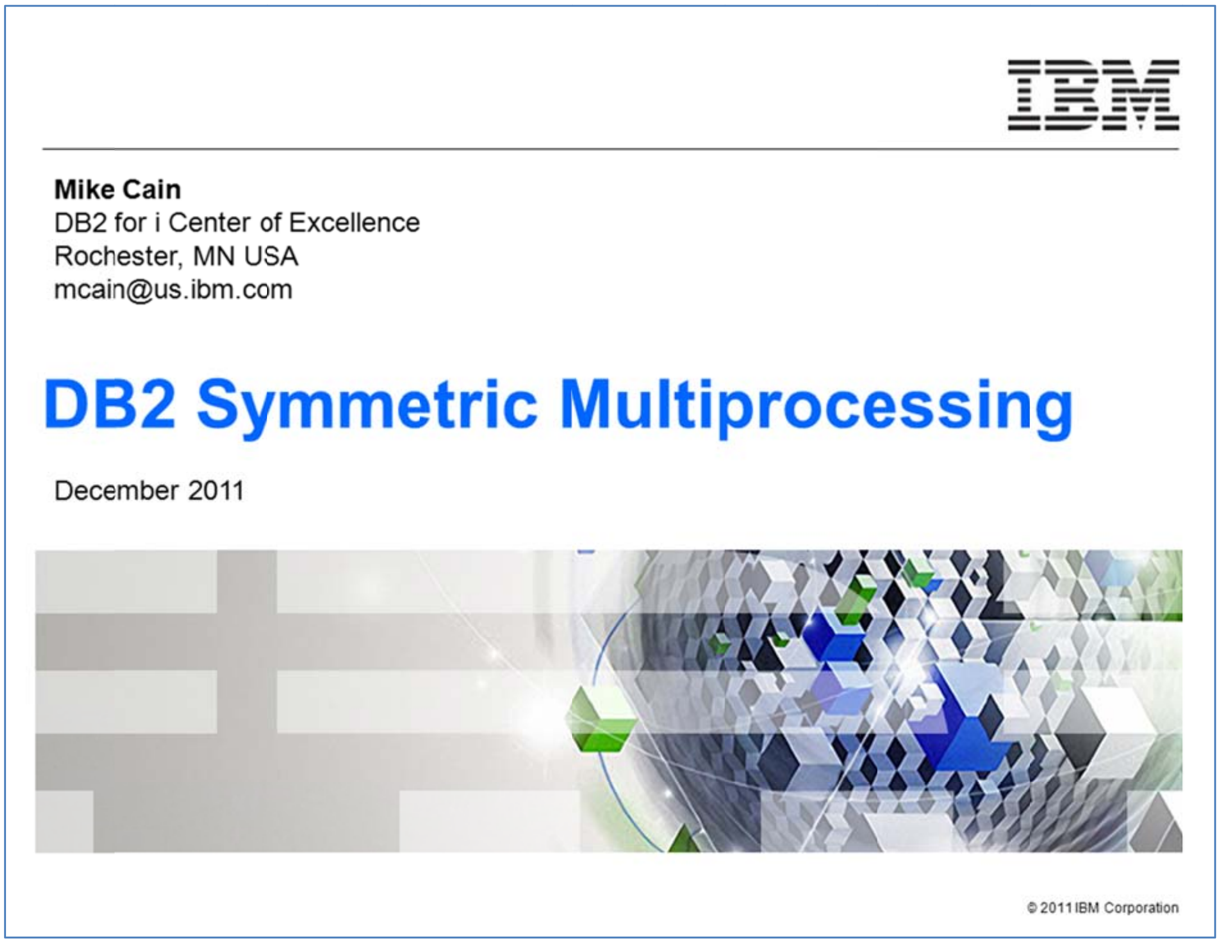
Introduction to
DB2 Symmetric Multiprocessing for IBM i

About the Author:

Mike Cain,
DB2 for i Center of Excellence,
IBM STG Lab Services and Training

Mike Cain is the leader of the DB2 for i Center of Excellence team in the IBM Systems and Technology Group Lab Services and Training organization in Rochester, Minnesota. Prior to his current position, he was the team leader of the AS/400 Teraplex Integration Center and worked as an IBM AS/400 systems engineer and technical consultant.

Mike can be reached at: mcain@us.ibm.com



Mike Cain
DB2 for i Center of Excellence
Rochester, MN USA
mcain@us.ibm.com

DB2 Symmetric Multiprocessing

December 2011

Introduction

Welcome to this online course, entitled DB2 Symmetric Multiprocessing: Database Parallelism within IBM i. The purpose of this course is to explain the DB2® Symmetric Multiprocessing (SMP) feature of IBM i™. The reader will learn about database parallelism within the IBM i operating system running on the IBM® POWER® platform.

IBM i was formerly called i5/OS®.

Prerequisites:
This course will be most useful to IBM i™ developers who code SQL queries to access DB2® for i data, or use solutions that make use of SQL to connect and retrieve data from DB2 for i.

Agenda

This course will first talk about the need for installing the DB2 Symmetric Multiprocessing (SMP) feature of IBM i. You will even learn how to determine if this feature has already been installed on your system or on a particular logical partition.

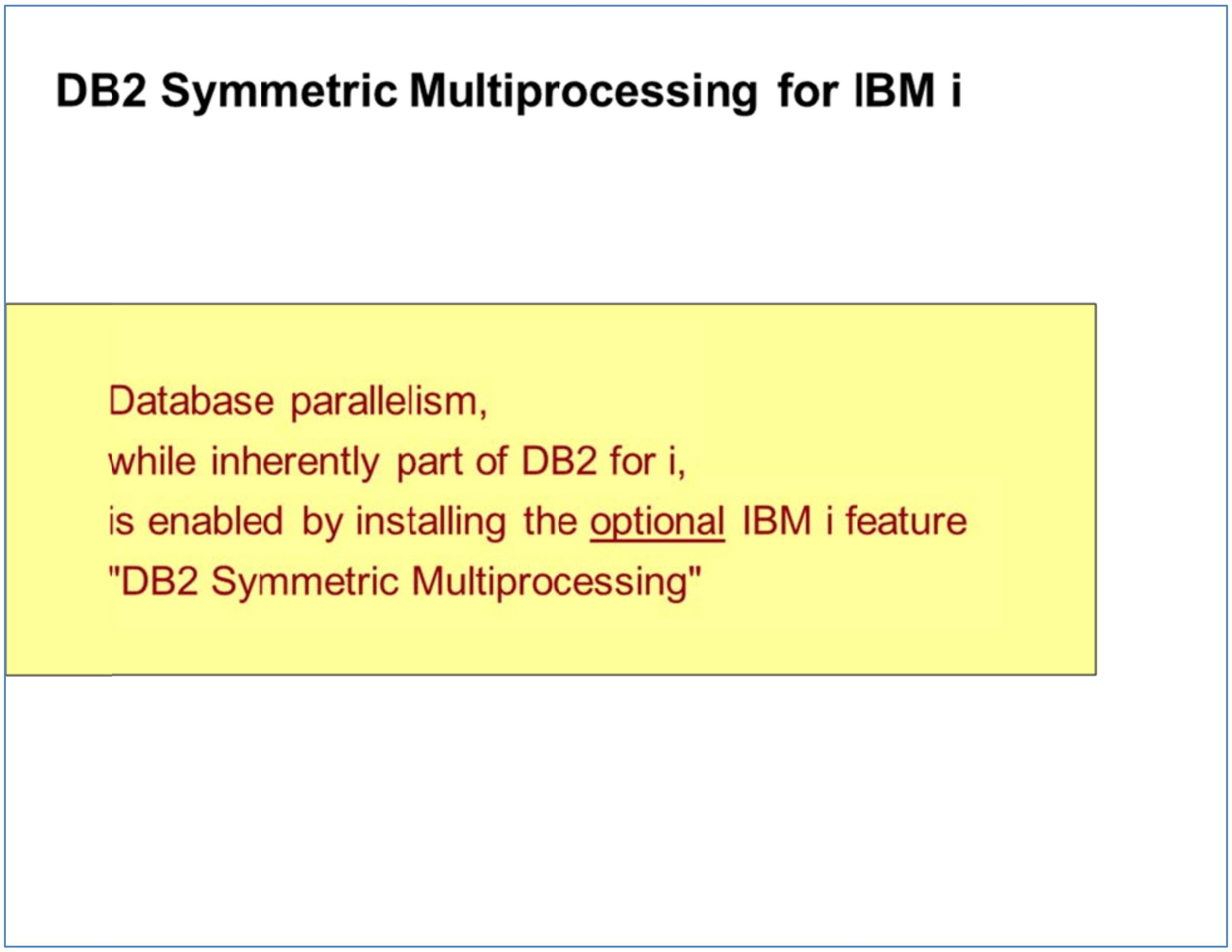
A discussion will follow on the IBM i architecture and how it works cohesively with DB2 for i to deliver powerful parallelism for database access.

Next, you will learn some usage details on how to turn SMP on and off, and how to manage the degree of parallelism that is used by a particular job.

Feedback is important to understanding how well your jobs are monitoring SMP performance, via Visual Explain and other tools. This will be reviewed.

This course will conclude by reviewing several tips and considerations related to the use of database parallelism in the IBM i environment.

It is strongly recommended that database administrators, engineers, analysts, and developers who are new to DB2 for i or using SQL on IBM i, attend the DB2 for i SQL Performance workshop. This class provides in-depth information on the way to architect and implement a high-performing and scalable DB2 for i solution. You can find more information about this workshop: ibm.com/systems/i/db2/db2performance.html



DB2 Symmetric Multiprocessing for IBM i

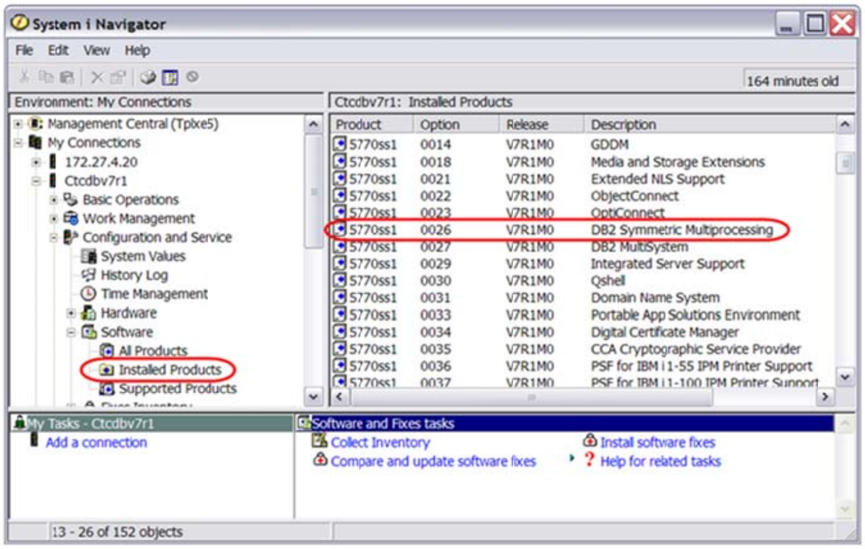
Database parallelism,
while inherently part of DB2 for i,
is enabled by installing the optional IBM i feature
"DB2 Symmetric Multiprocessing"

DB2 SMP must be installed

The DB2 SMP feature is an option of IBM i (a.k.a., i5/OS). This feature—when purchased, installed, and activated—enables DB2 for i to perform database operations in parallel. The database parallelism is based on technology that is inherent both within the operating system and within the database management system. This is one of the advantages of a database management system that is integrated within the operating system.

Symmetric Multiprocessing

System i Navigator, Configuration and Service, Software, Installed Products



Ctcdbv7r1: Installed Products

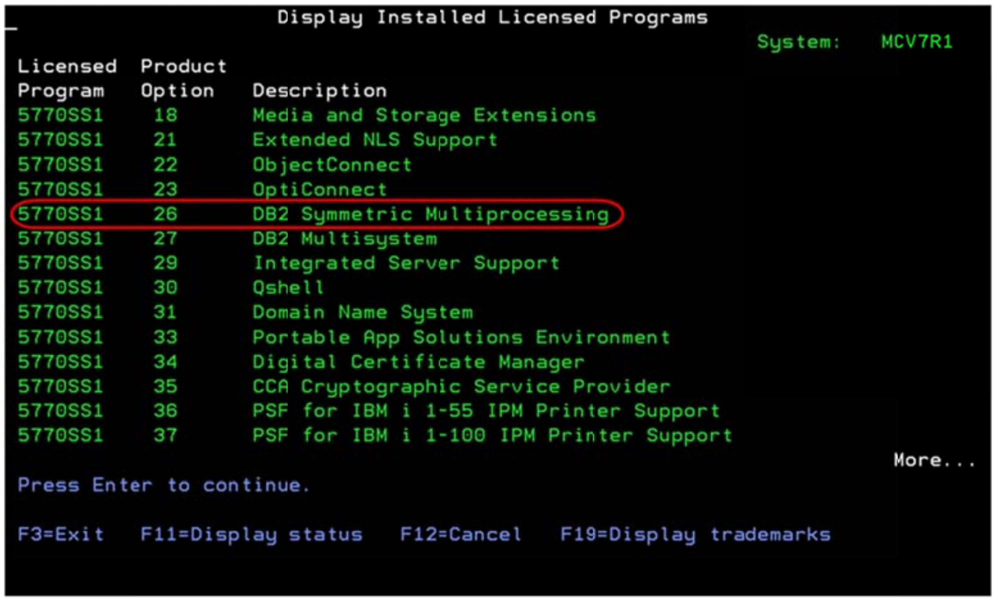
Product	Option	Release	Description
5770ss1	0014	V7R1M0	GDDM
5770ss1	0018	V7R1M0	Media and Storage Extensions
5770ss1	0021	V7R1M0	Extended NLS Support
5770ss1	0022	V7R1M0	ObjectConnect
5770ss1	0023	V7R1M0	OptiConnect
5770ss1	0026	V7R1M0	DB2 Symmetric Multiprocessing
5770ss1	0027	V7R1M0	DB2 MultiSystem
5770ss1	0029	V7R1M0	Integrated Server Support
5770ss1	0030	V7R1M0	Qshell
5770ss1	0031	V7R1M0	Domain Name System
5770ss1	0033	V7R1M0	Portable App Solutions Environment
5770ss1	0034	V7R1M0	Digital Certificate Manager
5770ss1	0035	V7R1M0	CCA Cryptographic Service Provider
5770ss1	0036	V7R1M0	PSF for IBM i 1-55 IPM Printer Support
5770ss1	0037	V7R1M0	PSF for IBM i 1-100 IPM Printer Support

Verify DB2 SMP via System i Navigator

To verify the presence of SMP on an IBM i system or one of its logical partitions, display a list of installed software by using System i Navigator. Drill down the navigation bar as follows: Configuration and Service –> Software –> Installed Products. DB2 Symmetric Multiprocessing is option 26 of IBM i (see red circles in the graphic).

