graphics processing units (GPU) whose primary purpose is rendering graphics in non-critical applications may have options for turning off certain error checking (such as ECC in memory) to allow for better performance. The expectation in such case is that there are applications where a single dropped pixel on a screen is of no real importance, and a solid fault is only an issue if it is noticed.

In general, I/O adapters may also have less hardware error detection capability where they can rely on a software protocol to detect and recover from faults.

**Leverage Technology and Design for Soft Error Management**

In a very real sense errors detected can take several forms. The most obvious is a functional fault in the hardware – a silicon defect, or a worn component that over time has failed.

Another kind of failure is what is broadly classified as a soft error. Soft errors are faults that occur in a system and are either occasional events inherent in the design or temporary faults that are due to an external cause.

Data cells in caches and memory, for example, may have a bit-value temporarily upset by an external event such as caused by a cosmic ray generated particle. Logic in processors cores can also be subject to soft errors where a latch may also flip due to a particle strike or similar event. Busses transmitting data may experience soft errors due to clock drift or electronic noise.

The susceptibility to soft errors in a processor or memory subsystem is very much dependent on the design and technology used in these devices. This should be the first line of defense. Choosing latches that are less susceptible to upsets due to cosmic ray events was discussed extensively in previous whitepapers for example.

Methods for interleaving data so that two adjacent bits in array flipping won’t cause undetected multi-bit flips in a data word is another design technique that is also important.

Ultimately when data is critical, detecting soft error events that occur needs to be done immediately, in-line to avoid relying on bad data because periodic diagnostics is insufficient to catch an intermittent problem before damage is done.

The simplest approach to detecting many soft error events may simply be having parity protection on data which can detect a single bit flip. When such simple single bit error detection is deployed, however, the impact of a single bit upset is bad data. Discovering bad data without being able to correct it will result in termination of an application, or even a system so long as data correctness is important.

To prevent such a soft error from having a system impact it is necessary not simply to detect a bit flip, but also to correct. This requires more hardware than simple parity. It has become common now to deploy a bit correcting error correction code (ECC) in caches that can contain modified data. Because such flips can occur in more than just caches, however, such ECC codes are widely deployed in POWER9 processors in critical areas on busses, caches and so forth.

Protecting a processor from more than just data errors requires more than just ECC checking and correction. CRC checking with a retry capability is used on a number of busses, for example.

Significantly POWER processors since POWER6® have been designed with sufficient error detection to not only notice key typical software upsets impacting calculations, but to notice quickly enough to allow processor operations to be retried. Where retry is successful, as would be expected for temporary events, system operation continues without application outages.
Deploy Strategic Spare Capacity to Self-Heal Hardware

Techniques that protect against soft errors are of limited protection against solid faults due to a real hardware failure. A single bit error in a cache, for example can be continually corrected by most ECC codes that allow double-bit detection and single bit correction.

However, if a solid fault is continually being corrected, the second fault that occurs will typically cause data that is not correctable. This would result in the need to terminate, at least, whatever is using the data.

In many system designs, when a solid fault occurs in something like a processor cache, the management software on the system (the hypervisor or OS) may be signaled to migrate the failing hardware off the system.

This is called predictive deallocation. Successful predictive deallocation may allow for the system to continue to operate without an outage. To restore full capacity to the system, however, the failed component still needs to be replaced, resulting in a service action.

Within POWER, the general philosophy is to go beyond simple predictive deallocation by incorporating strategic sparing or micro-level deallocation of components so that when a hard failure that impacts only a portion of the sub-system occurs, full error detection capabilities can be restored without the need to replace the part that failed.

Examples include a spare data lane on a bus, a spare bit-line in a cache, having caches split up into multiple small sections that can be deallocated, or a spare DRAM module on a DIMM.

Focus on OS Independence

Because Power Systems have long been designed to support multiple operating systems, the hardware RAS design is intended to allow the hardware to take care of the hardware largely independent of any operating system involvement in the error detection or fault isolation (excluding I/O adapters and devices for the moment.)

To a significant degree this error handling is contained within the processor hardware itself. However, service diagnostics firmware, depending on the error, may aid in the recovery. When fully virtualized, specific OS involvement in such tasks as migrating off a predictively failed component can also be performed transparent to the OS.

The PowerVM hypervisor is capable of creating logical partitions with virtualized processor and memory resources. When these resources are virtualized by the hypervisor, the hypervisor has the capability of deallocating fractional resources from each partition when necessary to remove a component such a processor core or logical memory block (LMB).

When an I/O device is directly under the control of the OS, the error handling of the device is the device driver responsibility. However, I/O can be virtualized through the VIOS offering meaning that I/O redundancy can be achieved independent of the OS.

Build System Level RAS Rather Than Just Processor and Memory RAS

IBM builds Power systems with the understanding that every item that can fail in a system is a potential source of outage.

While building a strong base of availability for the computational elements such as the processors and memory is important, it is hardly sufficient to ensure application availability.

The failure of a fan, a power supply, a voltage regulator, or I/O adapter might be more likely than the failure of a processor module designed and manufactured for reliability.

Scale-out servers will maintain redundancy in the power and cooling subsystems to avoid system outages due to common failures in those areas. Concurrent repair of these components is also provided.

For the Enterprise system, a higher investment in redundancy is made. The E980 system, for example is designed from the start with the expectation that the system must be generally shielded from the failure of these other components incorporating redundancy within the service infrastructure (such as redundant service processors, redundant processor boot images, and so forth.) An emphasis on the reliability of components themselves are highly reliable and meant to last.
This level of RAS investment extends beyond what is expected and often what is seen in other server designs. For example, at the system level such selective sparing may include such elements as a spare voltage phase within a voltage regulator module.

**Error Reporting and Handling**

*First Failure Data Capture Architecture*

POWER processor-based systems are designed to handle multiple software environments including a variety of operating systems. This motivates a design where the reliability and response to faults is not relegated to an operating system.

Further, the error detection and fault isolation capabilities are intended to enable retry and other mechanisms to avoid outages due to soft errors and to allow for use of self-healing features. This requires a very detailed approach to error detection.

This approach is beneficial to systems as they are deployed by end-users, but also has benefits in the design, simulation and manufacturing test of systems as well.

Putting this level of RAS into the hardware cannot be an after-thought. It must be integral to the design from the beginning, as part of an overall system architecture for managing errors. Therefore, during the architecture and design of a processor, IBM places a considerable emphasis on developing structures within it specifically for error detection and fault isolation.

Each subsystem in the processor hardware has registers devoted to collecting and reporting fault information as they occur. The design for error checking is rigorous and detailed. The value of data is checked generally wherever it is stored. This is true, of course for data used in computations, but also nearly any other data structure, including arrays used only to store performance and debug data.

Error checkers are derived for logic structures using various techniques such as checking the validity of state-machine transitions, defining and checking protocols for generated commands, doing residue checking for certain computational instructions and by other means in an effort to detect faults before the resulting impact propagates beyond the detecting sub-system.

The exact number of checkers and type of mechanisms isn’t as important as is the point that the processor is designed for very detailed error checking; much more than is required simply to report during run-time that a fault has occurred.

All these errors feed a data reporting structure within the processor. There are registers that collect the error information. When an error occurs, that event typically results in the generation of an interrupt.

The error detection and fault isolation capabilities maximize the ability to categorize errors by severity and handle faults with the minimum impact possible. Such a structure for error handling can be abstractly illustrated by the figure below and is discussed throughout the rest of this section.
Processor Runtime Diagnostics

In previous system designs, the dedicated server processor ran code, referred to here as Processor Runtime Diagnostics (PRD) which would access this information and direct the error management.

Ideally this code primarily handles recoverable errors including orchestrating the implementation of certain “self-healing” features such as use of spare DRAM modules in memory, purging and deleting cache lines, using spare processor fabric bus lanes, and so forth.

Code within a hypervisor does have control over certain system virtualized functions, particularly as it relates to I/O including the PCIe controller and certain shared processor accelerators. Generally, errors in these areas are signaled to the hypervisor.

In addition, there is still a reporting mechanism for what amounts to the more traditional machine-check or checkstop handling.

In a POWER7 generation system the PRD was said to run and manage most errors whether the fault occurred at run-time, or at system IPL time, or after a system-checkstop – which is the descriptive term for entire system termination by the hardware due to a detected error.

In POWER8 the processor module included a Self-Boot-Engine (SBE) which loaded code on the processors intended to bring the system up to the point where the hypervisor could be initiated. Certain faults in early steps of that IPL process were managed by this code and PRD would run as host code as part of the boot process.

In POWER9 process-based systems, during normal operation the PRD code is run in a special service partition in the system on the POWER9 processors using the hypervisor to manage the partition. This has the advantage in systems with a single service processor of allowing the PRD code to run during normal system operation even if the service processor is faulty.

Service Diagnostics in the POWER9 Design

In a POWER7 generation server, PRD and other service code was all run within the dedicated service processor used to manage these systems. The dedicated service processor was in charge of the IPL...
process used to initialize the hardware and bring the servers up to the state where the hypervisor could begin to run. The dedicated service processor was also in charge, as previously described, to run the PRD code during normal operation.

In the rare event that a system outage resulted from a problem, the service processor had access not only to the basic error information stating what kind of fault occurred, but also access to considerable information about the state of the system hardware – the arrays and data structures that represent the state of each processing unit in the system, and also additional debug and trace arrays that could be used to further understand the root cause of faults.

Even if a severe fault caused system termination, this access provided the means for the service processor to determine the root cause of the problem, deallocate the failed components, and allow the system to restart with failed components removed from the configuration.

POWER8 gained a System-Boot-Engine which allowed processors to run code and boot using the POWER8 processors themselves to speed up the process and provide for parallelization across multiple nodes in the high-end system. During the initial stages of the IPL process, the boot engine code itself handled certain errors and the PRD code ran as an application as later stages if necessary.

In POWER9 the design has changed further so that during normal operation the PRD code itself runs in a special hypervisor-partition under the management of the hypervisor. This has the advantage of continuing to allow the PRD code to run even if the service processor is non-functional (important in non-redundant environments.)

Should a fault the code running fail the hypervisor can restart the partition (reloading and restarting the PRD.)

The system service processors are also still monitored at run-time by the hypervisor code and can report errors if the service processors are not communicating.

**Processor Module Design and Test**

While Power Systems are equipped to deal with soft errors as well as random occasional hardware failures, manufacturing weaknesses and defects should be discovered and dealt with before systems are shipped. So before discussing error handling in deployed systems, how manufacturing defects are avoided in the first place, and how error detection and fault isolation is validated will be discussed.

Again, IBM places a considerable emphasis on developing structures within the processor design specifically for error detection and fault isolation.

The design anticipates that not only should errors be checked, but also that the detection and reporting methods associated with each error type also need to be verified. When there is an error that can be checked, and some sort of recovery or repair action initiated, there generally should be a hardware method to “inject” an error that will test the functioning of the hardware detection and firmware capabilities. Such error injecting can include different patterns of errors (solid faults, single events, and intermittent but repeatable faults.) Where direct injection is not provided, there should be a way to at least simulate the report that an error has been detected and test response to such error reports.