Symmetric Multiprocessing

GO LICPGM, option 10 Display installed licensed programs

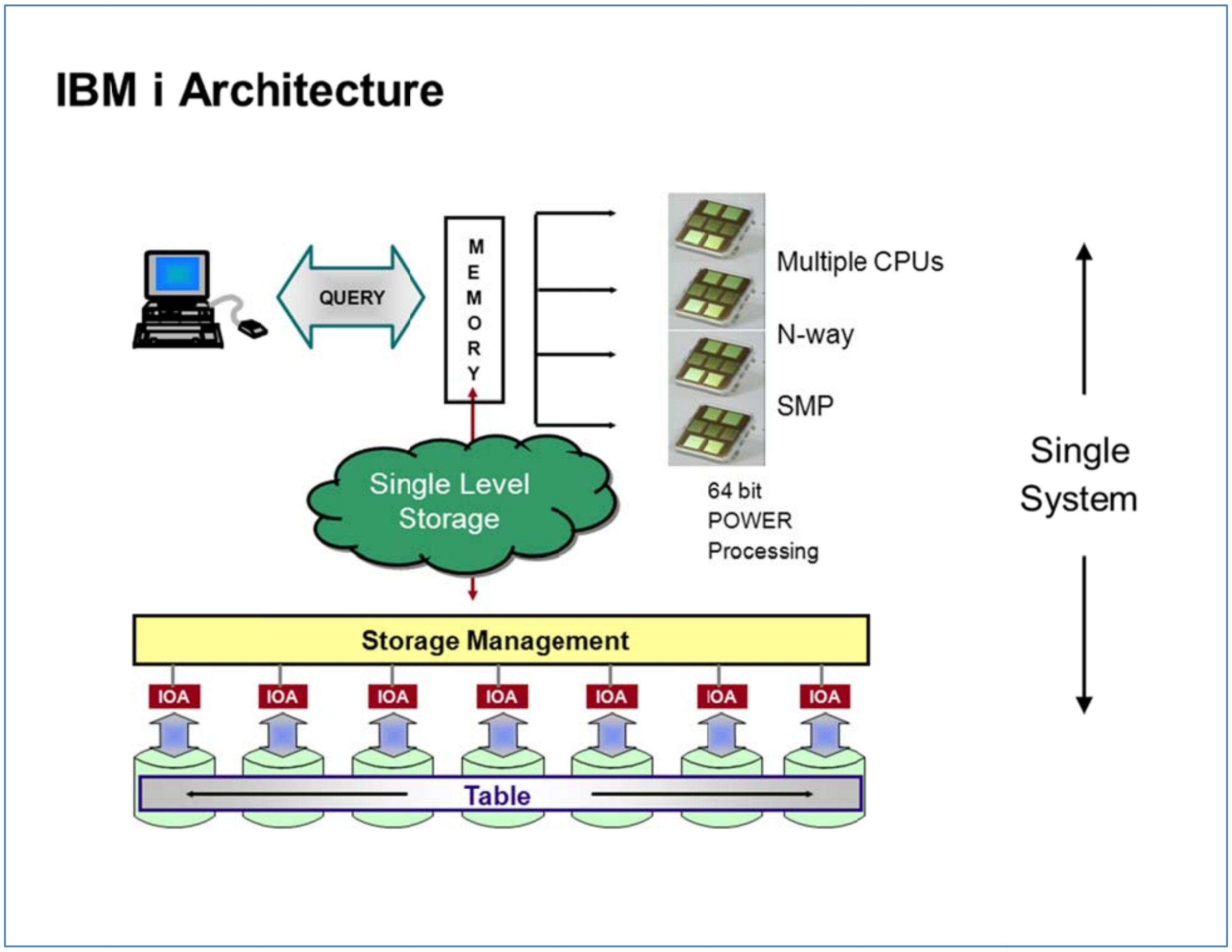


```
                        Display Installed Licensed Programs
                                                          System:    MCV7R1
Licensed  Product
Program   Option    Description
5770SS1    18       Media and Storage Extensions
5770SS1    21       Extended NLS Support
5770SS1    22       ObjectConnect
5770SS1    23       OptiConnect
5770SS1    26       DB2 Symmetric Multiprocessing
5770SS1    27       DB2 Multisystem
5770SS1    29       Integrated Server Support
5770SS1    30       Qshell
5770SS1    31       Domain Name System
5770SS1    33       Portable App Solutions Environment
5770SS1    34       Digital Certificate Manager
5770SS1    35       CCA Cryptographic Service Provider
5770SS1    36       PSF for IBM i 1-55 IPM Printer Support
5770SS1    37       PSF for IBM i 1-100 IPM Printer Support
                                                                     More...
Press Enter to continue.

F3=Exit   F11=Display status   F12=Cancel   F19=Display trademarks
```

Verify DB2 SMP via OS Command Line

To verify the DB2 SMP option from a command line, type GO LICPGM, then select option 10 to “Display installed licensed programs”. Page down through the IBM i options looking for number 26.



IBM i Architecture

The architecture and technology of the POWER system running IBM i provides the foundation for database parallelism. Starting at the bottom of the diagram…

- The independent I/O subsystem, along with IBM i storage management, allows for synchronous and asynchronous database I/O requests. These requests can make use of parallel operations to access data on multiple disk units simultaneously via the I/O processors (IOPs) and/or I/O adapters (IOAs), without using the CPU.

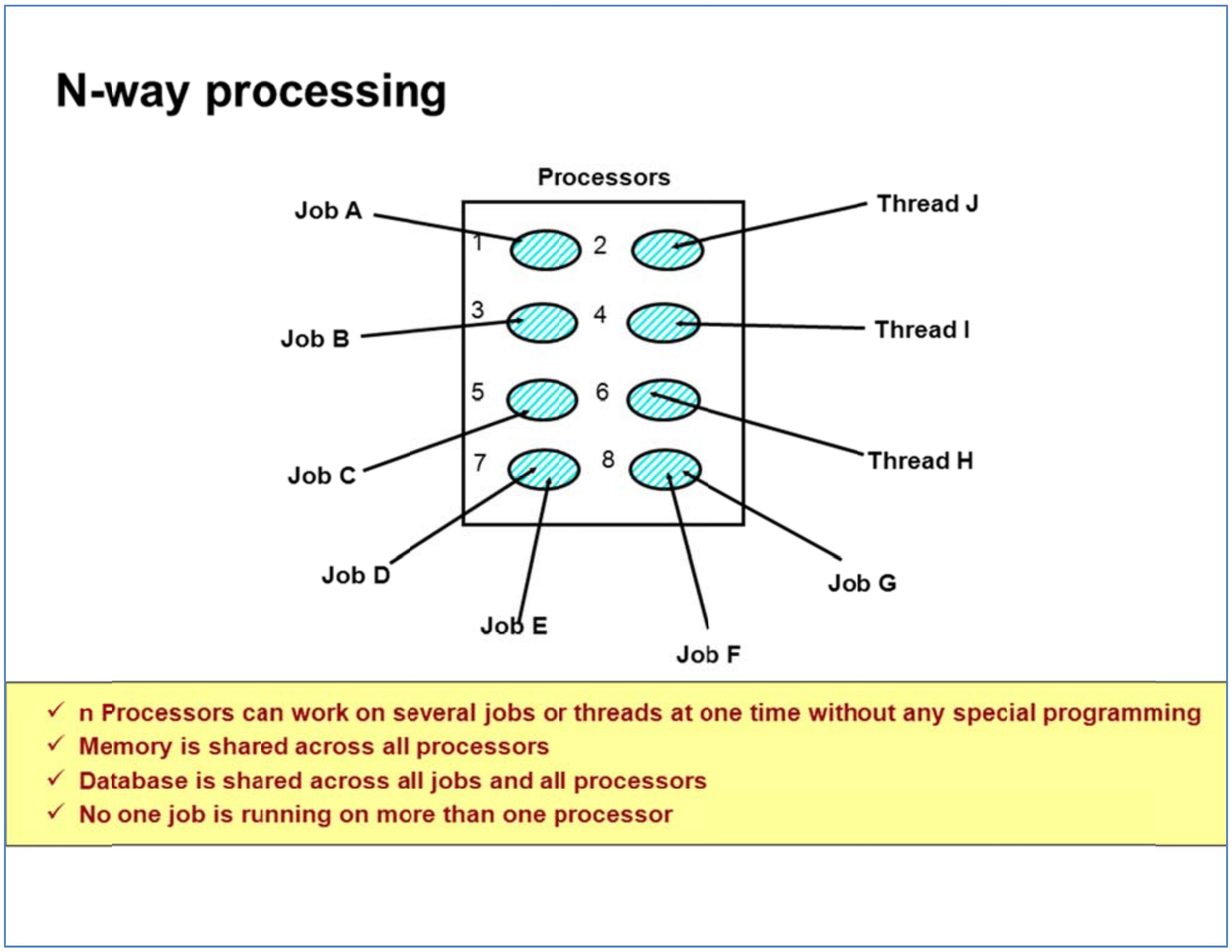
- DB2 for i uses storage management to spread the database objects across all the available disk units. For example, as a table is populated, the space is automatically allocated on the disk units for optimal performance. This spreading of data minimizes contention on any single disk unit, and it also provides the basis for parallel I/O.

- As the disk units are accessed, the data is brought into memory. The design of the POWER hardware supports a very large memory system. As a true 64-bit system, IBM i and DB2 for i can take full advantage of all the available memory. This provides the advantage of using main memory like cache for database objects, or an in-memory database.

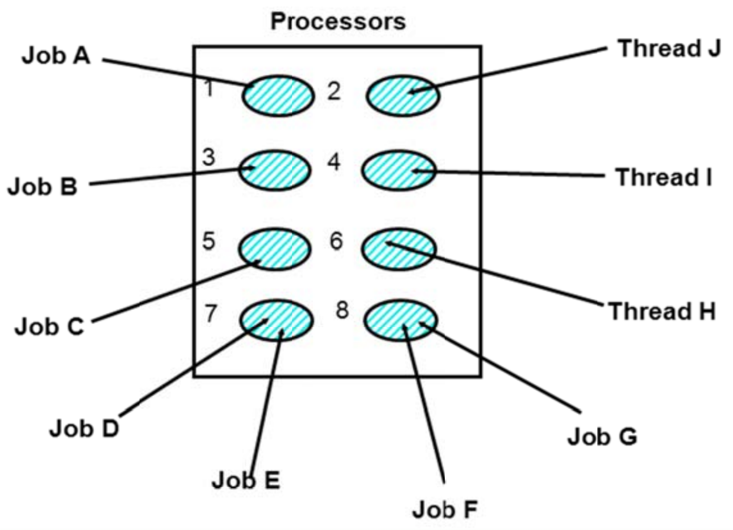
- IBM i systems have a unique way of addressing storage. It views the disk space on the server and the server's main memory as one large storage area. This way of addressing storage is known as single-level storage. The concept of single-level storage means that the knowledge of the underlying characteristics of hardware devices (in this case, main memory storage, solid state disk storage and spinning disk storage) resides in the System Licensed Internal Code (SLIC). All the storage is automatically managed by the system. No user intervention is needed to take full advantage of any storage technology. Programs work with objects; and objects are accessed by name, not by address.

- POWER™ systems support multiple 64-bit CPUs or cores (currently up to 256). IBM i can take advantage of multiple CPUs or cores by automatically dispatching work to one or more CPUs or cores.

As the query request is processed, the integrated database takes advantage of the advanced system and operating system technologies to exploit symmetric multiprocessing, and thus achieve database parallelism.



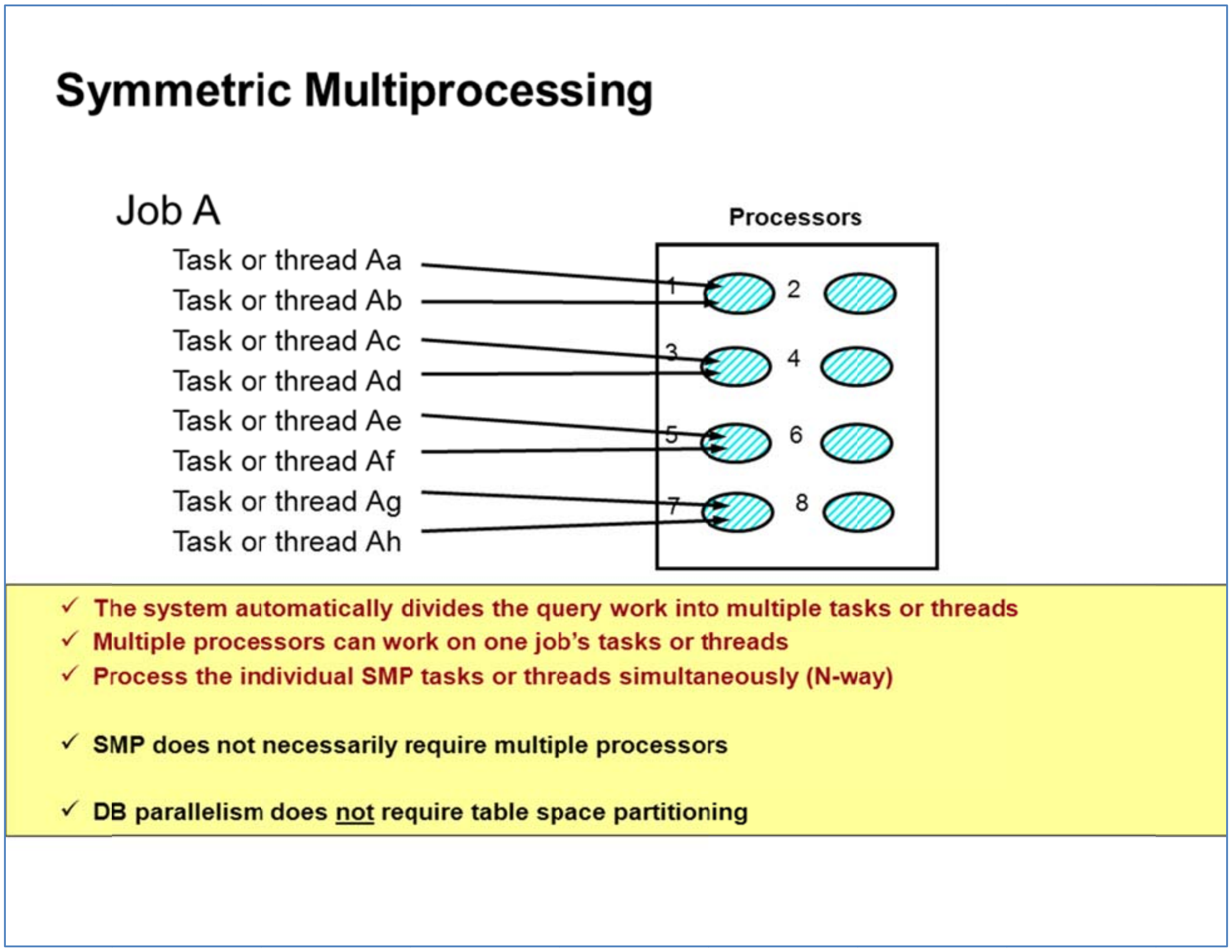
N-way processing



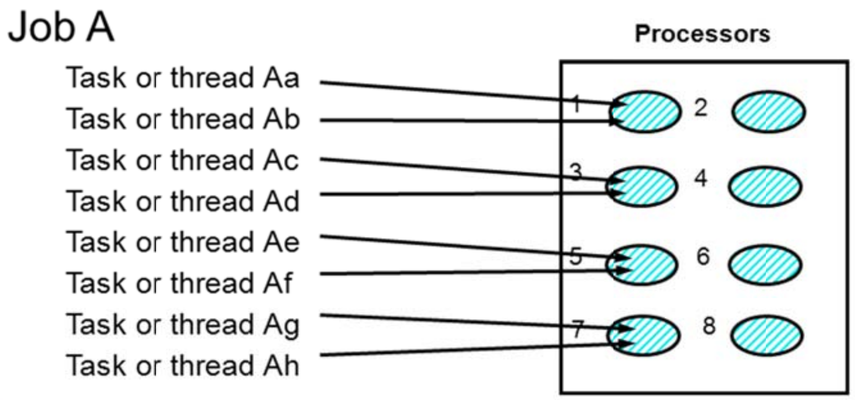
- ✓ n Processors can work on several jobs or threads at one time without any special programming
- ✓ Memory is shared across all processors
- ✓ Database is shared across all jobs and all processors
- ✓ No one job is running on more than one processor

n-way Processing

Within IBM i, a unit of work is defined as a job, thread, or task. Built into the operating system is the ability to dispatch this work to any one of the available CPUs or cores. This concept has the advantage of allowing more requests to be processed, as more CPUs are made available. Any individual job, thread, or task can only run on a single CPU or core. If additional CPUs are available, they provide little or no help. For example, if one job is executing on a server or LPAR with eight cores available, this job will only take advantage of one of the CPUs while the other seven sit idle. To utilize the other cores, additional techniques and strategies must be applied. This is where DB2 SMP comes into play.



Symmetric Multiprocessing



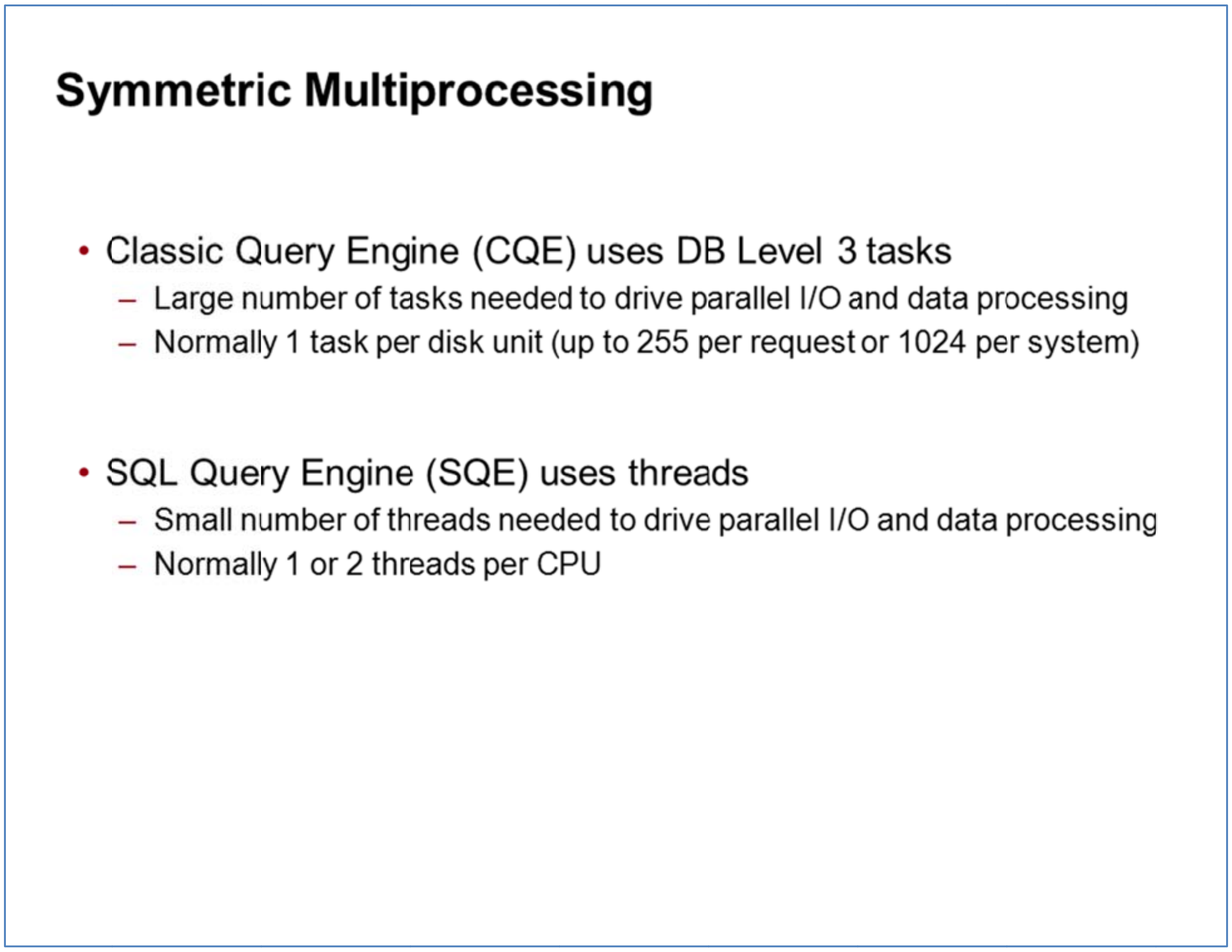
- ✓ **The system automatically divides the query work into multiple tasks or threads**
- ✓ **Multiple processors can work on one job's tasks or threads**
- ✓ **Process the individual SMP tasks or threads simultaneously (N-way)**
- ✓ **SMP does not necessarily require multiple processors**
- ✓ **DB parallelism does not require table space partitioning**

How SMP Works

The DB2 Symmetric Multiprocessing feature provides the optimizer and database engine with additional methods and strategies for retrieving data and processing data in parallel. SMP enables database parallelism on a single server or LPAR where multiple processors (CPU and I/O) that share memory and disk resources work simultaneously toward achieving a single end result. This parallel processing means that the database engine can have more than one (or all) the processors working on a single query simultaneously. The performance of a CPU-bound query can be significantly improved with this feature on multi-core systems by distributing the work load across more than one processor.

While using SMP does not require the presence of more than one CPU or core, database parallelism is most effective when there is more than one physical processor available to run the tasks or threads.

Given that SMP is achieved through the use of the POWER server and the IBM i advanced architecture, table partitioning is *not* required for database parallelism.



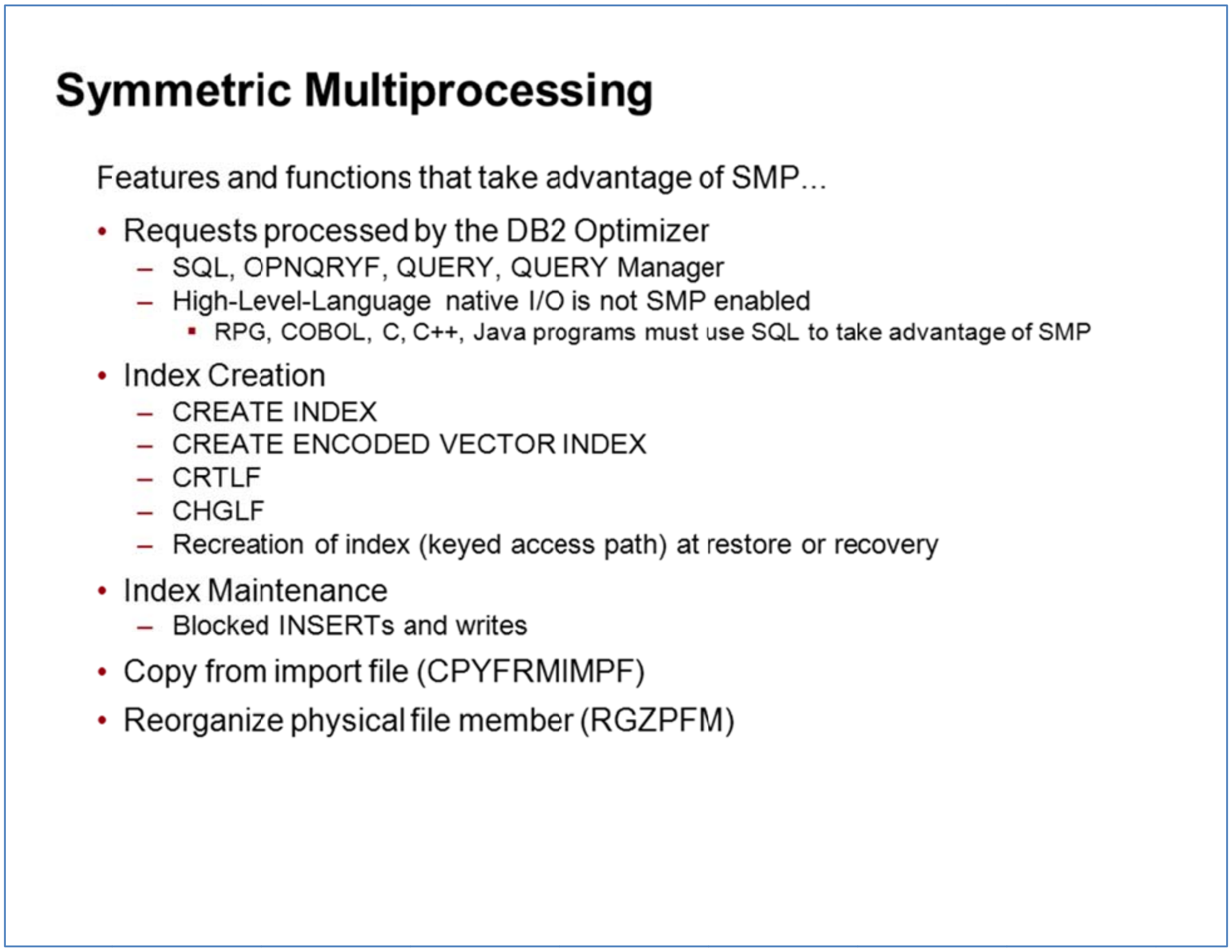
Symmetric Multiprocessing

- Classic Query Engine (CQE) uses DB Level 3 tasks
  - Large number of tasks needed to drive parallel I/O and data processing
  - Normally 1 task per disk unit (up to 255 per request or 1024 per system)
- SQL Query Engine (SQE) uses threads
  - Small number of threads needed to drive parallel I/O and data processing
  - Normally 1 or 2 threads per CPU

CQE versus SQE

Database parallelism is achieved through the use of system licensed internal code (SLIC) tasks or OS threads. The Classic Query Engine (CQE) database engine uses SLIC tasks as a means of achieving parallelism. The SQL Query Engine (SQE) query engine uses OS threads. Generally speaking, CQE limits the number of tasks (for a given request) to the number of disk units that contain the data. In contrast, SQE limits the number threads (for a given request) to the number of physical CPUs or cores available to it.