Under IBM control, assembled systems are also tested and a certain amount of system “burn-in” may also be performed, doing accelerated testing of the whole system to weed-out weak parts that otherwise might fail during early system life, and using the error reporting structure to identify and eliminate the faults.

In that mode the criteria may be more severe as to what constitutes a failure. A single fault, even if it’s recoverable, might be enough to fail a part during this system manufacturing process.

**Partitioning with PowerVM**

Power Systems with PowerVM all run with a single copy of the PowerVM hypervisor. The PowerVM hypervisor provides logical partitioning allowing multiple instances of an operating system to run in a server.

It is beyond the scope of this whitepaper to outline all the ways in which PowerVM provides logical isolation of the hardware components.

However, the design is not meant as a physical partitioning approach. There are not multiple independent PowerVM hypervisors running in a system. Faults that can impact PowerVM in a critical area can and will
result in outages either directly by PowerVM termination or indirectly by the hardware determining that for PowerVM integrity the system will need to checkstop.

The reason underlying this design is the belief that most effective means of physical partitioning really involves having separate physical systems. This belief derives from the concern that any time even redundant elements come together in a system there is some common access point and the possibility of a "common mode" fault that impacts the entire system.
Section 4: Subsystems RAS Details

Processor RAS Details

The previous discussion in this section gives a general description of how a POWER processor is designed for error detection and fault isolation and how this design is used during processor manufacture and test. It also outlines how the POWER processor technology is hardened against soft errors.

It is worth noting that the functions integrated within a processor has changed much over time. One point of comparison could be a POWER6 processor compared to the current POWER9.

Figure 18: POWER6 Processor Design Compared to POWER9

The illustration above shows a rough view of the POWER9 scale-up processor design leveraging SMT8 cores (a maximum of 12 cores shown.)

The POWER9 design is certainly more capable. There is a maximum of 12 SMT8 cores compared to 2 SMT2 cores. The core designs architecturally have advanced in function as well. The number of memory controllers has doubled, and the memory controller design is also different.

The addition of system-wide functions such as the NX accelerators and the CAPI and NVLINK interfaces provide functions just not present in the hardware of the POWER6 system.

The POWER9 design is also much more integrated. The L3 cache is internal and the I/O processor host bridge is integrated into the processor. The thermal management is now conducted internally using the on-chip controller.

There are reliability advantages to the integration. It should be noted that when a failure does occur it more likely to be a processor module at fault compared to previous generations with less integration.

Hierarchy of Error Avoidance and Handling

In general, there is a hierarchy of techniques used in POWER™ processors to avoid or mitigate the impact of hardware errors. At the lowest level in the hierarchy is the design for error detection and fault isolation, the technology employed, specifically as relates to reducing the instances of soft error not only through error correction, but in the selection of devices within the processor IC.

Because of the extensive amount of functionality beyond just processor cores and caches, listing all the RAS capabilities of the various system elements would require considerable detail. In somewhat broader strokes, the tables below discuss the major capabilities in each area.
### Figure 19 : Error Handling Methods Highlights

| **Cache Error Handling** | The L2 and L3 caches in the processor use an ECC code that allows for single bit correction. 
During operation, when a persistent correctable error occurs in these caches, the system has the capability of purging the data in the cache (writing to another level of the hierarchy) and deleting it (meaning subsequent cache operations won’t use the cache line during operation.) This is an example of "self-healing" that avoids taking a planned outage for a correctable error. 
The L1 cache (usually thought of as part of the processor Core element) checks for single bit errors. But instead of using an error correcting code, intermittent L1 cache errors can be corrected using data from elsewhere in the cache hierarchy. 
A portion of an L1 cache can be disabled (set delete) to avoid outages due to persistent hard errors. 
If too many errors are observed across multiple sets the core using the L1 can be predictively deallocated. 
Where the IBM memory buffer used the L4 cache also has an ECC code that allows for single bit error correction. 
Separate from the system caches and the description above are cache directories which provide indexing to the caches. These also have single bit error correction. Uncorrectable directory errors will typically result in system checkstops as discussed below. |
| **Other ECC Checking** | Beyond these elements, single bit correcting ECC is used in multiple areas of the processor as the standard means of protecting data against single bit upsets (beyond the reliability design features previously mentioned.) This includes a number of the internal busses where data is passed between units. |
| **Special Uncorrectable Error Handling** | Where there is ECC in the path for data and an uncorrectable error is encountered, the desire to prevent reliance on bad data means that whatever is using the data will need to be terminated. 
In simpler system designs, that would mean termination of something, at least the owner of the data, as soon as the uncorrectable error is encountered. 
POWER has long had the concept of marking the data with a special ECC code and watching for when and if the data is actually going to be “consumed” by the owner (if the data is ever used.) At that point whatever the data owner is can be terminated. 
This can be a single application if a processor running under AIX is consuming user data. For kernel data, the OS kernel may be terminated. If PowerVM attempts to use the data in a critical error, then PowerVM will terminate. 
One additional advantage of the special error correction code is that the hardware is able to distinguish between a fresh ECC error when data is transferred and one that has been passed along. This allows the correct component, the one originating the fault, to be called out as the component to be replaced. |
<p>| <strong>Bus CRC</strong> | As previously mentioned, ECC is used internally in various data-paths as data is transmitted between units. |</p>
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Externally to the processor, however, high speed data busses can be susceptible to the occasional multiple bit errors due to the nature of the bus design.</td>
<td>A cyclic redundancy check code can be used to determine if there are errors within an entire packet of data. If the fault is due to natural changing of bus characteristics over time (&quot;drift&quot;) the bus can re-train, retry the operation and continue.</td>
</tr>
<tr>
<td>CRC checking is done for the memory bus. New in POWER9: CRC checking is now done for the processor fabric bus interfaces sending data between processors.</td>
<td></td>
</tr>
<tr>
<td>Lane Repair</td>
<td>By itself CRC checking has the advantage of handling multiple bit errors. For persistent problems it is in certain ways inferior to bit correcting ECC if a single persistent error cannot be corrected by retry. The memory bus between processors and the memory uses CRC with retry. The design also includes a spare data lane so that if a persistent single data error exists the faulty bit can be &quot;self-healed.&quot; The POWER9 busses between processors also have a spare data lane that can be substituted for a failing one to &quot;self-heal&quot; the single bit errors.</td>
</tr>
<tr>
<td>Split Data Bus (1/2 bandwidth mode)</td>
<td>In busses going between processor nodes (between CEC drawers in a E980 system) if there is a persistent error that is confined to the data on a single cable (and the bus is split between cables) POWER9 can reduce the band-width of the bus and send data across just the remaining power cable. This allows for correction of a more systematic fault across the bus. In addition, the bus between processors within a node or single CEC drawer system is also also capable of a split-bus or ½ bandwidth mode. Full support of SMP bus features may depend on firmware level.</td>
</tr>
<tr>
<td>Processor Instruction Retry</td>
<td>Within the logic and storage elements commonly known as the &quot;core&quot; there can be faults other than errors in the L1 mentioned above. Some of these faults may also be transient in nature. The error detection logic within the core elements is designed extensively enough to determine the root cause of a number of such errors and catch the fault before an instruction using the facility is completed. In such cases processor instruction retry capabilities within the processor core may simply retry the failed operation and continue. Since this feature was introduced in the POWER6, processor field data has shown many instances where this feature alone had prevented serious outages associated with intermittent faults. In POWER9 faults that are solid in nature and where retry does not solve the problem will be handled as described further down in this table.</td>
</tr>
<tr>
<td>Predictive Deallocation</td>
<td>Because of the amount of self-healing incorporated in these systems as well as the extensive error recovery it is rare that an entire processor core needs to be predictively deallocated due to a persistent recoverable error. If such cases do occur, PowerVM can invoke a process for deallocating the failing processor dynamically at run-time (save in the rare case that the OS application doesn’t allow for it.)</td>
</tr>
<tr>
<td>Core Checkstops</td>
<td>On scale-up systems where the use of many partitions may be common, if a fault cannot be contained by any of the previous features defined in</td>
</tr>
</tbody>
</table>
the hierarchy, it may be still beneficial to contain an outage to just the partitions running the threads when the uncorrectable fault occurred.

The Core Checkstop feature may do this in these systems supporting the feature for faults that can be contained that way. This allows the outage associated with the fault to be contained to just the partition(s) using the core at the time of the failure if the core is only used at the time by partition(s) other than in hypervisor mode.

It should be noted that in such an environment system performance may be impacted for the remaining cores after the core contained checkstop.

Note: A core checkstop signaled for a fault occurred on a core running a hypervisor instruction will typically result in hypervisor termination as described below.

<table>
<thead>
<tr>
<th>PowerVM Handled errors</th>
<th>There are other faults that are handled by the hypervisor that relate to errors in architected features – for example handling a SLB multi-hit error. Since the required error handling is documented for any hypervisor, details can be found in the POWER9 Processor User’s Manual maintained by the OpenPOWER™ Foundation. <a href="https://openpowerfoundation.org/?resource_lib=power9-processor-users-manual">https://openpowerfoundation.org/?resource_lib=power9-processor-users-manual</a>.</th>
</tr>
</thead>
</table>
| PCIe Hub Behavior and Enhanced Error Handling | Each processor has three elements called PCIE hubs which generate the various PCIe Gen4 busses used in the system. The hub is capable of “freezing” operations when certain faults occur, and in certain cases can retry and recover from the fault condition.

This hub freeze behavior prevents faulty data from being written out through the I/O hub system and prevents reliance on faulty data within the processor complex when certain errors are detected.

Along with this hub freeze behavior is what has long been termed as Enhanced Error Handling for I/O.

This capability signals device drivers when various PCIe bus related faults occur. Device drivers may also attempt to restart the adapter after such faults (EEH recovery.)

A clock error in the PCIe clocking can be signaled and recovered using EEH in any system that incorporates redundant PCIe clocks with dynamic failover enabled. |
| Hypervisor Termination and System Checkstops | If a fault is contained to a core, but the core is running PowerVM critical code, PowerVM will terminate to maintain the integrity of the computation of the partitions running under it.

Each fault checked in the system is reviewed during design. If it is known in advance that a particular fault can impact the hypervisor or if there is a fault in system facility that cannot be localized, the hardware will generate a platform checkstop to cause system termination.

This by design allows for the most efficient recovery from such errors and invokes the full error determination capabilities of the service processor.

In scale-out systems where core-contained checkstop is not applicable, hypervisor termination or system checkstops would be expected for events managed as core-contained checkstops in scale-up systems. |
| Memory Channel Checkstops and memory mirroring | The memory controller communicating between the processor and the memory buffer has its own set of methods for containing errors or retrying operations. |
Some severe faults require that memory under a portion of the controller become inaccessible to prevent reliance on incorrect data. There are cases where the fault can be limited to just one memory channel. In these cases, the memory controller asserts what is known as a channel checkstop. In systems without hypervisor memory mirroring, a channel checkstop will usually result in a system outage. However, with hypervisor memory mirroring the hypervisor may continue to operate in the face of a memory channel checkstop.