The row selection and column processing is performed in the respective task or thread. The parent job schedules the work and communicates the requests to the tasks or threads. It also manages and merges the results into the buffer that is returned to the application.


Symmetric Multiprocessing

Features and functions that take advantage of SMP…

- Requests processed by the DB2 Optimizer
  - SQL, OPNQRYF, QUERY, QUERY Manager
  - High-Level-Language native I/O is not SMP enabled
    - RPG, COBOL, C, C++, Java programs must use SQL to take advantage of SMP
- Index Creation
  - CREATE INDEX
  - CREATE ENCODED VECTOR INDEX
  - CRTLF
  - CHGLF
  - Recreation of index (keyed access path) at restore or recovery
- Index Maintenance
  - Blocked INSERTs and writes
- Copy from import file (CPYFRMIMPF)
- Reorganize physical file member (RGZPFM)

Functions Parallel-enabled by SMP

By installing and enabling the DB2 SMP feature, the query optimizer is able to consider parallel methods and strategies. However, these parallel methods and strategies are only available for certain database features and functions. Generally speaking, any query request that is processed by the DB2 optimizer is eligible—regardless of programming interface. A static or dynamic SQL request from within a high-level language program is eligible to use SMP. In contrast, native, record level access from within high-level language programs is not enabled for SMP. For example, RPG READ or CHAIN operations are not parallel enabled. This is one reason why using SQL DML to access data is advantageous.

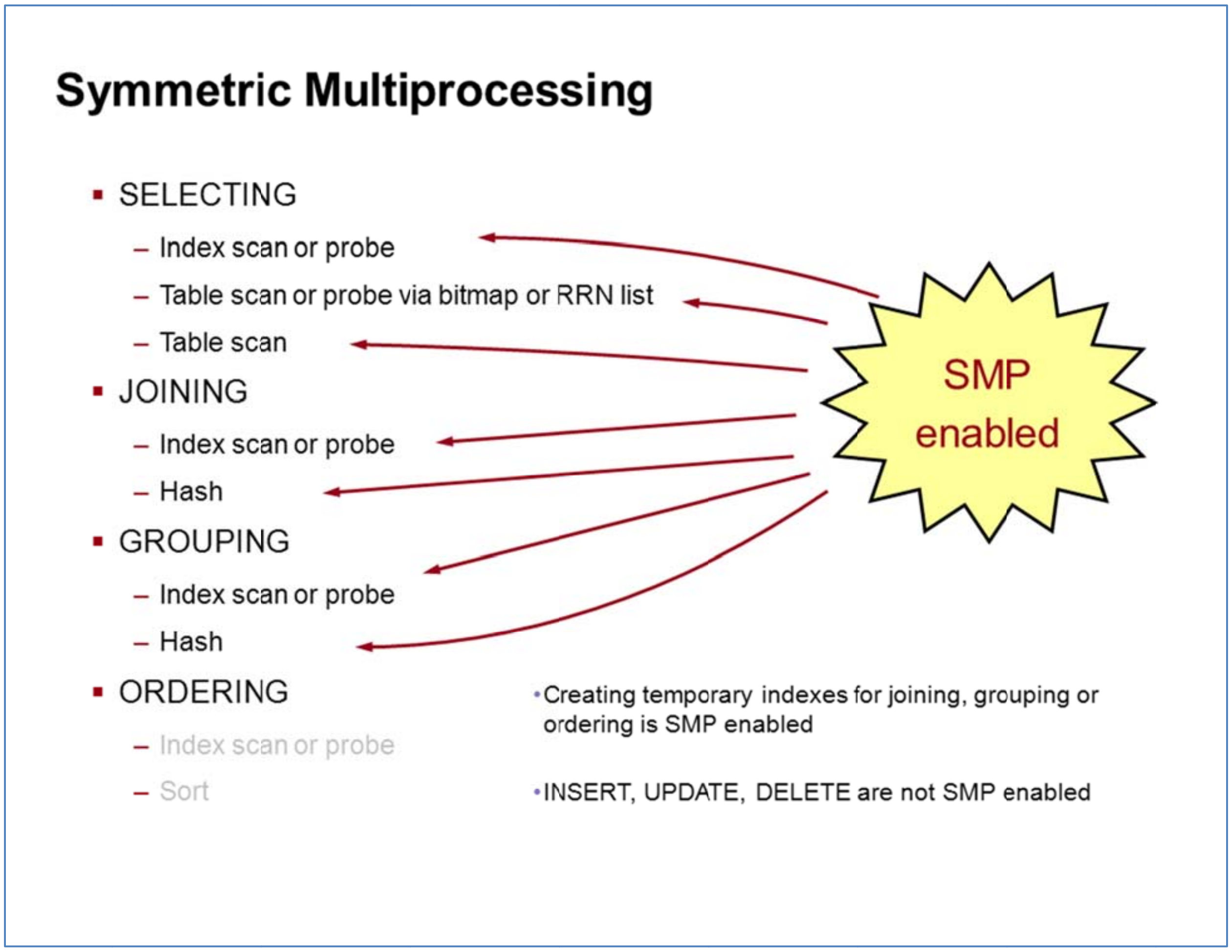
Another important use of SMP is to speed up the creation and maintenance of DB2 indexes (sometimes referred to as “keyed access paths”). Creating indexes with SMP allows the database engine to use all of the available resources to speed up the creation process. This may be an important strategy when an application is unavailable until an index for one of the files accessed by that application is available.

With multiple indexes or keyed logical files over a table or physical file, index maintenance occurs whenever rows are added, changed, or deleted. Normally, each index is maintained synchronously—that is, one at a time. With SMP enabled, blocked INSERT or WRITE operations can benefit from the fact that the database engine maintains each index in parallel. This is accomplished by using one database task to maintain each index – again trading the use of additional resources for a decrease in time. This has the benefit of reducing the overall index maintenance time and the overall INSERT time.

It should be noted, the IBM OmniFind Text Search Server for DB2 for i (5733-OMF) indexes are *not* affected by DB2 SMP. In other words, the text indexes are not created in parallel nor maintained in parallel via the SMP feature.

Within IBM i, DB2 has the ability to import fixed format or stream (CSV) data into a table. This function is known as *copy from import file*, and is invoked using the command CPYFRMIMPF. This feature is parallel-enabled via SMP. Note that only when copying data from a fixed format file is SMP available. Copying data from a stream file is *not* done in parallel via the SMP feature. CPYTOIMPF is also *not* parallel enabled.

Within IBM i, DB2 has the ability to physically reorganize the rows within a table. This function is known as *reorganize physical file member*, and is invoked using the System i Navigator (right click on a table) or the OS command RGZPFM. This feature is parallel-enabled via SMP.



Symmetric Multiprocessing

- SELECTING
  - Index scan or probe
  - Table scan or probe via bitmap or RRN list
  - Table scan
- JOINING
  - Index scan or probe
  - Hash
- GROUPING
  - Index scan or probe
  - Hash
- ORDERING
  - Index scan or probe
  - Sort

- Creating temporary indexes for joining, grouping or ordering is SMP enabled
- INSERT, UPDATE, DELETE are not SMP enabled

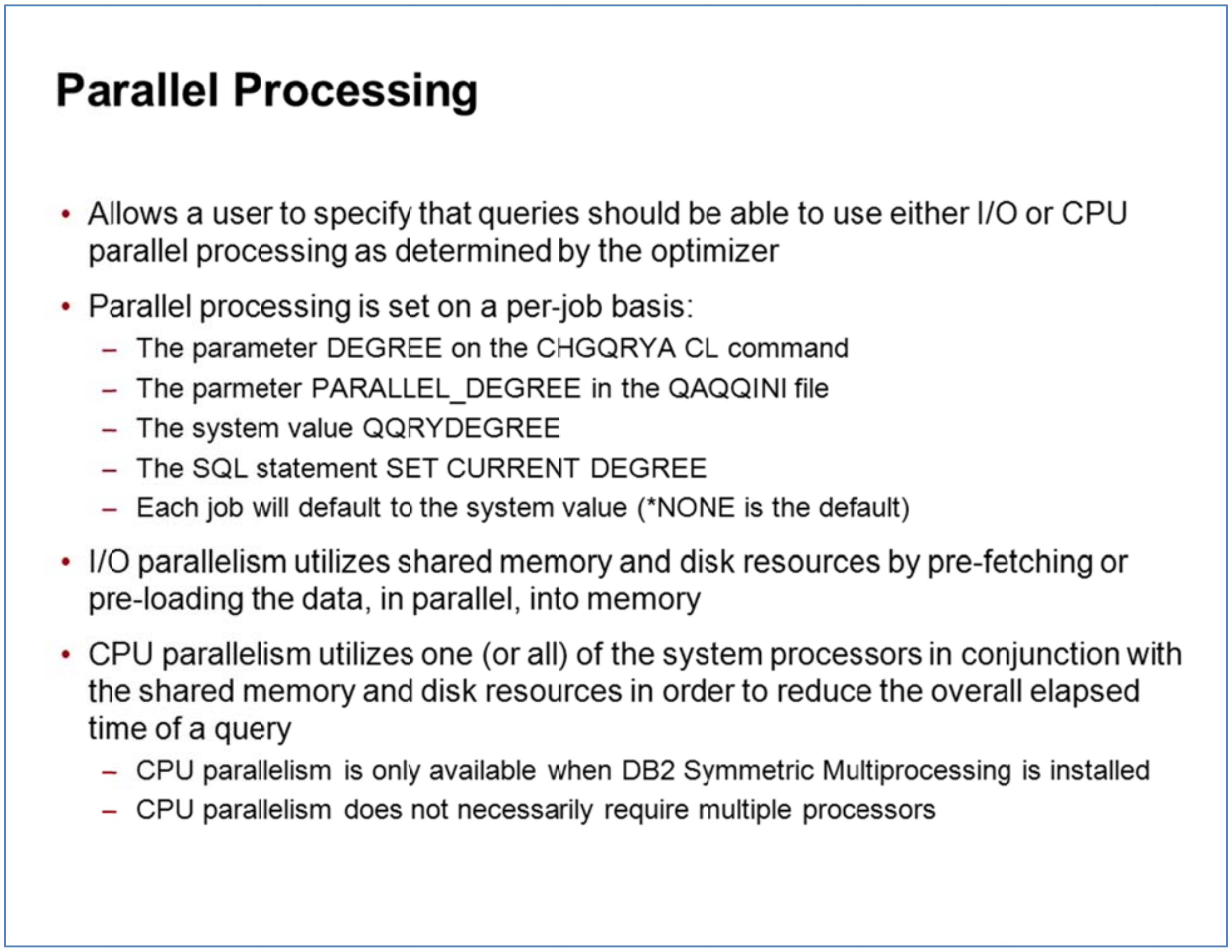
SQL Requests and SMP

While INSERT, UPDATE and DELETE operations cannot execute in parallel, most query methods are parallel enabled via DB2 SMP, This means that if the query is using a table scan, that particular operation can use parallelism and benefit from SMP. In other words, the table is scanned and the rows tested in parallel.

Some operations, such as ordering the data via a sort, are not performed in parallel and do not benefit from SMP.

If a temporary index is created during the query request, the operation can benefit from SMP.

As database operations are analyzed and tuned for performance, it is important to remember which statements, strategies and methods are parallel-enabled and potentially benefit from SMP.



Parallel Processing

- Allows a user to specify that queries should be able to use either I/O or CPU parallel processing as determined by the optimizer
- Parallel processing is set on a per-job basis:
  - The parameter DEGREE on the CHGQRYA CL command
  - The parmeter PARALLEL_DEGREE in the QAQQINI file
  - The system value QQRYDEGREE
  - The SQL statement SET CURRENT DEGREE
  - Each job will default to the system value (*NONE is the default)
- I/O parallelism utilizes shared memory and disk resources by pre-fetching or pre-loading the data, in parallel, into memory
- CPU parallelism utilizes one (or all) of the system processors in conjunction with the shared memory and disk resources in order to reduce the overall elapsed time of a query
  - CPU parallelism is only available when DB2 Symmetric Multiprocessing is installed
  - CPU parallelism does not necessarily require multiple processors

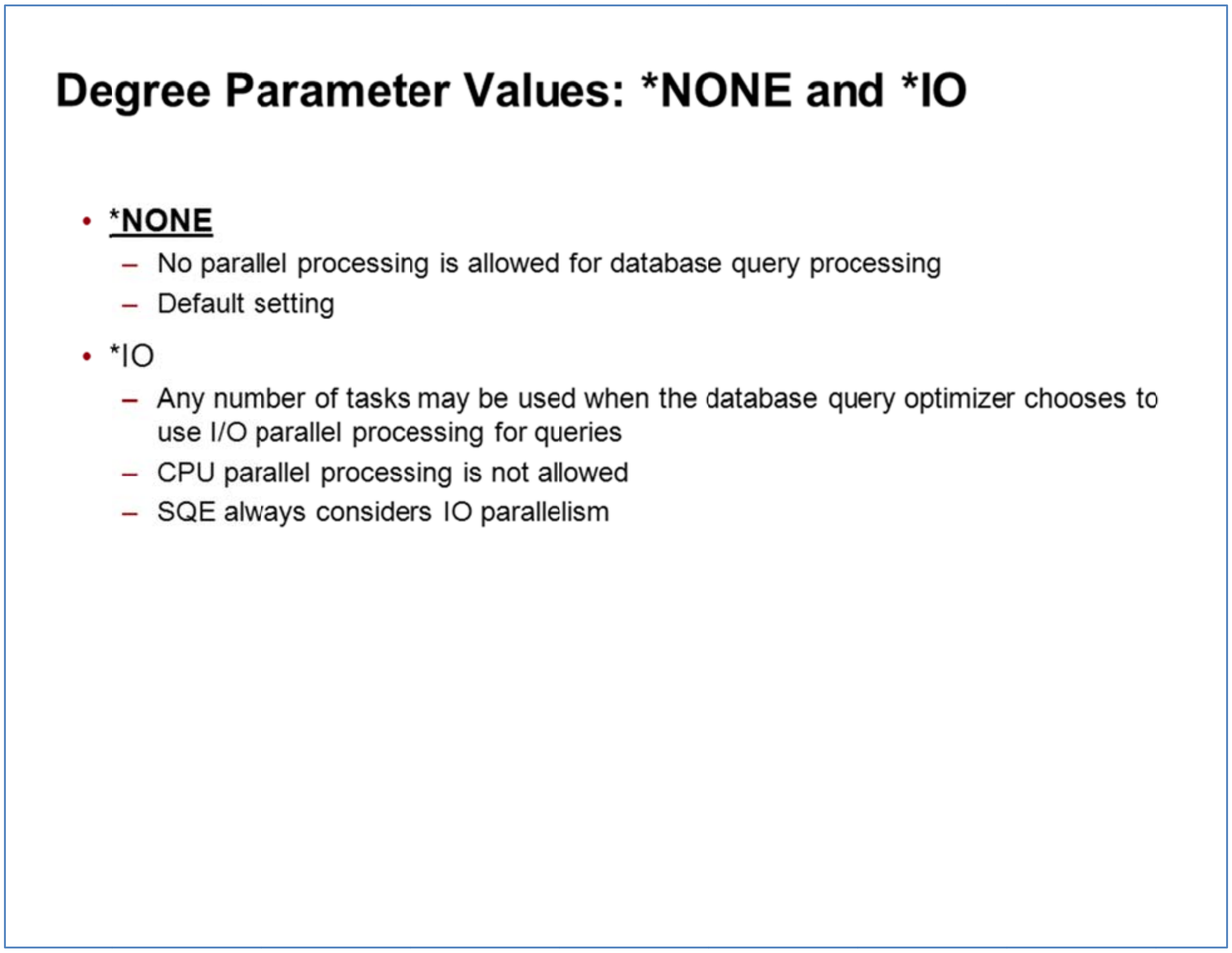
Parallel Database Processing

As already mentioned early, the use of database parallelism requires the SMP feature to be installed on the system (or LPAR). However, SMP must also be enabled for the job processing the request.

The default setting for DB2 is to allow no database parallelism. The parallel processing degree can be set via the system value QQRYDEGREE, a query option file (QAQQINI), the DEGREE parameter on the *change query attributes* (CHGQRYA) command, or the SQL statement SET CURRENT DEGREE. In fact, the CHGQRYA command or SET statement can be issued multiple times within a job to toggle parallelism on and off.

There are two forms of database parallelism: I/O parallelism and CPU parallelism. I/O parallelism allows for the accessing of data in parallel, but the processing of that data does not occur in parallel. CPU parallelism allows for both the accessing of data and the processing of that data in parallel. For example, a table scan can use parallel I/O to access and bring the rows into memory, but only process rows with one task. With the SMP feature installed and enabled, CPU parallelism can be used to both access the data in parallel and to process the rows in parallel, using multiple tasks or threads.

Setting the parallel degree allows the query optimizer to *consider* the use of SMP. The optimizer determines whether or not the query will benefit from parallel methods, and builds the appropriate strategy for using database parallelism within the query plan.

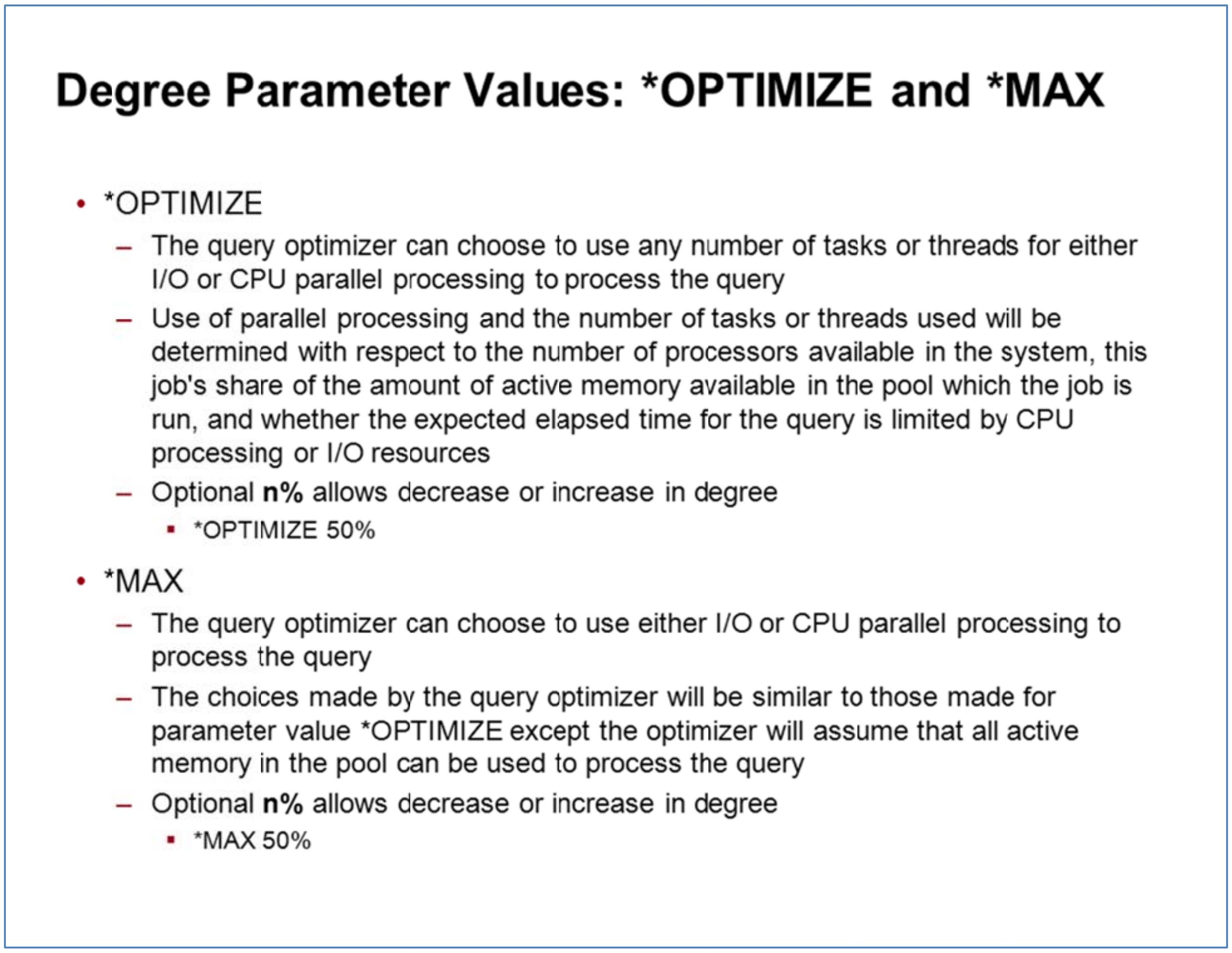

Degree Parameter Values: *NONE and *IO

- ***NONE**
  - No parallel processing is allowed for database query processing
  - Default setting
- *IO
  - Any number of tasks may be used when the database query optimizer chooses to use I/O parallel processing for queries
  - CPU parallel processing is not allowed
  - SQE always considers IO parallelism

Degree Parameter Values

***NONE** – This is the default value system wide. It stipulates that DB2 for i and the query optimizer cannot use any parallel methods, neither CPU nor I/O parallelism. Although, SQE can and will perform parallel I/O regardless of this setting.

***IO** – This value indicates that the CQE optimizer can use parallel I/O methods only. This setting does not require the SMP feature since the ability to use parallel I/O is intrinsically part of IBM i. SQE considers I/O parallelism with or without this setting.


Degree Parameter Values: *OPTIMIZE and *MAX

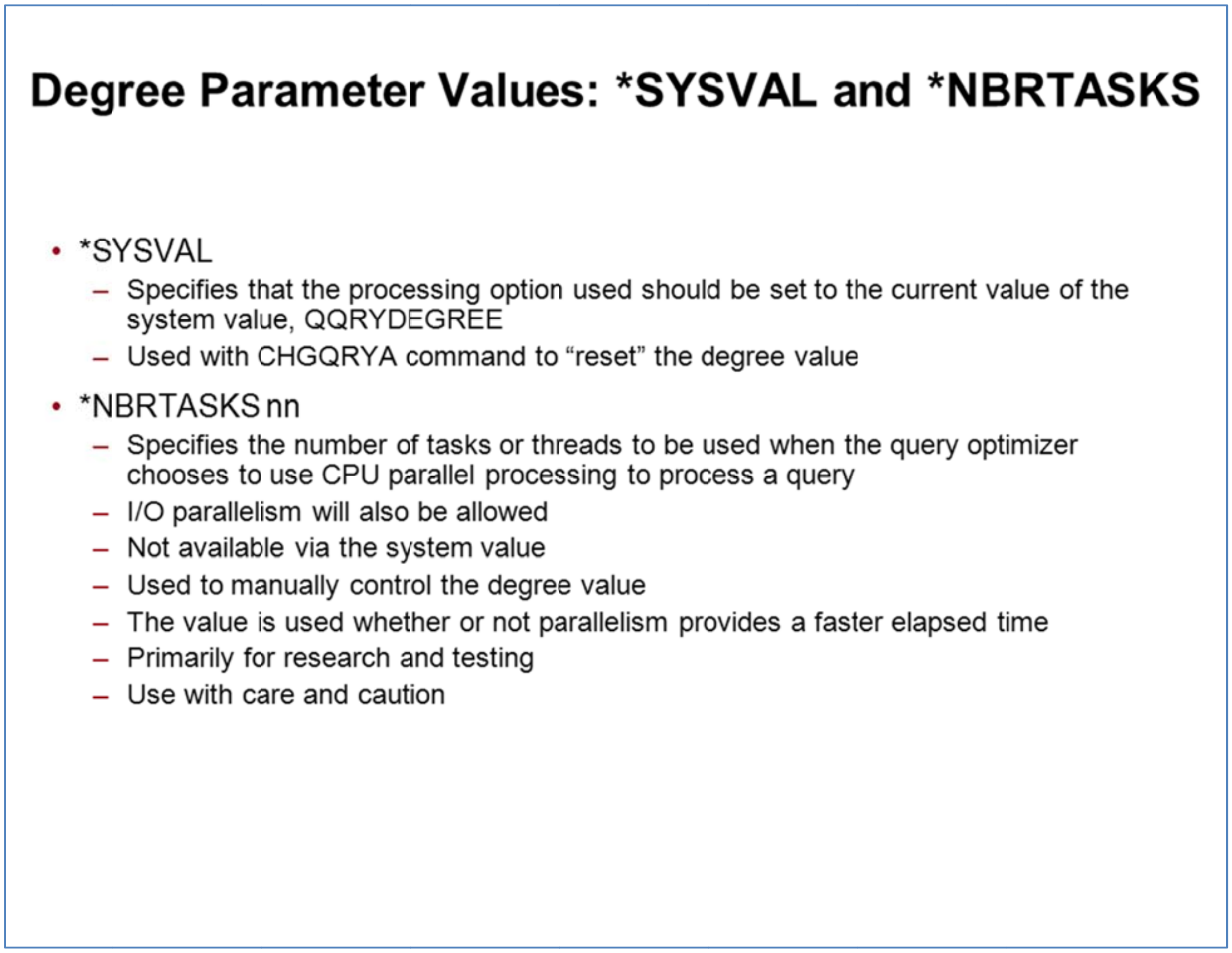
- *OPTIMIZE
  - The query optimizer can choose to use any number of tasks or threads for either I/O or CPU parallel processing to process the query
  - Use of parallel processing and the number of tasks or threads used will be determined with respect to the number of processors available in the system, this job's share of the amount of active memory available in the pool which the job is run, and whether the expected elapsed time for the query is limited by CPU processing or I/O resources
  - Optional **n%** allows decrease or increase in degree
    - *OPTIMIZE 50%
- *MAX
  - The query optimizer can choose to use either I/O or CPU parallel processing to process the query
  - The choices made by the query optimizer will be similar to those made for parameter value *OPTIMIZE except the optimizer will assume that all active memory in the pool can be used to process the query
  - Optional **n%** allows decrease or increase in degree
    - *MAX 50%

Degree Parameter Values

***OPTIMIZE** – This value specifies that the optimizer can use both parallel I/O and parallel CPU methods. (The use of parallel CPU methods requires the SMP feature.) The optimizer will build a plan to use a good share of the computing resources, but not all of the resources. Think of this as a “good neighbor policy.”