**Processor Safe-mode**

Failures of certain processor wide facilities such as power and thermal management code running on the on-chip-sequencer and the self-boot-engine used during run-time for out of band service processor functions can occur. To protect system function, such faults can result in the system running in a “safe-mode” which allows processing to continue with reduced performance in the face of errors where the ability to access power and thermal performance or error data is not available.

**Persistent Guarding of Failed Elements**

Should a fault in a processor core become serious enough that the component needs to be repaired (persistent correctable error with all self-healing capabilities exhausted or an unrecoverable error), the system will remember that a repair has been called for and not re-use the processor on subsequent system reboots, until repair. This functionality is extended to other processor elements and to entire processor modules whenever the risk is that relying on that element in the future means risking another outage.

In a multi-node system, deconfiguration of a processor node for fabric bus consistency reasons results in deconfiguration of a node.

As systems design continues to mature the RAS features are may be adjusted based on the needs of the newer designs as well as field experience; therefore, this list differs from the POWER8 design.

For example, in POWER7+, an L3 cache column repair mechanism was implemented to be used in addition to the ability to delete rows of a cache line.

This feature was carried forward into the POWER8 design, but the field experience in the POWER8 technology did not show the benefit based on how faults that might implement multiple rows surfaced. For POWER9 enterprise systems, given the size of the caches, the number of lines that were deleted was extended instead of continuing to carry column repair as a separate procedure.

Likewise, in POWER6 a feature known as alternate processor recovery was introduced. The feature had the purpose of handling certain faults in the POWER6 core. The faults handled were limited to certain faults where the fault was discovered before an instruction was committed and the state of the failing core could be extracted. The feature, in those cases, allowed the failing workload to be dynamically retried on an alternate processor core. In cases where no alternate processor core was available, some number of partitions would need to be terminated, but the user was allowed to prioritize which partitions would be terminated to keep the most important partition running.

The mixture of which parts could be handled by this kind of design, and the method of delivering the design itself, changed from generation to generation. In POWER8, field observations showed that essentially, and in part due to advances in other error handling mechanisms, the faults where alternate processor recovery was required and viable had become exceedingly rare. Therefore, the function is not being carried over in POWER9.
POWER 9 Memory RAS

Memory RAS Fundamentals

Model Comparative Memory RAS

Unlike the previous generation, the different PowerVM based POWER9 systems have different approaches to the memory subsystem.

These are summarized roughly in the figure below:

Figure 20: Memory Design Points

POWER9 Memory Options

1-2 Socket Scale-out Systems
- Leverages new direct attach memory approach with no buffer chip
- Allows for Chipkill correction using X4 DRAMS
  - Improved over P7 with ISDIMMs by correcting 2 instead of 1 random symbols in the absence of chipkill
- No Sparing

E950 4 Socket System
- With x4 DIMMS
  - Across a DIMM pair:
    - x4 Chipkill correction
    - Plus 1 spare DRAM module
    - And Row Repair*
  - Will only ship x4 DIMMS
- Riser Card
  - Separate FRU used for memory buffers

E980 Systems
- Supports x4 and x8 DIMMS
- Enterprise Level of error correction for x4 and x8 DIMMS
- Spare DRAM Modules on Custom DIMM even when x8 DIMMS are used
- Highest DIMM integration means fewer failures compared to the 950 ISDIMM design (details in the reliability estimates)

* For some DIMMs with FW released on 11/16/2018 or later

Memory Design Basics

To fully understand the reasons for the differences some background is required.

In many servers the basic building block of main system memory is Dynamic Random Access Memory integrated circuit modules (DRAMs).

In a very simplified sense a DRAM module can be thought of as an array of bits many rows deep but typically either 4 or 8 columns wide. These can be referred to as x4 or x8 DRAMs.

Processors read main memory in units called a memory cache line. Many non-POWER systems have a memory cache line that is 64 bytes. These cache lines are typically read in bursts of activity a memory “word” at a time where a memory word would be 64 bits plus additional check bits allowing for some level of error correction.
On a DIMM multiple DRAM modules are accessed to create a memory word. An “Industry Standard” DIMM is commonly used to supply processors fetching 64 byte cache lines.

A rank of an x4 “Industry Standard” DIMM contains enough DRAMs to provide 64 bits of data at a time with enough check bits to correct the case of a single DRAM module being bad after the bad DRAM has been detected and then also at least correct an additional faulty bit.

The ability to correct an entire DRAM being bad is what IBM has traditionally called Chipkill correction. Being able to correct this kind of fault is essential in protecting against a memory outage and should be considered as a minimum error correction for any modern server design.

Inherently DIMMs with x8 DRAMs use half the number of memory modules and there can be a reliability advantage just in that alone. From an error checking standpoint, only 4 bad adjacent bits in an ECC word need to be corrected to have Chipkill correction with a x4 DIMM. When x8 DIMMS are used, a single Chipkill event requires correcting 8 adjacent bits. A single Industry Standard DIMM in a typical design will not have enough check bits to handle a Chipkill event in a x8 DIMM.

Rather than directly accessing the DIMMS through the processor, some server designs take advantage of a separate memory buffer. A memory buffer can improve performance and it can also allow servers to take advantage of two Industry standard DIMMs by extending the ECC word in some sense across two DIMMS.
Hence across two DIMMs, with a x4 DRAM module, a single DRAM module being failed can be corrected and then another, or what might be called double-Chipkill though this is not a precise term. With x8 DRAMs at least across the two DIMMS a single Chipkill event can be corrected once the fault has been identified and the bad DRAM marked.

In system designs where only 64 byte ECC caches are filled there can be a performance penalty associated with running in 128 byte mode across two DIMMS since the cache lines are still 64 byte and the DIMMs are still designed to access 64 bytes in burst. Hence in newer designs, 128 byte ECC word access may only be switched to after the first Chipkill event.

**POWER Traditional Design**

IBM Power systems traditionally don't use a 64 byte-cache line. Instead, a 128 byte cache line is used. Traditionally an external IBM designed buffer module is used to allow the equivalent of two Industry Standard DIMMs to be accessed in the 128 byte cache line. In POWER8, instead of using Industry Standard DIMMS, the servers used a custom DIMM design where the IBM memory buffer was packaged on the DIMM with the equivalent of the DRAMS of four industry standard DIMMs. In addition, the custom DIMMS carried additional spare DRAMs not found in industry standard DIMMs.

With custom DIMM design, IBM could use x8 DRAMS where it was advantageous to do so, still mark out a failed memory module, and on top of that offer spare DRAM modules.

For x4 DIMMs, Chipkill correction is provided with even more spare capacity.

**1s and 2s Scale-out Design**

The POWER9 processor used in 1 and 2 socket scale-out servers are designed internally for Industry Standard DIMMs without an external buffer chip. The ECC checking is at the 64 bit level, so Chipkill protection is provided with x4 DIMMs; plus some additional sub-Chipkill level error checking after a Chipkill event. Because it is a 64 bit ECC word, however, there is no spare capability.
IBM does not believe that using DIMMs without Chipkill capability is appropriate in this space, so only x4 DIMMS are offered by IBM.

**E950 Design Point**

As previously illustrated, the E950 makes use of IBM’s memory buffer and also Industry standard DIMMs. With x4 DIMMs this allows for memory ECC with Chipkill correction, plus a spare DRAM module across a pair of DIMMS for each DIMM rank.

If x8 DIMMS were offered, there would be Chipkill correction but no spares. Believing that avoiding both the unplanned and planned outages in this space is important, x8 DIMMS are not offered by IBM.

**DRAM Row Repair**

To further extend the ability to avoid planned outages for the E950, firmware released on 11/16/2018 or later support a concept known as DRAM row repair for certain DIMMS.

DRAMs on the 32GB, 64GB and 128GB DIMMS are manufactured with at least an extra, spare “row” of data. If a Spare DRAM is used during operation due to a fault that is contained to a single row, on system reboot, row repair can substitute the spare row for the bad row and bring the DRAM back into use. This allows the recovered DRAM to be used again as a spare.

This can improve the self-healing capabilities of the DIMMs beyond simply the use of the spare DRAM module alone.

In future firmware releases, DRAM row repair might be extended to the 8GB and 16GB DIMMS as well.
E980 Design Point

DDR4 Custom DIMMS as described previously are used in the E980 design. This allows for both x4 and x8 DIMMS to be used and still offer more spare capability than the E950 that uses standard DIMMs.

Figure 24: POWER8 Memory Subsystem

Memory Controller
- Supports 128 Byte Cache Line
- Reply buffer to retry after soft internal faults
- Special Uncorrectable error handling for solid faults

Memory Bus
- CRC protection with recalibration and retry on error
- Spare Data lane can be dynamically substituted for failed one

Memory Buffer
- Can retry after internal soft errors
- L4 Cache DED/SEC ECC Code
  - Persistent correctable error handling

16 GB DIMM
- 4 Ports of Memory
  - 10 DRAMs x8 DRAMs attached to each port
  - 8 Needed For Data
  - 1 Needed For Error Correction Coding
  - 1 additional Spare
- 2 Ports are combined to form 128 data bits
  - 8 Reads fill a processor cache
- Second port can be used to fill a second cache line
  - (Much like having 2 DIMMs under one Memory buffer but housed in the same physical DIMM)

Memory RAS Beyond ECC

A truly robust memory design incorporates more RAS features than just the ECC protection of course. The ability to use hardware accelerated scrubbing to refresh memory that may have experienced soft errors is a given. The memory bus interface is also important. The direct bus attach memory used in the scale-out servers supports RAS features expected in that design including register clock driver (RCD) parity error detection and retry.

IBM's memory buffer supports bus CRC with spare data-lane and retry as previously described plus soft error handling features inside the buffer.

Hypervisor Memory Mirroring

As an additional capability to memory sub-systems RAS, the ability to mirror memory can provide an additional protection against memory related outages.

In general mirroring memory has a down-side. There is additional cost of the mirrored memory, mirroring memory can reduce memory capacity and may also have an impact on performance such as due to the need to write to two different memory locations whenever data is stored.

POWER9 Scale-up processors provide a means to mirror just the memory used in critical areas of the PowerVM hypervisor. This provides protection to the hypervisor so that it does not need to terminate due to faults on a DIMM that cause uncorrectable errors.
By selectively mirroring only the segments used by the hypervisor, this protection is provided without the need to mirror large amounts of memory.

It should be noted that the PowerVM design is a distributed one. PowerVM code can reside in small amounts in memory anywhere in the system where a processor has need to access PowerVM services. The selective mirroring approach is fine grained enough not to require the Hypervisor to sit in any particular memory DIMMs. This provides the function while not compromising the Hypervisor performance as might be the case if the code had to reside remotely from the processors it serves.

**Dynamic Deallocation/Memory Substitution**

Especially in the Enterprise systems the primary means of deallocating predictively bad memory in the system is leveraging the extensive set of spare DRAMs provided.

However, there are other mechanisms that could be leveraged through PowerVM for handling predictive memory faults. These can include deallocating a single page of memory through the OS for a single cell fault in memory.

If an uncorrectable fault occurs in a logical memory block used by a partition, the logical memory block can also be deallocated by the hypervisor to prevent its re-use.

Because of the virtualization approach used by PowerVM even if an entire DIMM is predictively faulty with no further ability to spare out at the DRAM level, PowerVM could migrate out the bad memory and replace it with good memory if there were sufficient spare capacity available.

It should be understood that how much spare memory is required depends on how the memory is interleaved in the system. Typically, memory is interleaved across multiple DIMMs and when deallocating memory, the entire set of memory in a interleave group has be to deallocated and substituted for the equivalent amount of spare memory.
RAS beyond Processors and Memory

Introduction

Processor and memory differences aside, there are many design considerations and features that distinguish between scale-out and enterprise systems and between different enterprise systems as well. One of these is the amount of infrastructure redundancy provided.

In theory, providing redundant components in a system may be thought of as a highly effective way to avoid outages due to failures of a single component. The care with which redundancy is designed and implemented plays an important factor on how effective redundancy really is in eliminating failures in a subsystem.

There are a number of considerations as described below.

Serial Failures, Load Capacity and Wear-out

Nearly any system running enterprise applications will supply redundant power supplies. This power supply redundancy is typically meant to assure that if one power supply fails, the other can take up the power load and continue to run. Likewise, most systems are designed to tolerate a failure in fans needed to cool the system. More sophisticated systems may increase fan speed to compensate for a failing fan, but a single fan failure itself should not cause a significant cooling issue.

In such a design, it would be expected that a system would never experience an outage due to a single power supply or fan fault. However, if a power supply fails in a redundant design and the second power supply should happen to fail before it is repaired, then the system will obviously be down until one or the other of the supplies is fixed. The expectation is that this would be a rare event.

If the system were incapable of determining that one of a pair of redundant parts had failed, then this can be more common, however. The ability to constantly monitor the health of the secondary component is therefore essential in a redundant design, but not always easy.

For example, two power supplies may share the load in a system by supplying power to components. When no power is supplied to a component, that condition is fairly easy to detect. If one power supply were able to supply some current, but an insufficient amount to carry the load by itself, depending on the design, the fact that the supply is “weak” may not be detected until the good supply fails.

In a lightly loaded system, it may not even be possible to distinguish between a “weak” supply and one that is providing no current at all. In some redundant designs for light loads, only one supply may even be configured to carry the load.

Even when both supplies are operating optimally, if a power supply is not well tested, designed and specified to run a system indefinitely on a single power source; it may happen that when the first power supply fails, the second carries a load that stresses it to the point where it soon also fails.

This kind of failure mode can be exasperated perhaps by environmental conditions: Say the cooling in the system is not very well designed so that a power supply runs hotter than it should. If this can cause a failure of the first supply over time, then the back-up supply might not be able to last much longer under the best of circumstances, and when taking over a load would soon be expected to fail.

As another example, fans can also be impacted if they are placed in systems that provide well for cooling of the electronic components, but where the fans themselves receive excessively heated air that is a detriment to the fans long-term reliability.

Therefore, understanding that excessive heat is one of the primary contributors to component “wear-out”, IBM requires that even components providing cooling to other components should be protected from excessive heat.

Common Mode Failures

Still even with well-designed redundancy and elimination of serial failures, and attention to component cooling, some faults can occur within a sub-system where redundancy is insufficient to protect against outages.
An easily understood example is when two power supplies are given the same AC power source. If that source fails, then the system will go down despite the redundancy. But failures include events besides simple power loss. They can include issues with surges due to electrical storm activity, short dips in power due to brown-out conditions, or perhaps when converting to backup power generation.

These incidents will be seen by both supplies in a redundant configuration and all the power supplies need to be able to withstand transient faults.

As another example, suppose that two supplies end up providing voltage to a common component, such as a processor module. If any power input to the module were to be shorted to ground, it would be expected that both supplies would see this fault. Both would have to shut down to prevent an over-current condition.

In an installed and running system, one would rarely expect to see wires themselves suddenly shorting to ground. However, if a component such as a cable or card were allowed to be removed or replaced in a system, without a good design to protect against it, even an experienced person doing that service activity could cause a short condition.

Also, the failure of certain components may essentially manifest as a short from power to ground. Something as simple as a capacitor used for electrical noise decoupling could fail in that fashion.

Proper system design would mitigate the impact of these events by having soft-switches, or effectively circuit breakers isolating key components, especially those that can be hot-plugged.

Ultimately there will be someplace electrically where redundant components come together to provide a function, and failures there can cause outages.

**Power and Cooling Redundancy Details**

**Power Supply Redundancy**

Power supplies in a system broadly refer to components that take utility power from the data system (typically alternating current, or AC power) and convert to DC power that is distributed throughout the system. Power supplies generally supply one DC voltage level (e.g. 12 volts) and voltage regulators may be used to supply different voltage levels required by various system voltages.

Power supply redundancy has two main goals. The first is to make sure that a system can continue to operate when a power supply fails. This is usually known as power supply redundancy. Conceptually if a system had four power supplies and could continue to run with one supply failed, that would be considered n+1 redundancy with 3 supplies needed for operation and 1 redundant supply.

The second goal is to allow for the data center to supply two sources of power into the system (typically using two power distribution units or PDUs). Should one of these sources fail, the system should continue to operate.

In theory this could be achieved by feeding each power supply with two separate line cords, one from each PDU. Depending on the design and how one power source failed, however, there could be scenarios of power supply failure where operation cannot be maintained.

Alternative, multiple power supplies might be supplied. For example, a system may be designed with four power supplies, E1, E2, E3 and E4 with E1 and E2 designed to connect to one power distribution unit (PDU) and E3 and E4 to another PDU.

If the system can run one of the PDUs failing, even if it causes some performance degradation for some loads, it may be said to also have “line cord redundancy.”

If 4 power supplies are used in this fashion then when a PDU fails, the system will be running with two power supplies. It may be expected, therefore that this would provide 2+2 or n+2 redundancy. However, this need not always be the case. The system may be designed to operate when certain power supplies fail that are supplied to a single PDU, but not when one power supply under each PDU fails. Or it may be designed to run with two supplies but have performance degradation for certain configurations/loads.

**Voltage Regulation**

There are many different designs that can be used for supplying power to components in a system.
As described above, power supplies may that takes alternating current (AC) from a data center power source, and then converts that to a direct current voltage level (DC).

Modern systems are designed using multiple components, not all of which use the same voltage level. Possibly a power supply either can provide multiple different DC voltage levels to supply all the components in a system. Failing that, it may supply a voltage level (e.g. 12v) to voltage regulators which then convert to the proper voltage levels needed for each system component (e.g. 1.6 V, 3.3 V, etc.) Use of such voltage regulators can also ensure that voltage levels are maintained within the tight specifications required for the modules they supply.

Typically, a voltage regulator module (VRM) has some common logic plus a component or set of components (called converters, channels or phases). At minimum, a VRM provides one converter (or phase) that provides the main function of stepped-down voltage, along with some control logic. Depending on the output load required, however, multiple phases may be used in tandem to provide that voltage level.

If the number of phases provided is just enough for the load it is driving, the failure of a single phase can lead to an outage. This can be true even when the 12V power supplies are redundant. Therefore, additional phases may be supplied to prevent the failure due to a single phase fault. Additional phases may also be provided for sparing purposes. The distinction between spare and redundant is that when a spare phase fails, the system continues to operate without the need to repair. After any spare phases fail, the failure of a redundant phase will require a service action.

While it is important to provide redundancy to critical components within a device such as an adapter or DIMM may not be necessary if the devices themselves are redundant within the system.

There are other applications for voltage conversion or division that are not as power-demanding as the applications above: A regulator only used during IPL for initialization of a component for example, or a memory DIMM or riser which takes supplied voltage and further divides on card for purposes such as reference voltage or signal termination. Such uses are not included in the discussion of voltage regulation or voltage regulator modules discussed in the rest of this paper.

POWER9 1 and 2 socket systems as scale-out models do not provide redundant voltage phases. This is very typical of systems in the scale-out space and often in systems that are considered Enterprise.

Generally speaking the E950 provides an n+1 redundant phase for VRMs feeding the critical components previously mentioned: the processor, the VRMs going out to the memory riser and VRMs for certain other components.

The E980 systems provides this n+1 redundancy to the equivalent elements (such as to the processors and to the custom DIMMs) but goes a step further and provides an additional spare phase for these elements.

**Redundant Clocks**

Vendors looking to design enterprise class servers may recognize the desirability of maintaining redundant processor clocks so that the failure of a single clock oscillator doesn’t cause a system to go down and stay down until repaired. However, a system vendor cannot feed redundant clocks all the way to a processor unless the processor itself is designed to accept redundant clock sources.

The POWER9 scale-up processor is designed with redundant clock inputs. In POWER8 systems a global clock for all of the processor components was required. Hence two global processor clock modules were provided in the system control drawer of the systems and a dynamic failover was provided.

In the E980 the design has been changed so that the main (run-time) processor core clock only need to stay synchronized within each CEC drawer (node). Hence the processor clock logic is now duplicated for each drawer in the system.

In this design the clock logic for the processor is now separate from a multi-function clock used for such components as PCIE bus. A fault on this multi-function clock where redundancy is provided can be tolerated through PCIe recovery.
Internal and External Cache Coherent Accelerators

To accelerate certain computational workloads beyond the traditional computational units in each core, POWER8 had a facility called the NX unit which provided a cache coherent interface to certain processor level resources such as a random number generator.

In addition, the Coherent Accelerator Processor Proxy (CAPP) module provided a way to interface to computational accelerators external to the processor. The CAPP communicated through PCIe adapter slots through a Coherent Accelerator Processor Interface (CAPI) which allowed the connected adapters, unlike regular I/O devices, to participate in the systems cache coherency domain through the CAPP.

POWER9 still has an NX unit for certain system resources and support for CAPI (at a 2.0 level to be consistent with Gen4 PCIe) though the internal design for these has somewhat changed.

It should be noted, however, that like any I/O adapter design, the RAS characteristics of what is attached to through the link is dependent on the design of the specific accelerator attached. These can vary according to the design and purpose of such accelerators.

Service Processor and Boot Infrastructure

There is more function than just the service processor itself that can be considered part of the service processor infrastructure in a system.

Systems with a single service processor still have some service processor infrastructure that may be redundant (i.e. having two system VPD modules on a VPD card.) Still, the E980 system provides the highest degree of service processor infrastructure redundancy among the systems being discussed.