***MAX** – This value states that the optimizer can use both parallel I/O and parallel CPU methods. (The use of parallel CPU methods requires the SMP feature.) The optimizer will build a plan to use *all* or a good share of the computing resources. Think of this as a “bad neighbor policy.” Or better yet, a way to allow a much larger share of resources to be used for a given DB job or request.

Once the query optimizer has determined the parallel degree, the amount can be dialed up or down based on a percentage. Specifying *OPTIMIZE 50% tells DB2 to reduce the parallel degree determined at optimization time by 50% - a degree of 8 becomes a degree of 4. Specifying *OPTIMIZE 150% tells DB2 ro increase the parallel degree determined at optimization time by 150% - a degree of 8 becomes a degree of 12. This way the optimizer is still responsible for determining the initial parallel degree value, but the database engineer can adjust the value based on environmental considerations or criteria unknown to the optimizer.



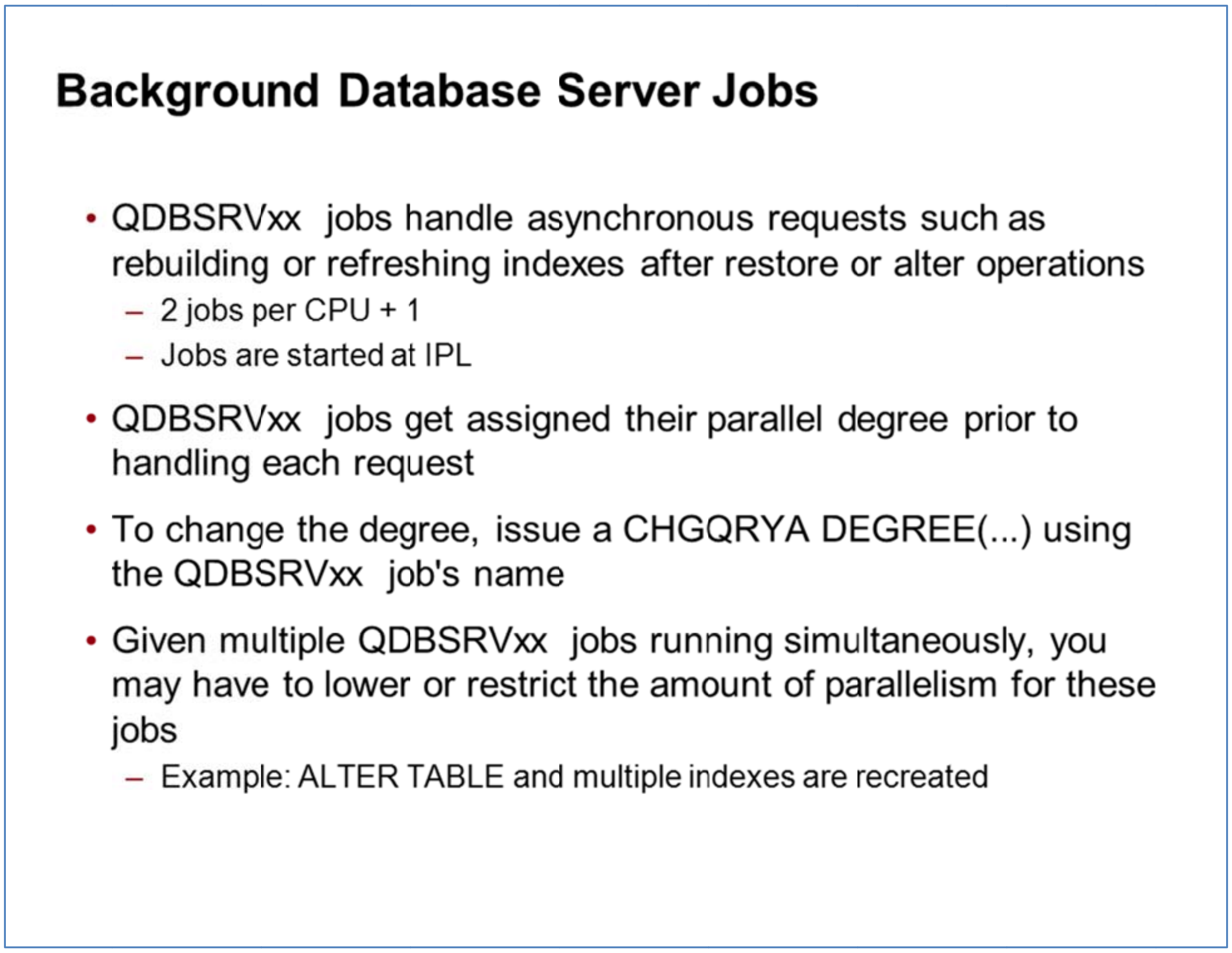
Degree Parameter Values: *SYSVAL and *NBRTASKS

- *SYSVAL
  - Specifies that the processing option used should be set to the current value of the system value, QQRYDEGREE
  - Used with CHGQRYA command to "reset" the degree value
- *NBRTASKS nn
  - Specifies the number of tasks or threads to be used when the query optimizer chooses to use CPU parallel processing to process a query
  - I/O parallelism will also be allowed
  - Not available via the system value
  - Used to manually control the degree value
  - The value is used whether or not parallelism provides a faster elapsed time
  - Primarily for research and testing
  - Use with care and caution

Degree Parameter Values

***SYSVAL** – This value is used on the CHGQRYA command to "reset" the job's query degree to the system value without knowing what the system value actually is.

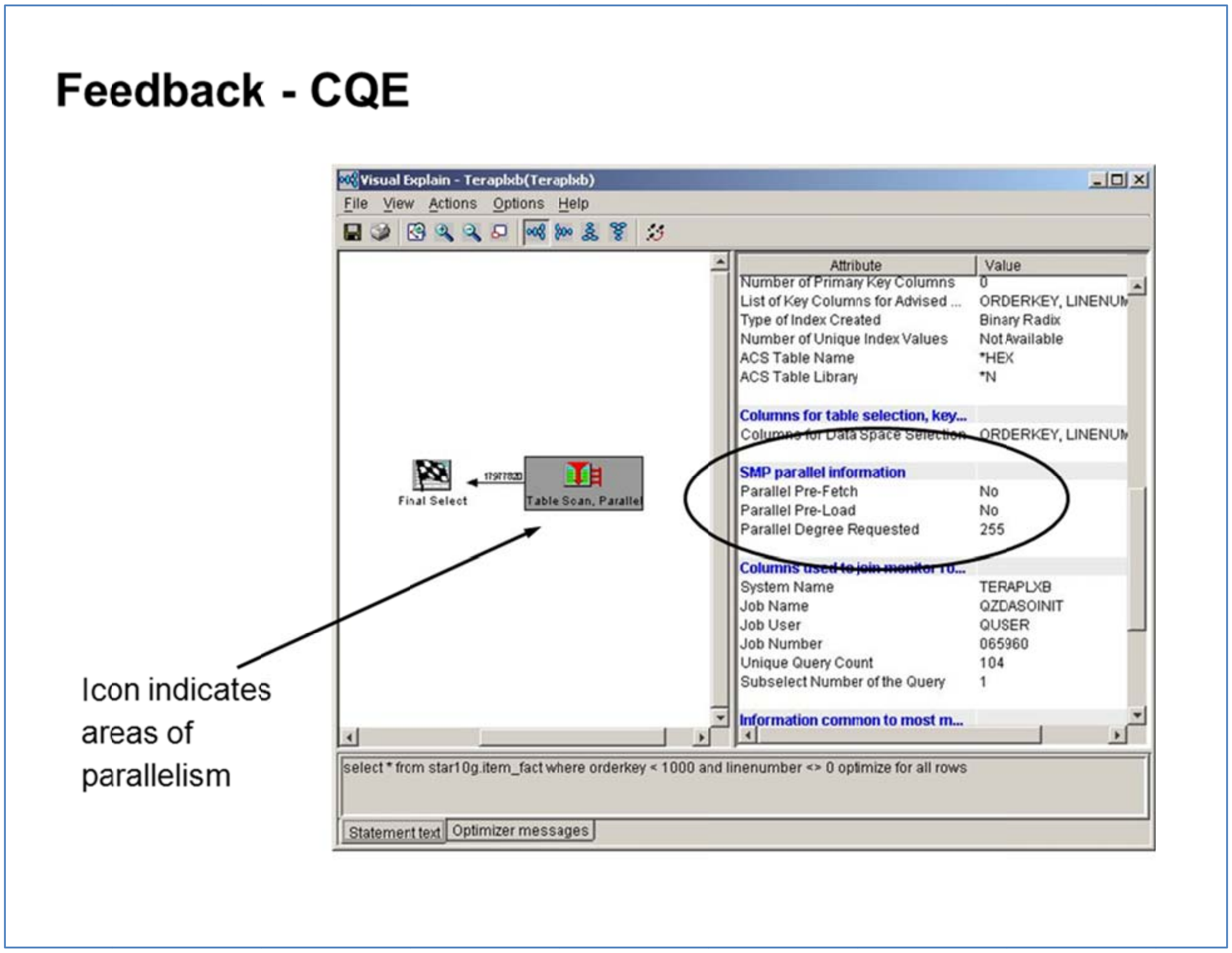
***NBRTASKS nn** – This value indicates that the optimizer can use both parallel I/O and parallel CPU methods The use of parallel CPU methods requires the SMP feature.) The optimizer will build a plan to use the number of task or threads requested. This value is used for experimentation and testing, and is not necessarily designed for use in a production environment given that it forces. (the optimizer to use parallelism. AVOID THIS VALUE OR USE WITH EXTREME CAUTION.


Background Database Server Jobs

- QDBSRVxx jobs handle asynchronous requests such as rebuilding or refreshing indexes after restore or alter operations
  - 2 jobs per CPU + 1
  - Jobs are started at IPL
- QDBSRVxx jobs get assigned their parallel degree prior to handling each request
- To change the degree, issue a CHGQRYA DEGREE(...) using the QDBSRVxx job's name
- Given multiple QDBSRVxx jobs running simultaneously, you may have to lower or restrict the amount of parallelism for these jobs
  - Example: ALTER TABLE and multiple indexes are recreated

Background Database Server Jobs

Some database requests are executed within the IBM i background database server jobs, and are recognized on the job list by the name “QDBSRVxx.” These jobs pick up the parallel degree value from the system value QQRYDEGREE prior to handling a request. To change the degree value to something other than the system value, use the CHGQRYA command and specify the particular QDBSRVxx job whose degree value you wish to alter.



Feedback - CQE

Attribute	Value
Number of Primary Key Columns	0
List of Key Columns for Advised ...	ORDERKEY, LINENUM
Type of Index Created	Binary Radix
Number of Unique Index Values	Not Available
ACS Table Name	*HEX
ACS Table Library	*N
Columns for table selection, key...	
Columns for Data Space Selection	ORDERKEY, LINENUM
SMP parallel information	
Parallel Pre-Fetch	No
Parallel Pre-Load	No
Parallel Degree Requested	255
Columns used to join monitor ro...	
System Name	TERAPLXB
Job Name	QZDASOINIT
Job User	QUSER
Job Number	065960
Unique Query Count	104
Subselect Number of the Query	1
Information common to most m...	