In particular, it should be noted that for a system to boot or IPL, a system needs to have a healthy service processor, a functioning processor to boot from (using the internal self-boot engine) as well as functioning firmware.

In systems with a single service processor there is a single processor module and self-boot-engine that can be used for booting, and a single module used to store the associated firmware images.

In the E980, each system has two service processors. Each service processor can use a different processor module’s self-boot-engine on each node and each of these two processor modules has access to a different firmware module.

Trusted Platform Module (TPM)

Each of the systems discussed in this paper also incorporates a Trusted Platform Module (TPM) for support of Secure Boot and related functions. For redundancy purposes, The E980 system ships with two TPMs per node. Other systems have a single TPM.

Within the active service processor ASMI interface, a “TPM required” policy can be enabled. A system or node will not boot if the system is in secure mode, TPM is required, and no functional TPM is found.

I/O Subsystem and VIOS™

The descriptions for each system describe how internal I/O slots and external I/O drawers can be used for PCIe adapters.

In a PCIe environment, not all I/O adapters require full use of the bandwidth of a bus; therefore, lane reduction can be used to handle certain faults. For example, in an x16 environment, loss of a single lane, depending on location, could cause a bus to revert to a x8, x4, x2 or x1 lane configuration in some cases. This, however, can impact performance.

Not all faults that impact the PCIE I/O subsystem can be handled just by lane reduction. It is important when looking at I/O to not just consider all faults that cause loss of the I/O adapter function.

Power Systems are designed with the intention that traditional I/O adapters will be used in a redundant fashion. In the case, as discussed earlier where two systems are clustered together, the LAN adapters used to communicate between processors and the SAN adapters would all be redundant.

In a single 1s or 2s system, that redundancy would be achieved by having one of each type of adapter physically plugged into a PCIe socket controlled by one PCIe processor, and the other in a slot controlled by another.

The software communicating with the SAN would take care of the situation that one logical SAN device might be addressed by one of two different I/O adapters.

For LAN communicating heart-beat messages, both LAN adapters might be used, but messages coming from either one would be acceptable, and so forth.

This configuration method would best ensure that when there is a fault impacting an adapter, the redundant adapter can take over. If there is a fault impacting the communication to a slot from a processor, the other processor would be used to communicate to the other I/O adapter.

The error handling throughout the I/O subsystem from processor PCIe controller to I/O adapter would ensure that when a fault occurs anywhere on the I/O path, the fault can be contained to the partition(s) using that I/O path.

Furthermore, PowerVM supports the concept of I/O virtualization with VIOS™ so that I/O adapters are owned by I/O serving partitions. A user partition can access redundant I/O servers so that if one fails because of an I/O subsystem issue, or even a software problem impacting the server partition, the user partition with redundancy capabilities as described should continue to operate.

This End-to-End approach to I/O redundancy is a key contributor to keeping applications operating in the face of practically any I/O adapter problem. This concept is illustrated below using a figure first published in the POWER8 RAS whitepaper.

Figure 26: End-to-End I/O Redundancy

**PCIe Gen3 Expansion Drawer Redundancy**

As elsewhere described the optically connected PCIe Gen3 I/O expansion drawer provides significant RAS features including redundant fans/power supplies and independently operating I/O modules. Certain components such as the mid-plane, will require that the drawer be powered off during the repair and could potentially impact operation of both I/O modules.

For the highest level of redundancy, it is recommended that redundant adapter pairs be connected to separate I/O drawers, and these separate I/O drawers be connected to different processor modules where possible.
**Planned Outages**

Unplanned outages of systems and applications are typically very disruptive to applications. This is certainly true of systems running standalone applications, but is also true, perhaps to a somewhat lesser extent, of systems deployed in a scaled-out environment where the availability of an application does not entirely depend on the availability of any one server. The impact of unplanned outages on applications in both such environments is discussed in detail in the next section.

Planned outages, where the end-user picks the time and place where applications must be taken off-line can also be disruptive. Planned outages can be of a software nature – for patching or upgrading of applications, operating systems or other software layers. They can also be for hardware, for reconfiguring systems, upgrading or adding capacity, and for repair of elements that have failed but have not caused an outage because of the failure.

If all hardware failures required planned downtime, then the downtime associated with planned outages in an otherwise well designed system would far-outpace outages due to unplanned causes.

While repair of some components cannot be accomplished with workload actively running in a system, design capabilities to avoid other planned outages are characteristic of systems with advanced RAS capabilities. These may include:

**Updating Software Layers**

Maintaining updated code levels up and down the software stack may avoid risks of unplanned outages due to code bugs. However, updating code can require planned outages of applications, partitions or entire systems.

Generally, POWER Systems are designed to allow a given level of “firmware” to be updated in the code used by service processors, the PowerVM hypervisor and other areas of system hardware, without needing an outage.
Migrating from one firmware level to another, where a level provides new function, is not supported dynamically.

Dynamic updating of hypervisors other than the PowerVM hypervisor and of operating systems and applications depend on the capabilities of each such software layer.

**Concurrent Repair**

When redundancy is incorporated into a design, it is often possible to replace a component in a system without taking any sort of outage.

As examples, Enterprise Power Systems support concurrently removeable and replaceable elements such as power supplies and fans.

In addition, Enterprise Power Systems as well as POWER9 processor-based 1s and 2s systems support concurrently removing and replacing I/O adapters.

**Integrated Sparing**

As previously mentioned, to reduce replacements for components that cannot be removed and replaced without taking down a system, Power Systems strategy includes the use of integrated spare components that can be substituted for failing ones.
Clustering and Cloud Support

**PowerHA SystemMirror**

IBM Power Systems running under PowerVM and AIX™ and Linux support a spectrum of clustering solutions. These solutions are designed to meet requirements not only for application availability as regards to server outages, but also data center disaster management, reliable data backups and so forth. These offerings include distributed applications such as with db2 pureScale™, HA solutions using clustering technology with PowerHA™ SystemMirror™ and disaster management across geographies with PowerHA SystemMirror Enterprise Edition™.

It is beyond the scope of this paper to discuss the details of each of the IBM offerings or other clustering software, especially considering the availability of other material.

**Live Partition Mobility**

However, Live Partition Mobility (LPM), available for Power Systems running PowerVM Enterprise Edition, will be discussed here in particular with reference to its use in managing planned hardware outages.

LPM is a technique that allows a partition running on one server to be migrated dynamically to another server.

**Figure 28: LPM Minimum Configuration**

In simplified terms, LPM typically works in an environment where all the I/O from one partition is virtualized through PowerVM and VIOS and all partition data is stored in a Storage Area Network (SAN) accessed by both servers.

To migrate a partition from one server to another, a partition is identified on the new server and configured to have the same virtual resources as the primary server including access to the same logical volumes as the primary using the SAN.

When an LPM migration is initiated on a server for a partition, PowerVM begins the process of dynamically copying the state of the partition on the first server to the server that is the destination of the migration.

Thinking in terms of using LPM for hardware repairs, if all the workloads on a server are migrated by LPM to other servers, then after all have been migrated, the first server could be turned off to repair components.

LPM can also be used for doing firmware upgrades or adding additional hardware to a server when the hardware cannot be added concurrently in addition to software maintenance within individual partitions.

When LPM is used, while there may be a short time when applications are not processing new workload, the applications do not fail or crash and do not need to be restarted. Roughly speaking then, LPM, allows for planned outages to occur on a server without suffering downtime that would otherwise be required.
Minimum Configuration

For detailed information on how LPM can be configured the following references may be useful: An IBM Redbook titled: *IBM PowerVM Virtualization Introduction and Configuration*⁴ as well as the document *Live Partition Mobility*⁵

In general terms LPM requires that both the system containing a partition to be migrated and the system being migrated have a local LAN connection using a virtualized LAN adapter. In addition, LPM requires that all systems in the LPM cluster be attached to the same SAN. If a single HMC is used to manage both systems in the cluster, connectivity to the HMC also needs to be provided by an Ethernet connection to each service processor.

The LAN and SAN adapters used by the partition must be assigned to a Virtual I/O server and the partitions access to these would be by virtual LAN (vLAN) and virtual SCSI (vSCSI) connections within each partition to the VIOS.

I/O Redundancy Configurations and VIOS

LPM connectivity in the minimum configuration discussion is vulnerable to a number of different hardware and firmware faults that would lead to the inability to migrate partitions. Multiple paths to networks and SANs is therefore recommended. To accomplish this, Virtual I/O servers (VIOS) can be used.

VIOS as an offering for PowerVM virtualizes I/O adapters so that multiple partitions will be able to utilize the same physical adapter. VIOS can be configured with redundant I/O adapters so that the loss of an adapter does not result in a permanent loss of I/O to the partitions using the VIOS.

Externally to each system, redundant hardware management consoles (HMCs) can be utilized for greater availability. There can also be options to maintain redundancy in SANs and local network hardware.

Figure 29: I/O Infrastructure Redundancy

Figure 29 generally illustrates multi-path considerations within an environment optimized for LPM.

Within each server, this environment can be supported with a single VIOS. However, if a single VIOS is used and that VIOS terminates for any reason (hardware or software caused) then all the partitions using that VIOS will terminate.

Using Redundant VIOS servers would mitigate that risk. There is a caution, however that LPM cannot migrate a partition from one system to another when a partition is defined to use a virtual adapter from a VIOS and that VIOS is not operating. Maintaining redundancy of adapters within each VIOS in addition to having redundant VIOS will avoid most faults that keep a VIOS from running. Where redundant VIOS are used, it should also be possible to remove the vscsi and vlan connections to a failed VIOS in a partition.
before migration to allow migration to proceed using the remaining active VIOS in a non-redundant configuration.

**Figure 30: Use of Redundant VIOS**

Since each VIOS can largely be considered as an AIX based partition, each VIOS also needs the ability to access a boot image, having paging space, and so forth under a root volume group or rootvg. The rootvg can be accessed through a SAN, the same as the data that partitions use. Alternatively, a VIOS can use storage locally attached to a server, either DASD devices or SSD drives such as the internal NVMe drives provided for the E980 and E950 systems. For best availability, the rootvgs should use mirrored or other appropriate RAID drives with redundant access to the devices.

**PowerVC™ and Simplified Remote Restart**

PowerVC is an enterprise virtualization and cloud management offering from IBM that streamlines virtual machine deployment and operational management across servers. The IBM Cloud PowerVC Manager edition expands on this to provide self-service capabilities in a private cloud environment; IBM offers a Redbook that provides a detailed description of these capabilities. As of the time of this writing: **IBM PowerVC Version 1.3.2 Introduction and Configuration** describes this offering in considerable detail.

Deploying virtual machines on systems with the RAS characteristics previously described will best leverage the RAS capabilities of the hardware in a PowerVC environment. Of interest in this availability discussion is that PowerVC provides a virtual machine remote restart capability, which provides a means of automatically restarting a VM on another server in certain scenarios (described below).

Systems with a Hardware Management Console (HMC) may also chose to leverage a simplified remote restart capability (SRR) using the HMC. This is described in IBM support technical document: **http://www-01.ibm.com/support/docview.wss?uid=isg3T7000738**

**Error Detection in a Failover Environment**

The conditions under which a failover is attempted It is important when talking about any sort of failover scenario. Some remote restart capabilities, for example, operate only after an error management system, e.g. an HMC reports that a partition is in an Error or Down State.

This alone might miss hardware faults that just terminate a single application or impact the resources that an application uses, without causing a partition outage. Operating system “hang” conditions would also not be detected.

In contrast, PowerHA leverages a heartbeat within a partition to determine when a partition has become unavailable. This allows for fail-over in cases where there is a software or other cause while a partition is not able to make forward progress even if an error is not recorded.

---


It is a consideration that for the highest level of application availability an application itself might want to leverage some sort of heartbeat mechanism to determine when an application is hung or unable to make forward progress.
Section 5: Reliability and Availability in the Data Center

The R, A and S of RAS

Introduction
All of the previous sections in this document discussed server specific RAS features and options. This section looks at the more general concept of RAS as it applies to any system in the data center. The goal is to briefly define what RAS is and look at how reliability and availability are measured. It will then discuss how these measurements may be applied to different applications of scale-up and scale-out servers.

RAS Defined
Mathematically, reliability is defined in terms of infrequently something fails.
At a system level, availability is about how infrequently failures cause workload interruptions. The longer the interval between interruptions, the more available a system is.
Serviceability is all about how efficiently failures are identified and dealt with, and how application outages are minimized during repair.
Broadly speaking systems can be categorized as "scale-up" or "scale-out" depending on the impact to applications or workload of a system being unavailable.
True scale-out environments typically spread work load among multiple systems so that the impact of a single system failing, even for a short period of time is minimal.
In scale-up systems, the impact of a server taking a fault, or even a portion of a server (e.g. an individual partition) is significant. Applications may be deployed in a clustered environment so that extended outages can in a certain sense be tolerated (e.g. using some sort of fail-over to another system) but even the amount of time it takes to detect the issue and fail-over to another device is deemed significant in a scale-up system.

Reliability Modeling
The prediction of system level reliability starts with establishing the failure rates of the individual components making up the system. Then using the appropriate prediction models, the component level failure rates are combined to give us the system level reliability prediction in terms of a failure rate.
In literature, however, system level reliability is often discussed in terms of Mean Time Between Failures (MTBF) for repairable systems rather than a failure rate. For example, 50 years Mean Time Between Failures. A 50 years MTBF may suggest that a system will run 50 years between failures, but means more like that given 50 identical systems, one in a year will fail on average over a large population of systems.
The following illustration explains roughly how to bridge from individual component reliability to system reliability terms with some rounding and assumptions about secondary effects:
Different Levels of Reliability

When a component fails, the impact of that failure can vary depending on the component.

A power supply failing, in a system with a redundant power supply will need to be replaced. By itself, however, a failure of a single power supply should not cause a system outage and should lead to a concurrent repair with no down-time to replace.

There are other components in a system might fail causing a system-wide outage where concurrent repair is not possible.

Therefore, it is typical to talk about different MTBF numbers.

For example:

- MTBF – Resulting Repair Actions
- MTBF – That require concurrent repair
- MTBF – That require a non-concurrent repair
MTBF – Resulting in an unplanned application outage
MTBF – Resulting in an unplanned system outage

Scale-out systems may invest in having a long MTBF for unplanned outages even if it means more recurrent repair actions.

**Costs and Reliability**

**Service Costs**

It is common for software costs in an enterprise to include the cost to acquire or license code and a recurring cost for software maintenance. There is also a cost associated with acquiring the hardware and a recurring cost associated with hardware maintenance – primarily fixing systems when components break.

Individual component failure rates, the cost of the components, and the labor costs associated with repair can be aggregated at the system level to estimate the direct maintenance costs expected for a large population of such systems.

The more reliable the components the less the maintenance costs should be.

Design for reliability can not only help with maintenance cost to a certain degree but also with the initial cost of a system as well – in some cases. For example, designing a processor with extra cache capacity, data lanes on a bus or so forth can make it easier to yield good processors at the end of a manufacturing process as an entire module need not be discarded due to a small flaw.

At the other extreme, designing a system with an entire spare processor socket could significantly decrease maintenance cost by not having to replace anything in a system should a single processor module fail. However, each system will incur the costs of a spare processor for the return of avoiding a repair in the small proportion of those that need repair. This is usually not justified from a systems cost perspective. Rather it is better to invest in greater processor reliability.

On scale-out systems redundancy is generally only implemented on items where the cost is relatively low, and the failure rates expected are relatively high – and in some cases where the redundancy is not complete. For example, power supplies and fans may be considered redundant in some scale-out systems because when one fails, operation will continue. However, depending on the design, when a component fails, fans may have to be run faster, and performance-throttled until repair.

On scale-up systems redundancy that might even add significantly to maintenance costs is considered worthwhile to avoid indirect costs associated with downtime, as discussed below.

**End User Costs**

Generally, of greater concern are the indirect costs that end users will incur whenever a service action needs to be taken. The costs are the least when a component fails and can be concurrently replaced, and the highest when a component fails such that the system goes down and must stay down until repaired.

The indirect cost typically depends on the importance of the workloads running in the system and what sort of mechanisms exist to cope with the fault. If an application is distributed across multiple systems and there is little to no impact to the application when a single system goes down, then the cost is relatively low and there is less incentive to invest in RAS. If there is some application downtime in such an environment, or at least a reduction in workload throughput, then the cost associated with the downtime can be quantified and the need for investing in greater RAS can be weighed against those costs.

The previous section discussed how the highest levels of availability are typically achieved even in scale-up environments by having multiple systems and some means of failing over to another in the event of a problem. Such failovers, even when relatively short duration can have costs. In such cases, investing in RAS at the server level can pay particular dividends.

When a failover solution is not employed, the impact to workload is obvious. Even in cases where failover is enabled, not every application may be considered critical enough to have an automated failover means. The impact of a prolonged service outage on these applications can be significant.
It is also significant to note that even when plans are put into place for automated failover when a fault occurs, there are times when failover does not happen as smoothly as expected. These are typically due to multiple factors, unavailability of the backup system, unintended code level mismatches, insufficient testing, and so forth.

Perhaps the less reliable the system, the more often the failover mechanisms might be tested, but also the more likely that some reason for an incident to occur in failover leading to an extended outage. The latter suggests also that investing in both hardware reliability and failover testing can be beneficial.

**Measuring Availability**

**Measuring Availability**

Mathematically speaking, availability is often expressed as a percentage of the time something is available or in use over a given period of time. An availability number for a system can be mathematically calculated from the expected reliability of the system so long as both the mean time between failures and the duration of each outage is known.

For example: Consider a system that always runs exactly one week between failures and each time it fails, it is down for 10 minutes. For the 168 hours in a week, the system is down (10/60) hours. It is up 168hrs – (10/60) hrs. As a percentage of the hours in the week, it can be said that the system is (168-(1/6)) *100% = 99.9% available.

99.999% available means approximately 5.3 minutes down in a year. On average, a system that failed once a year and was down for 5.3 minutes would be 99.999% available. This is often called 5 9’s of availability.

When talking about modern server hardware availability, short weekly failures like in the example above is not the norm. Rather the failure rates are much lower and the mean time between failures (MTBF) is often measured in terms of years – perhaps more years than a system will be kept in service.

Therefore, when a MTBF of 10 years, for example, is quoted, it is not expected that on average each system will run 10 years between failures. Rather it is more reasonable to expect that on average in a given year, one server out of ten will fail. If a population of ten servers always had exactly one failure a year, a statement of 99.999% availability across that population of servers would mean that the one server that failed would be down 530 minutes when it failed (over eight hours.)

In theory 5 9’s of availability can be achieved by having a system design which fails frequently, multiple times a year, but whose failures are limited to very small periods of time. Conversely 5 9’s of available might mean a server design with a very large MTBF, but where a given server takes a fairly long time to recover from the very rare outage.
The figure above shows that 5 9's of availability can be achieved with systems that fail frequently for miniscule amounts of time, or very infrequently with much larger downtime per failure.

The figure is misleading in the sense that servers with low reliability are likely to have many components that, when they fail, take the system down and keep the system down until repair. Conversely servers designed for great reliability often are also designed so that the systems, or at least portions of the system can be recovered without having to keep a system down until repaired.

Hence on the surface systems with low MTBF would have longer repair-times and a system with 5 9's of availability would therefore be synonymous with a high level of reliability.

However, in quoting an availability number, there needs to be a good description of what is being quoted. Is it only concerning unplanned outages that take down an entire system? Is it concerning just hardware faults, or are firmware, OS and application faults taken into account?

Are applications even taken into account? If they are, if multiple applications are running on the server, is each application outage counted individually? Or does one event causing multiple application outages count as a single failure?

If there are planned outages to repair components either delayed after an unplanned outage, or predictively, is that repair time included in the unavailability time? Or are only unplanned outages considered?

Perhaps most importantly when reading that a certain company achieved 5 9’s of availability for an application - is knowing if that number counted application availability running in a standalone environment? Or was that a measure of application availability in systems that might have a failover capability.

**Contributions of Each Element in the Application Stack**

When looking at application availability it is apparent that there are multiple elements that could fail and cause an application outage. Each element could have a different MTBF and the recovery time for different faults can also be different.

When an application crashes, the recovery time is typically just the amount of time it takes to detect that the application has crashed, recover any data if necessary to do so, and restart the application.

When an operating system crashes and takes an application down, the recovery time includes all the above, plus the time it takes for the operating system to reboot and be ready to restart the application.
An OS vendor may be able to estimate a MTBF for OS panics based on previous experience. The OS vendor, however, can’t really express how many 9s of availability will result for an application unless the OS vendor really knows what application a customer is deploying, and how long its recovery time is.

Even more difficulty can arise with calculating application availability due to the hardware.

For example, suppose a processor has a fault. The fault might involve any of the following:

1. Recovery or recovery and repair that causes no application outage.
2. An application outage and restart but nothing else
3. A partition outage and restart.
4. A system outage where the system can reboot and recover, and the failing hardware can subsequently be replaced without taking another outage
5. Some sort of an outage where reboot and recovery is possible, but a separate outage will eventually be needed to repair the faulty hardware.
6. A condition that causes an outage, but recovery is not possible until the failed hardware is replaced; meaning that the system and all applications running on it are down until the repair is completed.

The recovery times for each of these incidents is typically progressively longer, with the final case very dependent on how quickly replacement parts can be procured and repairs completed.

Figure 33 is an example with hypothetical failure rates and recovery times for the various situations mentioned above, looking at a large population of standalone systems each running a single application.