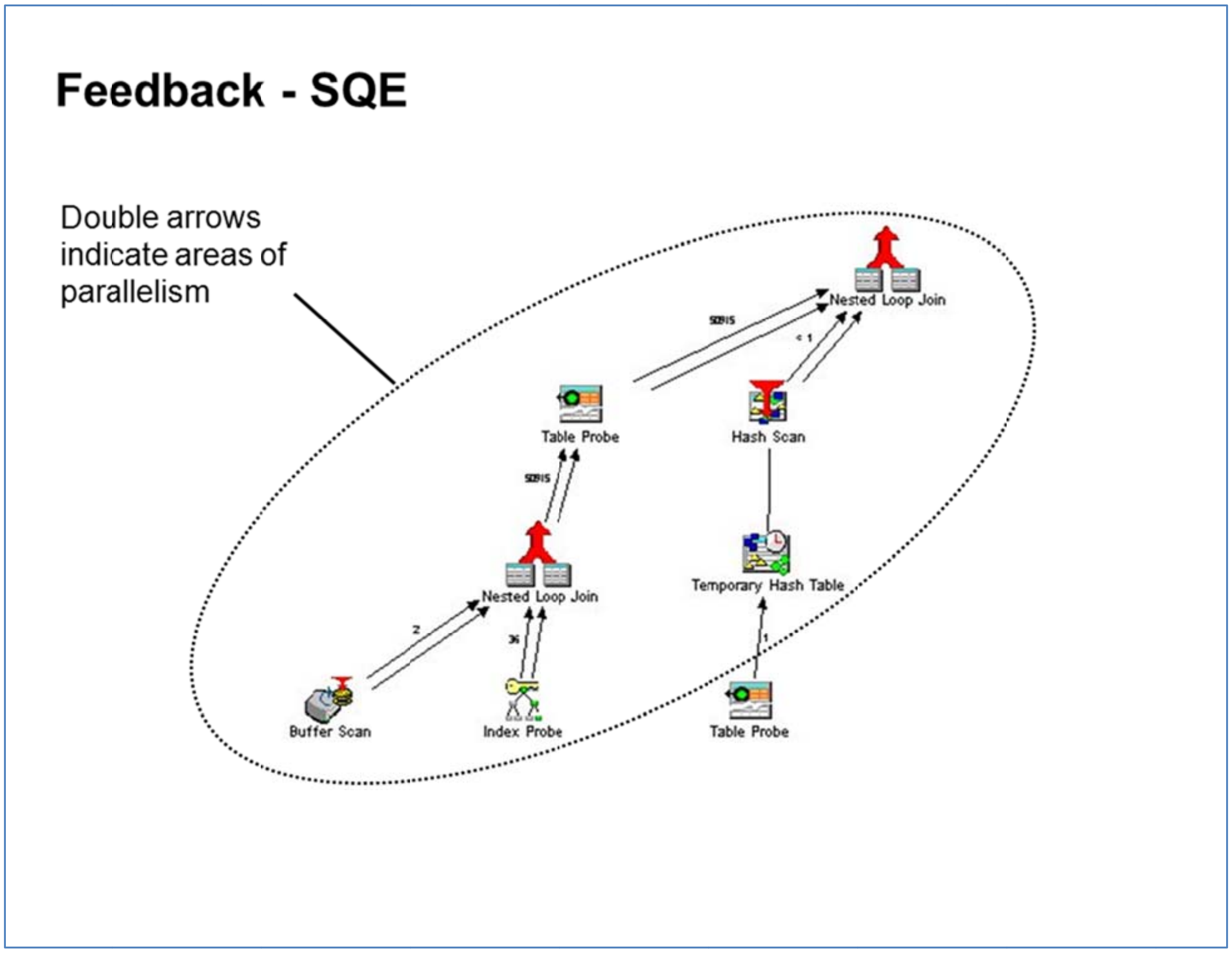
```
select * from star10g.item_fact where orderkey < 1000 and linenumber <> 0 optimize for all rows
```

Icon indicates areas of parallelism

Visual Explain Feedback – CQE

To assist with verifying the use of parallel methods, query plan information is provided. One of the primary feedback mechanisms is Visual Explain, which is part of IBM i Access for Windows - System i Navigator - Database. Visual Explain renders the query plan as a picture or graph.

CQE depicts parallel database methods using specific icons within the query graph (shown on the left in this screen capture), and as textual feedback to the right of the query graph when the respective icon is selected.

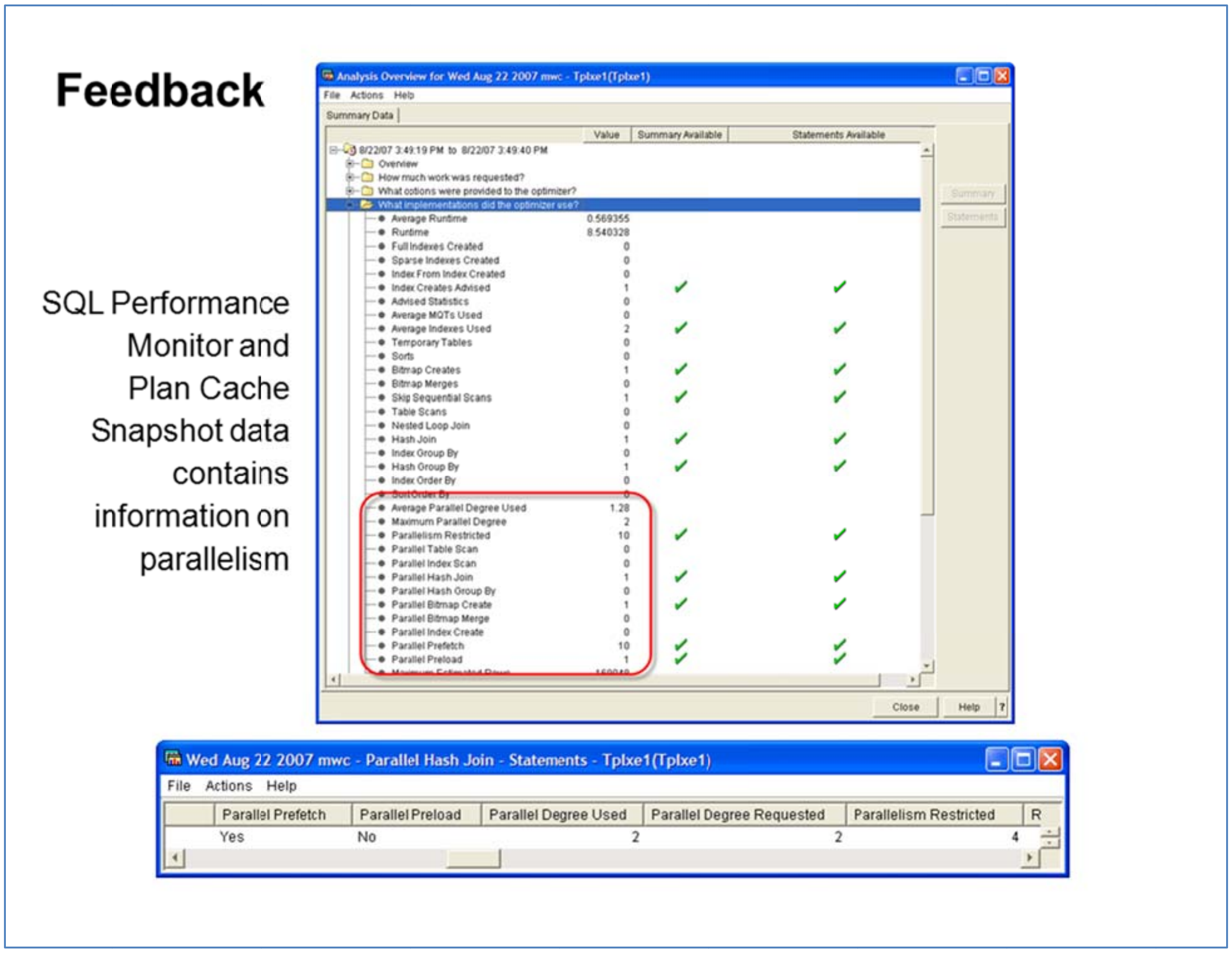


Feedback - SQE

Double arrows indicate areas of parallelism

Visual Explain Feedback – SQE

SQE depicts database parallelism by using double arrows – regardless of the parallel degree used. In these areas of the plan, parallel strategies and methods are used to decrease the run time of the query. The parallel capabilities of SQE are extensive and not just limited to a particular method.



Feedback

SQL Performance Monitor and Plan Cache Snapshot data contains information on parallelism

8/22/07 3:49:19 PM to 8/22/07 3:49:40 PM
- Overview
- How much work was requested?
- What options were provided to the optimizer?
- What implementations did the optimizer use?

	Value	Summary Available	Statements Available
Average Runtime	0.569355		
Runtime	8.540328		
Full Indexes Created	0		
Sparse Indexes Created	0		
Index From Index Created	0		
Index Creates Advised	1	✓	✓
Advised Statistics	0		
Average MQTs Used	0		
Average Indexes Used	2	✓	✓
Temporary Tables	0		
Sorts	0		
Bitmap Creates	1	✓	✓
Bitmap Merges	0		
Skip Sequential Scans	1	✓	✓
Table Scans	0		
Nested Loop Join	0		
Hash Join	1	✓	✓
Index Group By	0		
Hash Group By	1	✓	✓
Index Order By	0		
Sort Order By	0		
Average Parallel Degree Used	1.28		
Maximum Parallel Degree	2		
Parallelism Restricted	10	✓	✓
Parallel Table Scan	0		
Parallel Index Scan	0		
Parallel Hash Join	1	✓	✓
Parallel Hash Group By	0		
Parallel Bitmap Create	1	✓	✓
Parallel Bitmap Merge	0		
Parallel Index Create	0		
Parallel Prefetch	10	✓	✓
Parallel Preload	1	✓	✓

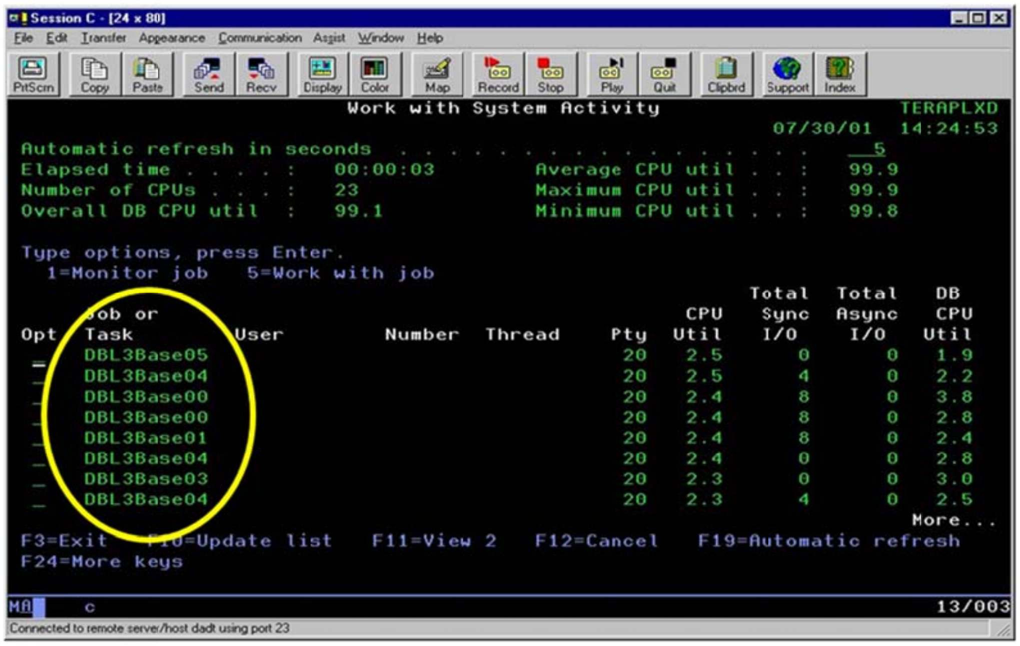
Wed Aug 22 2007 mwc - Parallel Hash Join - Statements - Tplxe1(Tplxe1)
File Actions Help

	Parallel Prefetch	Parallel Preload	Parallel Degree Used	Parallel Degree Requested	Parallelism Restricted	R
	Yes	No	2	2	4	

Textual Feedback

Another mechanism for providing optimizer information and query plan feedback is the SQL Performance Monitor (aka the detailed Database Monitor). The Monitor captures and provides data that indicates the use of parallel methods and strategies within a query. This data can be displayed using either handwritten queries or the reports provided by the DB2 Performance Center within System i Navigator. Right click on an SQL Performance Monitor or SQL Plan Cache Snapshot and choose Analyze.

Work Management Feedback - WRKSYSACT


Session C - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
PrtScrn Copy Paste Send Recv Display Color Map Record Stop Play Quit Clipbrd Support Index