**Figure 33: Standalone System Availability Considerations**

<table>
<thead>
<tr>
<th>Outage Reason</th>
<th>Mean time to Outage (in Years)</th>
<th>Recovery Activities Needed</th>
<th>Total Recovery minutes/Incident</th>
<th>Minutes Down Per Year</th>
<th>Associated Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Limited To Application</td>
<td>3</td>
<td>x</td>
<td>7.00</td>
<td>2.33</td>
<td>99.99958%</td>
</tr>
<tr>
<td>Fault Causing OS crash</td>
<td>10</td>
<td>x</td>
<td>11.00</td>
<td>1.10</td>
<td>99.99979%</td>
</tr>
<tr>
<td>Fault causing hypervisor crash</td>
<td>80</td>
<td>x</td>
<td>16.00</td>
<td>0.20</td>
<td>99.99996%</td>
</tr>
<tr>
<td>Fault impacting system (crash) but system recovers on reboot with enough resources to restart application</td>
<td>80</td>
<td>x</td>
<td>26.00</td>
<td>0.33</td>
<td>99.99994%</td>
</tr>
<tr>
<td>Planned hardware repair for hw fault (where initial fault impact could be any of the above)</td>
<td>70</td>
<td>x</td>
<td>56.00</td>
<td>0.80</td>
<td>99.99985%</td>
</tr>
<tr>
<td>Fault impacting system where application is down until system is repaired</td>
<td>500</td>
<td>x</td>
<td>236.00</td>
<td>0.47</td>
<td>99.99991%</td>
</tr>
</tbody>
</table>

This is not intended to represent any given system. Rather, it is intended to illustrate how different outages have different impacts. An application crash only requires that the crash be discovered and the application restarted. Hence there is only an x in the column for the 7 minutes application restart and recovery time.
If an application is running under an OS and the OS crashes, then the total recovery time must include
the time it takes to reboot the OS plus the time it takes to detect the fault and recover the application after
the OS reboots. In the example with an x in each of the first two columns the total recovery time is 11
minutes (4 minutes to recover the OS and 7 for the application.)

The worst-case scenario as described in the previous section is a case where the fault causes a system
to go down and stay down until repaired. In the example, with an x in all the recovery activities column,
that would mean 236 minutes of recovery for each such incident.

In the example numbers were chosen to illustrate 5 9s of availability across a population of systems.

This required that the worst-case outage scenarios to be extremely rare compared to the application only-
outages.

In addition, the example presumed that:

1. All the software layers can recover reasonably efficiently even from entire system crashes.
2. There were no more than a reasonable number of application driven and operating system driven
   outages.
3. A very robust hypervisor is used, expecting it to be considerably more robust that the application
   hosting OS.
4. Exceptionally reliable hardware is used. (The example presumes about 70 MTBF for hardware
   faults.)
5. Hardware that can be repaired efficiently, using concurrent repair techniques for the vast majority
   of the faults.
6. As previously mentioned, the system design ensures few faults exist that could keep a system
down until repaired. In the rare case that such a fault does occur it presumably an efficient support
   structure that can rapidly deliver the failed part to the failing system and efficiently make the
   repair.
7. A method of insuring quick restart of a system after hardware outages, which might impact the
   ability to do extensive fault analysis.

It must also be stressed that the example only looks at the impact of hardware faults that caused some
sort of application outage. It does not deal with outages for hardware or firmware upgrades, patches, or
repairs for failing hardware that have not caused outages.

**Critical Application Simplification**

Under a single operating system instance, it is possible to run multiple applications of various kinds,
though typically only one important application is deployed. Likewise, in a system with multiple operating
system partitions, it is possible to run multiple applications using multiple partitions.

To calculate the availability of each such application, throughout an entire system, calculations would
have to change to account for the number of partitions and the outage time associated with each for each
fault that could be experienced. The aggregate availability percentage would represent an aggregation of
many different applications, not all of equal importance.

Therefore, in the examples in this section, a simplifying assumption is made that each server is running a
single application of interest to availability calculations. In other words, the examples look at availability of
a critical applications presuming one such application per system.

**Measuring Application Availability in a Clustered Environment**

It should be evident that clustering can have a big impact on application availability since, if recovery time
after an outage is required for the application, the time for nearly every outage can be reliably predicted
and limited to just that “fail-over” time.

Figure 34 shows what might be achieved with the earlier proposed hypothetical enterprise hardware
example in such a clustered environment.
Figure 34: Ideal Clustering with Enterprise-Class Hardware Example

<table>
<thead>
<tr>
<th>Outage Reason</th>
<th>Mean time to Outage (in Years)</th>
<th>Total Recovery minutes/Incident</th>
<th>Minutes Down Per Year</th>
<th>Associated Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Limited To Application</td>
<td>3</td>
<td>6.00</td>
<td>2.00</td>
<td>99.99962%</td>
</tr>
<tr>
<td>Fault Causing OS crash</td>
<td>10</td>
<td>6.00</td>
<td>0.60</td>
<td>99.99989%</td>
</tr>
<tr>
<td>Fault causing hypervisor crash</td>
<td>80</td>
<td>6.00</td>
<td>0.08</td>
<td>99.99999%</td>
</tr>
<tr>
<td>Fault impacting system (crash) but system recovers on reboot with enough resources to restart application</td>
<td>80</td>
<td>6.00</td>
<td>0.08</td>
<td>99.99999%</td>
</tr>
<tr>
<td>Planned hardware repair for hw fault (where initial fault impact could be any of the above)</td>
<td>70</td>
<td>6.00</td>
<td>0.09</td>
<td>99.99998%</td>
</tr>
<tr>
<td>Fault impacting system where application is down until system is repaired</td>
<td>500</td>
<td>6.00</td>
<td>0.01</td>
<td>100.00000%</td>
</tr>
<tr>
<td>Total for all Outage Reasons</td>
<td></td>
<td>2.85</td>
<td></td>
<td>99.99946%</td>
</tr>
</tbody>
</table>

Similar to the single-system example, it shows the unavailability associated with various failure types. However, it presumes that application recovery occurs by failing over from one system to another. Hence the recovery time for any of the outages is limited to the time it takes to detect the fault and fail-over and recover on another system. This minimizes the impact of faults that in the standalone case, while rare, would lead to extended application outages.

The example suggests that fail-over clustering can extend availability beyond what would be achieved in the standalone example.

Recovery Time Caution

The examples given presume that it takes about six minutes to recover from any system outage. This may be realistic for some applications, but not for others. As previously shown, doubling the recovery time has a corresponding large impact on the availability numbers.

Whatever the recovery time associated with such a fail-over event is for a given application needs to be very well understood. Such a HA solution is really no HA solution at all if a service level agreement requires that no outage exceed, for example, 10 minutes, and the HA fail-over recovery time is 15 minutes.

Clustering Infrastructure Impact on Availability

A clustering environment might be deployed where a primary server runs with everything it needs under the covers, including maintaining all the storage, data and otherwise, within the server itself. This is sometimes referred to as a “nothing shared” clustering model.

In such a case, clustering involves copying data from one server to the backup server, with the backup server maintaining its own copy of the data. The two systems communicate by a LAN and redundancy is achieved by sending data across the redundant LAN environment.
It might be expected in such a case that long outages would only happen in the relatively unlikely 
scenarios where both servers are down simultaneously, both LAN adapters are down simultaneously, or 
there is a bug in the failover itself.

Hardware, software and maintenance practices must ensure that high availability for this infrastructure is 
achieved if high application availability is to be expected.

The “nothing shared” scenario above does not automatically support the easy migration of data from one 
server to another for planned events that don’t involve a failover.

An alternative approach makes use of a shared common storage such as a storage area network (SAN), 
where each server has accessed to and only makes use of the shared storage.

**Real World Fail-over Effectiveness Calculations**

The previous examples do also presume that such high availability solutions are simple enough to 
implement that they can be used for all applications and that fail-over, when initiated, always works.

In the real world that may not always be the case. As previously mentioned, if the secondary server is 
down for any reason when a failover is required, then failover is not going to happen and the application 
in question is going to be down until either the primary or secondary server is restored. The frequency of 
such an event happening is directly related to the underlying availability characteristics of each individual 
server – the more likely the primary server is to fail, the more likely it needs to have the backup available 
and the more likely the backup server is to be down, the less likely it will be available when needed.

Any necessary connections between the two must also be available, and that includes any shared 
storage.

It should be clear that if the availability of an application is to meet 5 9s of availability, then shared storage 
must have better than 5 9s of availability.

Different kinds of SAN solutions may have sufficient redundancy of hard drives to assure greater than 5 
9s availability of data on the drives but may fail in providing availability of the I/O and control that provide 
access to the data.

The most highly available SAN solutions may require redundant servers under the cover using their own 
form of fail-over clustering to achieve the availability of the SAN.

Advanced HA solutions may also take advantage of dual SANs to help mitigate against SAN outages, 
and support solutions across multiple data centers. This mechanism effectively removes the task of 
copied data from the servers and puts it on the SAN.

Such mechanism can be very complicated both in the demands of the hardware and in the controller 
software.

The role of software layers in cluster availability is crucial, and it can be quite difficult to ensure that all the 
software deployed is bug-free. Every time an application fails, it may fail at a different point. Ensuring 
detection of every hardware fault and data-lossless recovery of an application for every possible failure is 
complicated by the possibility of exercising different code paths on each failure. This difficulty increases 
the possibility of latent defects in the code. There are also difficulties in making sure that the fail-over 
environment is properly kept up to date and in good working order with software releases that are 
consistently patched and known to be compatible.

To ensure proper working order of a fail-over solution as regards to all of the above, testing may be 
worthwhile, but live testing of the fail-over solution itself can add to application unavailability.

When the fail-over solution does not work as intended, and especially when something within the 
clustering infrastructure goes wrong, the recovery time can be long. Likewise if a primary server fails 
when the secondary is not available, or if the state of the transactions and data from the one server 
cannot be properly shadowed, recovery can include a number of long-downtime events such as waiting 
on a part to repair a server, or the time required to rebuild or recover data.

A good measurement of availability in a clustered environment should therefore include a factor for the 
efficiency of the fail-over solution implemented; having some measure for how frequently a fail-over fails 
and how long it takes to recover from that scenario.
In the figures below, the same Enterprise and Non-Enterprise clustering examples are evaluated with an added factor that one time in twenty a fail-over event doesn’t go as planned and recovery from such events takes a number of hours.

**Figure 35: More Realistic Model of Clustering with Enterprise-Class Hardware**

<table>
<thead>
<tr>
<th>Outage Reason</th>
<th>Mean Time to Outage (in Years)</th>
<th>Total Recovery minutes/Incident</th>
<th>Minutes Down Per Year</th>
<th>Mean Time to Fail-over Issue (years)</th>
<th>Additional Time To Account for Fail-over issues (minutes)</th>
<th>Total Minutes Down Per Year</th>
<th>Availability Associated with Fault Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Limited To Application</td>
<td>3</td>
<td>6.00</td>
<td>2.00</td>
<td>60</td>
<td>1.00</td>
<td>1</td>
<td>99.99943%</td>
</tr>
<tr>
<td>Fault Causing OS crash</td>
<td>10</td>
<td>6.00</td>
<td>0.60</td>
<td>200</td>
<td>0.3</td>
<td>0.90</td>
<td>99.9983%</td>
</tr>
<tr>
<td>Fault Causing hypervisor crash</td>
<td>80</td>
<td>6.00</td>
<td>0.08</td>
<td>1600</td>
<td>0.0375</td>
<td>0.11</td>
<td>99.9998%</td>
</tr>
<tr>
<td>Fault impacting system (crash) but system recovers on reboot with enough resources to restart application</td>
<td>80</td>
<td>6.00</td>
<td>0.08</td>
<td>1600</td>
<td>0.0375</td>
<td>0.11</td>
<td>99.9998%</td>
</tr>
<tr>
<td>Planned hardware repair for hw fault (where initial fault impact could be any of the above)</td>
<td>70</td>
<td>6.00</td>
<td>0.09</td>
<td>1400</td>
<td>0.042857143</td>
<td>0.13</td>
<td>99.9998%</td>
</tr>
<tr>
<td>Fault impacting system where application is down until system is repaired</td>
<td>500</td>
<td>6.00</td>
<td>0.01</td>
<td>10000</td>
<td>0.006</td>
<td>0.02</td>
<td>100.00000%</td>
</tr>
</tbody>
</table>

Total Minutes of downtime per year 4.27
Availability 99.99919%

**Figure 36: More Realistic Clustering with Non-Enterprise-Class Hardware**

<table>
<thead>
<tr>
<th>Outage Reason</th>
<th>Mean Time to Outage (in Years)</th>
<th>Total Recovery minutes/Incident</th>
<th>Minutes Down Per Year</th>
<th>Mean Time to Fail-over Issue (years)</th>
<th>Additional Time To Account for Fail-over issues (minutes)</th>
<th>Total Minutes Down Per Year</th>
<th>Availability Associated with Fault Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Limited To Application</td>
<td>3</td>
<td>6.00</td>
<td>2.00</td>
<td>60</td>
<td>2.5</td>
<td>4.50</td>
<td>99.99914%</td>
</tr>
<tr>
<td>Fault Causing OS crash</td>
<td>10</td>
<td>6.00</td>
<td>0.60</td>
<td>200</td>
<td>0.75</td>
<td>1.35</td>
<td>99.9974%</td>
</tr>
<tr>
<td>Fault Causing hypervisor crash</td>
<td>30</td>
<td>6.00</td>
<td>0.20</td>
<td>800</td>
<td>0.25</td>
<td>0.45</td>
<td>99.99991%</td>
</tr>
<tr>
<td>Fault impacting system (crash) but system recovers on reboot with enough resources to restart application</td>
<td>40</td>
<td>6.00</td>
<td>0.15</td>
<td>800</td>
<td>0.1875</td>
<td>0.34</td>
<td>99.99994%</td>
</tr>
<tr>
<td>Planned hardware repair for hw fault (where initial fault impact could be any of the above)</td>
<td>40</td>
<td>6.00</td>
<td>0.15</td>
<td>800</td>
<td>0.1875</td>
<td>0.34</td>
<td>99.99994%</td>
</tr>
<tr>
<td>Fault impacting system where application is down until system is repaired</td>
<td>40</td>
<td>6.00</td>
<td>0.15</td>
<td>800</td>
<td>0.1875</td>
<td>0.34</td>
<td>99.99994%</td>
</tr>
</tbody>
</table>

Total Minutes of downtime per year 7.31
Availability 99.99861%

The example presumes somewhat longer recovery for the non-enterprise hardware due to the other kinds of real-world conditions described in terms of parts acquisition, error detection/fault isolation (ED/FI) and so forth.
Though these examples presume too much to be specifically applicable to any given customer environment, they are intended to illustrate two things:

*The less frequently the hardware fails, the better the ideal availability, and the less perfect clustering must be to achieve desired availability.*

*If the clustering and failover support elements themselves have bugs/pervasive issues, or single points of failure besides the server hardware, less than 5 9s of availability (with reference to hardware faults) may still occur in a clustered environment. It is possible that availability might be worse in those cases than in comparable stand-alone environment.*

**Reducing the Impact of Planned Downtime in a Clustered Environment**

The previous examples did not look at the impact of planned outages except for deferred repair of a part that caused some sort of outage.

Planned outages of systems may in many cases occur more frequently than unplanned outages. This is especially true of less-than-enterprise class hardware that:

1. Require outages to patch code (OS, hypervisor, etc.)
2. Have no ability to repair hardware using integrated sparing and similar techniques and instead must predictively take off-line components that may otherwise subsequently fail and require an outage to repair.
3. Do not provide redundancy and concurrent repair of components like I/O adapters.

In a clustered environment, it seems reasonable that when it is known in advance that a system needs to be taken down during a planned maintenance window, that recovery and fail-over times could be minimized with some advanced planning. Still, so long as fail-over techniques are used for the planned outages, one should still expect recovery time to be in the minutes range.

However, it is also possible to take advantage of a highly virtualized environment to migrate applications from one system to another in advanced of a planned outage, without having to recover/restart the applications.

PowerVM Enterprise Edition™ offers one such solution called Live Partition Mobility (LPM). In addition to use in handling planned hardware outages, LPM can mitigate downtime associated with hardware and software upgrades and system reconfiguration and other such activities which could also impact availability and are otherwise not considered in even the “real world” 5 9s availability discussion.

**HA Solutions Cost and Hardware Suitability**

**Clustering Resources**

One of the obvious disadvantages of running in a clustered environment, as opposed to a standalone system environment, is the need for additional hardware to accomplish the task.

An application running full-throttle on one system, prepared to failover on another, needs to have a comparable capability (available processor cores, memory and so forth) on that other system.

There does not need to be exactly one back-up server for every server in production, however. If multiple servers are used to run work-loads, then only a single backup system with enough capacity to handle the workload of any one server might be deployed.

Alternatively, if multiple partitions are consolidated on multiple servers, then presuming that no server is fully utilized, fail-over might be planned so that one failing server will restart on different partitions on multiple different servers.

When an enterprise has sufficient workload to justify multiple servers, either of these options reduces the overhead for clustering.
There are several additional variations that could be considered.

In practice there is typically a limit as to how many systems are clustered together. These include concerns about increased risk of simultaneous server outages, relying too much on the availability of shared storage where shared storage is used, the overhead of keeping data in sync when shared storage is not used, and practical considerations ensuring that where necessary all systems aligned in terms of code-updates, hardware configuration and so forth.

It many cases it should also be noted that applications may have licensing terms that make clustering solutions more expensive. For example, applications, databases etc. may be licensed on a per core basis. If the license is not based on how many cores are used at any given time, but on how many specific cores in specific systems the application might run on, then clustering increases licensing cost.

**Using High Performance Systems**

In these environments, it becomes not only useful to have a system that is reliable, but also capable of being highly utilized. The more the processors and memory resources of a system are used, the fewer total system resources are required and that impacts the cost of backup server resources and licenses.

Power Systems are designed with per core performance in mind and highly utilized. Deploying PowerVM allows for a great depth in virtualization allowing applications to take the most advantage of the processing, I/O and storage capabilities. Power Systems thus have a natural affinity towards use in clustered environments where maximizing resources is important in reducing ownership costs.
Summary

Investing in RAS

Systems designed for RAS may be more costly at the "bill of materials" level than systems with little investment in RAS.

Some examples as to why this could be so:

In terms of error detection and fault isolation: Simplified, at the low level, having an 8-bit bus takes a certain amount of circuits. Adding an extra bit to detect a single fault, adds hardware to the bus. In a class Hamming code, 5 bits of error checking data might be required for 15 bits of data to allow for double-bit error detection, and single bit correction. Then there is the logic involved in generating the error detection bits and checking/correcting for errors.

In some cases, better availability is achieved by having fully redundant components which more than doubles the cost of the components, or by having some amount of n+1 redundancy or sparing which still adds costs at a somewhat lesser rate.

In terms of reliability, highly reliable components will cost more. This may be true of the intrinsic design, the materials used including the design of connectors, fans and power supplies.

Increased reliability in the way components are manufactured can also increase costs. Extensive time in manufacture to test, a process to "burn-in" parts and screen out weak modules increases costs. The highest levels of reliability of parts may be achieved by rejecting entire lots –even good components - when the failure rates overall for a lot are excessive. All of these increase the costs of the components.

Design for serviceability, especially for concurrent maintained typically is more involved than a design where serviceability is not a concern. This is especially true when designing, for example, for concurrent maintenance of components like I/O adapters.

Beyond the hardware costs, it takes development effort to code software to take advantage of the hardware RAS features and time again to test for the many various "bad-path" scenarios that can be envisioned.

On the other hand, in all systems, scale-up and scale-out, investing in system RAS has a purpose. Just as there is recurring costs for software licenses in most enterprise applications, there is a recurring cost associated with maintaining systems. These include the direct costs, such as the cost for replacement components and the cost associated with the labor required to diagnose and repair a system.

The somewhat more indirect costs of poor RAS are often the main reasons for investing in systems with superior RAS characteristics and overtime these have become even more important to customers. The importance is often directly related to:

Importance associated with discovery errors before relying on faulty data or computation including the ability to know when to switch over to redundant or alternate resources.

The costs associated with downtime to do problem determination or error recreation, if insufficient fault isolation is provided in the system.

The cost of downtime when a system fails unexpectedly, or needs to fail over when an application is disrupted during the failover process.

The costs associated with planning an outage to or repair of hardware or firmware, especially when the repair is not concurrent.

In a well-designed system investing in RAS minimizes the need to repair components that are failing. Systems that recover rather than crash and need repair when certain soft errors occur will minimize all of the indirect costs associated with such events. Use of selective self-healing so that, for example, a processor does not have to be replaced simply because a single line of data on an I/O bus has a fault reduces planned outage costs.
In scale-out environments the reliability of components and their serviceability can be measured and weighed against the cost associated with maintaining the highest levels of reliability in a system.

In a scale-up environment, the indirect costs of outages and failovers usually outweigh the direct costs of the repair. An emphasis is therefore put on designs that increase availability in the face of frequent costs – such as having redundancy – even when the result is higher system and maintenance costs.

Final Word

The POWER9 processor-based systems discussed leverage the long heritage of Power Systems designed for RAS. The different servers aimed at different scale-up and scale-out environments provide significant choice in selecting servers geared towards the application environments end-users will deploy. The RAS features in each segment differ but in each provide substantial advantages compared to designs with less of an up-front RAS focus.
About the principal/editor:

Daniel Henderson is an IBM Senior Technical Staff Member. He has been involved with POWER and predecessor RISC based products development and support since the earliest RISC systems. He is currently the lead system hardware availability designer for IBM Power Systems PowerVM based platforms.

Notices:

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information is intended to give a general understanding of concepts only. This information could include technical inaccuracies or typographical errors. Changes may be made periodically made to the information herein in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.