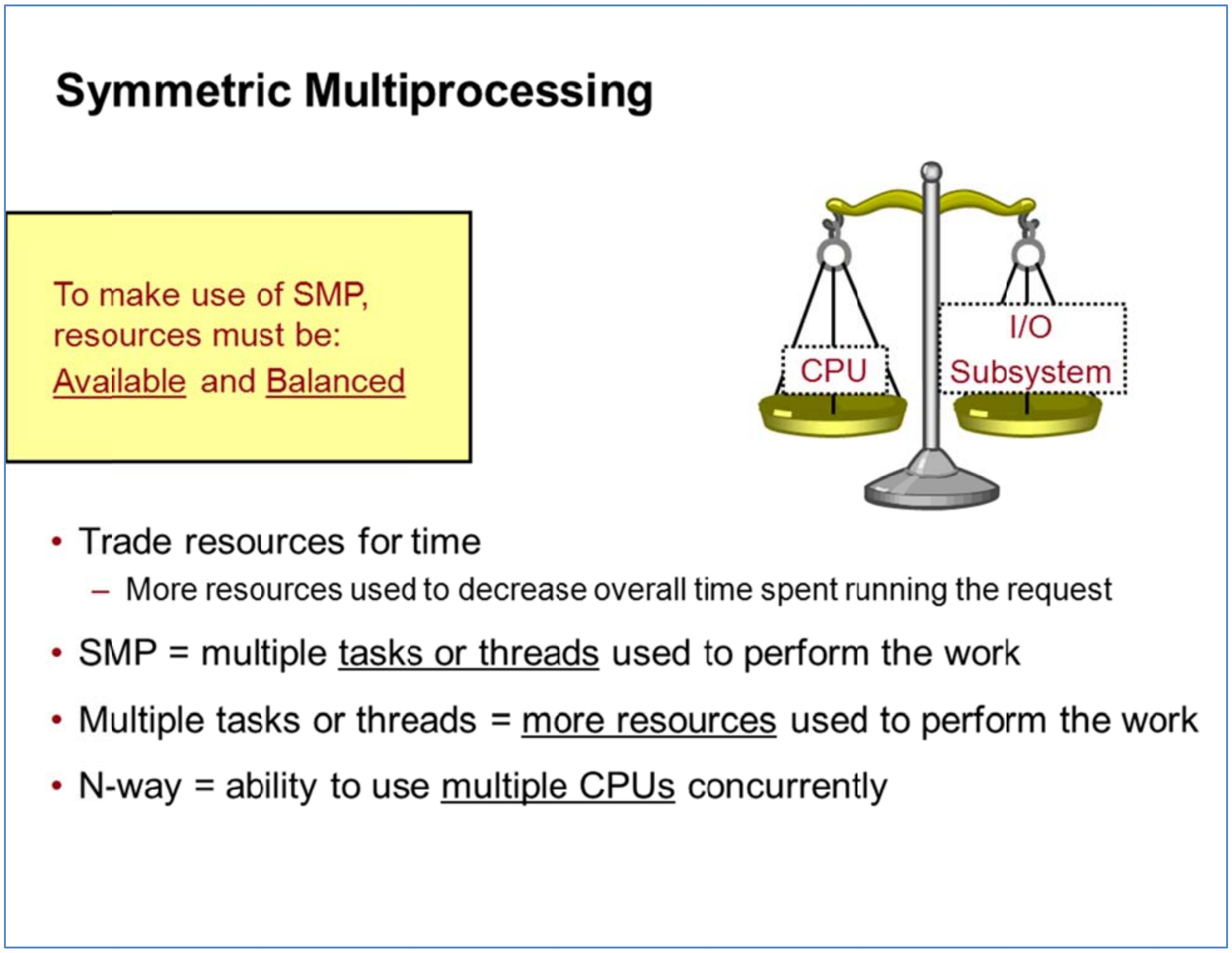
```
                          Work with System Activity                TERAPLXD
                                                           07/30/01  14:24:53
 Automatic refresh in seconds  . . . . . . . . . . . . . . . . .   __5
 Elapsed time . . . . :    00:00:03        Average CPU util . . :   99.9
 Number of CPUs . . . :    23              Maximum CPU util . . :   99.9
 Overall DB CPU util  :    99.1            Minimum CPU util . . :   99.8

 Type options, press Enter.
   1=Monitor job    5=Work with job
                                                          Total    Total     DB
      Job or                                        CPU    Sync    Async    CPU
 Opt  Task        User         Number   Thread  Pty  Util    I/O      I/O    Util
  _   DBL3Base05                                 20   2.5      0        0     1.9
  _   DBL3Base04                                 20   2.5      4        0     2.2
  _   DBL3Base00                                 20   2.4      8        0     3.8
  _   DBL3Base00                                 20   2.4      8        0     2.8
  _   DBL3Base01                                 20   2.4      8        0     2.4
  _   DBL3Base04                                 20   2.4      0        0     2.8
  _   DBL3Base03                                 20   2.3      0        0     3.0
  _   DBL3Base04                                 20   2.3      4        0     2.5
                                                                          More...
 F3=Exit   F10=Update list   F11=View 2   F12=Cancel   F19=Automatic refresh
 F24=More keys
MA     c                                                                 13/003
Connected to remote server/host dadt using port 23
```

Work Management Feedback – WRKSYSACT

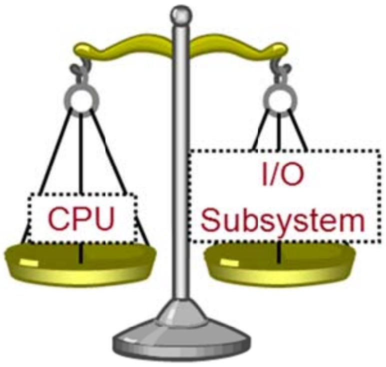
When a database request – query, index creation, etc. – uses SMP, multiple tasks or threads will be utilized. Those tasks or threads can be seen using the *work system activity* command (WRKSYSACT). This command is part of the Performance Tools license program product. There is no way to tie the SLIC tasks back to a given job. The SMP threads are associated with a given job. The SLIC tasks that support database parallelism are represented with the name “DBL3…”.

Even though the tasks or threads do much of the parallel I/O and CPU processing of a query, the accounting of the I/O and CPU resources used is transferred to the job. After the work completes the summarized I/O and CPU resources used by the tasks, threads and job will be accurately displayed by the command *work with active jobs* - WRKACTJOB. During the execution of the query, the job’s I/O and CPU resources may not include the work being performed by the tasks or threads. In other words, the tasks or threads will not have reported back any metrics to the parent job while the work is being performed.



Symmetric Multiprocessing

To make use of SMP, resources must be: Available and Balanced

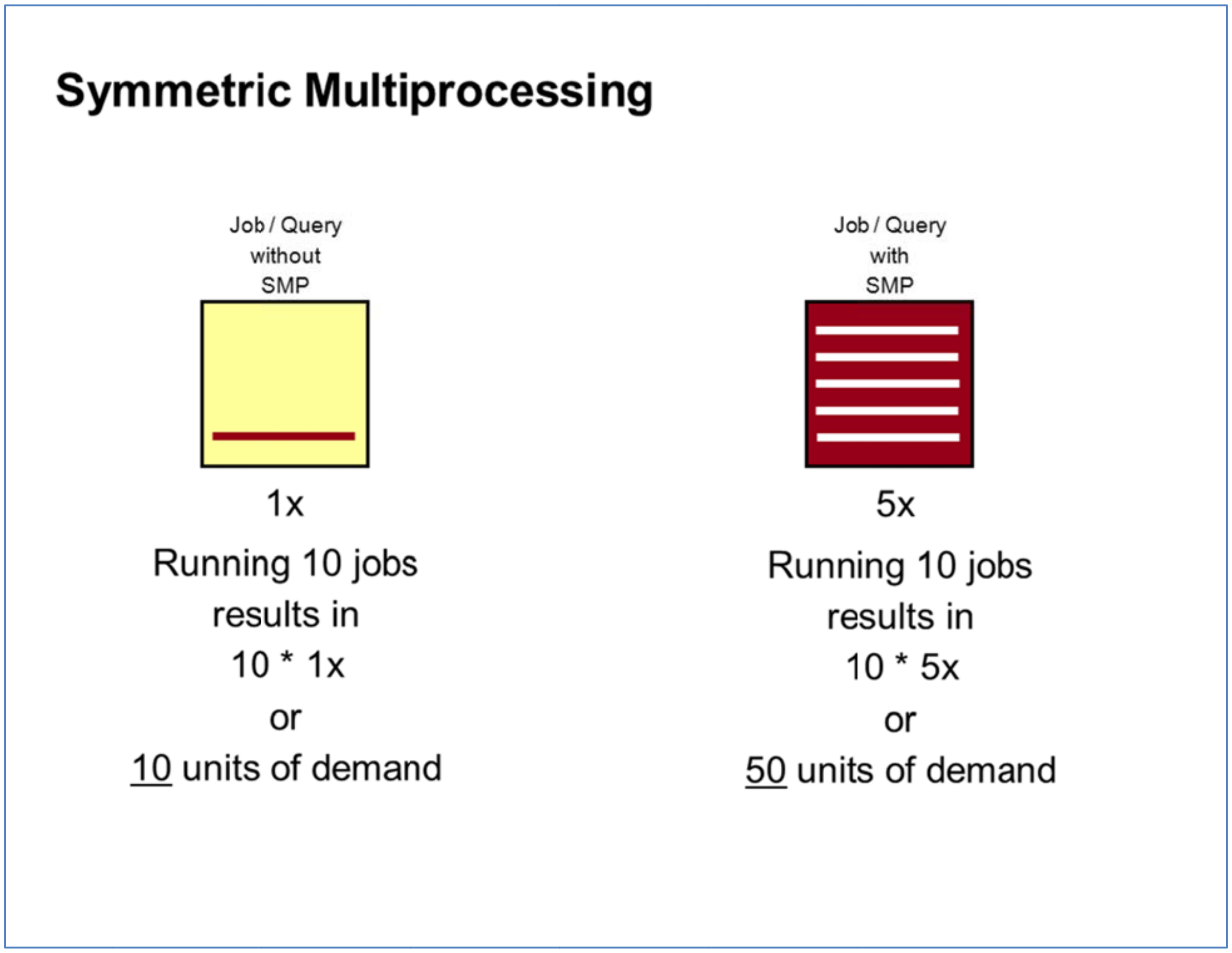


- Trade resources for time
  - More resources used to decrease overall time spent running the request
- SMP = multiple tasks or threads used to perform the work
- Multiple tasks or threads = more resources used to perform the work
- N-way = ability to use multiple CPUs concurrently

Available and Balanced Resources

To take advantage of SMP, the computing resources must be *available* and *balanced*. In other words, the system or LPAR configuration must have a good balance of CPU, memory, and disk components. Otherwise, the optimizer will not be able to make the best use of SMP. For example, a robust set of CPU resources must be supported with a robust set of I/O resources; else the processors will be waiting for I/O requests to be fulfilled. On the other hand, if the CPU resources are limited, then the amount of parallelism will also be limited.

Enabling SMP for a given job will allow that job to use multiple tasks or threads to perform the work. Those multiple tasks or threads will consume more resources with the goal of faster response times. Multiple CPUs or cores will allow the tasks or threads to execute in parallel. In other words, more work will be accomplished in the same unit of time.



Symmetric Multiprocessing

Job / Query
without
SMP

1x

Running 10 jobs
results in
10 * 1x
or
10 units of demand

Job / Query
with
SMP

5x

Running 10 jobs
results in
10 * 5x
or
50 units of demand

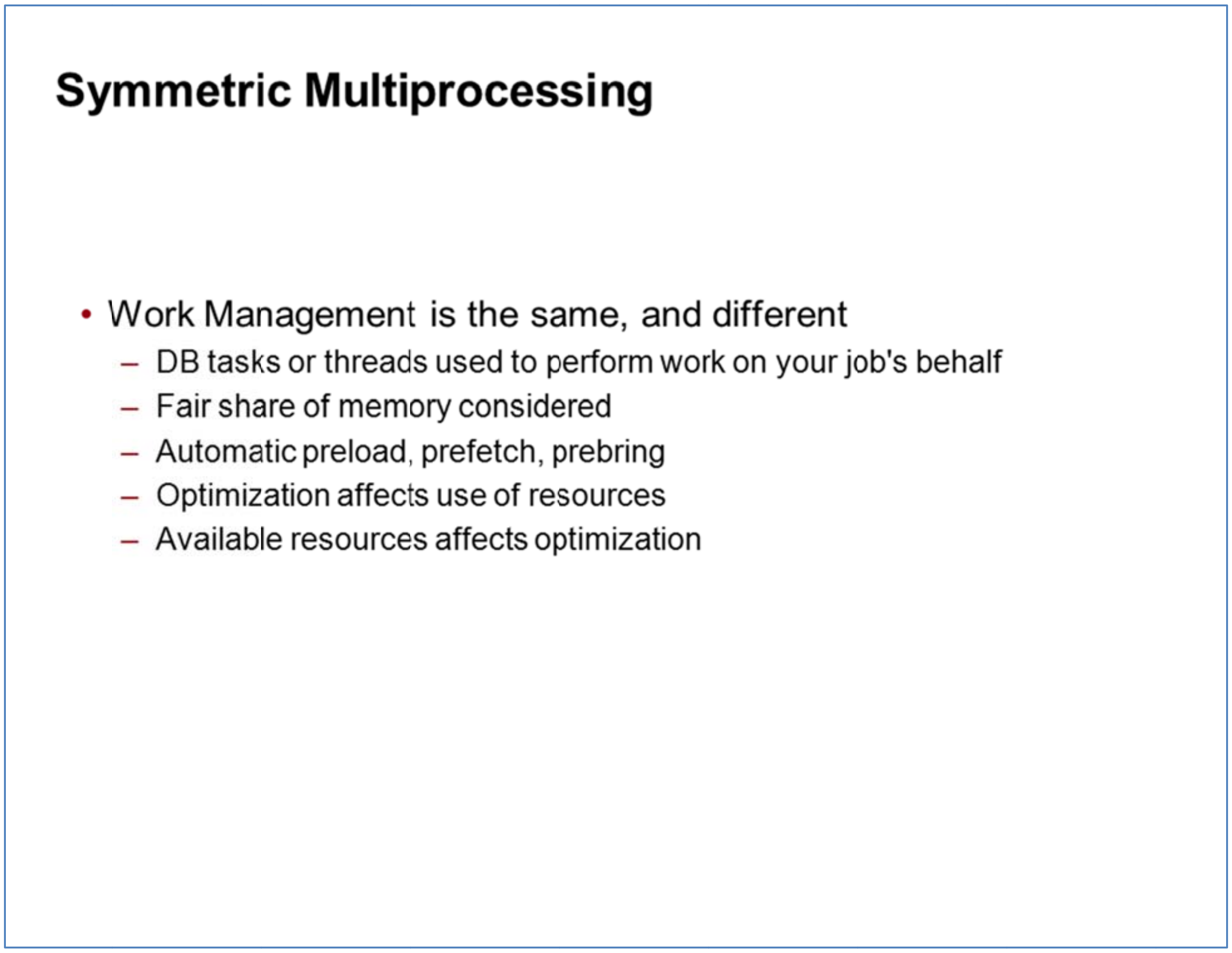
Allotting Adequate Resources

Assume a given job demands one unit of resources *without* SMP. Running 10 of those jobs simultaneously would place 10 units of demand on the system resources.

Now assume a given job demands five units of resources *with* SMP. Running 10 of those jobs simultaneously would place 50 units of demand on the system resources.

In other words, enabling the optimizer to choose parallel methods and strategies for the job will also allow that job to consume more resources. Running multiple jobs with SMP may cause even more resources to be consumed. If the resources are not available, the computing resources will be overcommitted and the throughput will be reduced.

For this reason, when considering the use of SMP, ensure there will be enough resources available to handle the increased demand.



Symmetric Multiprocessing

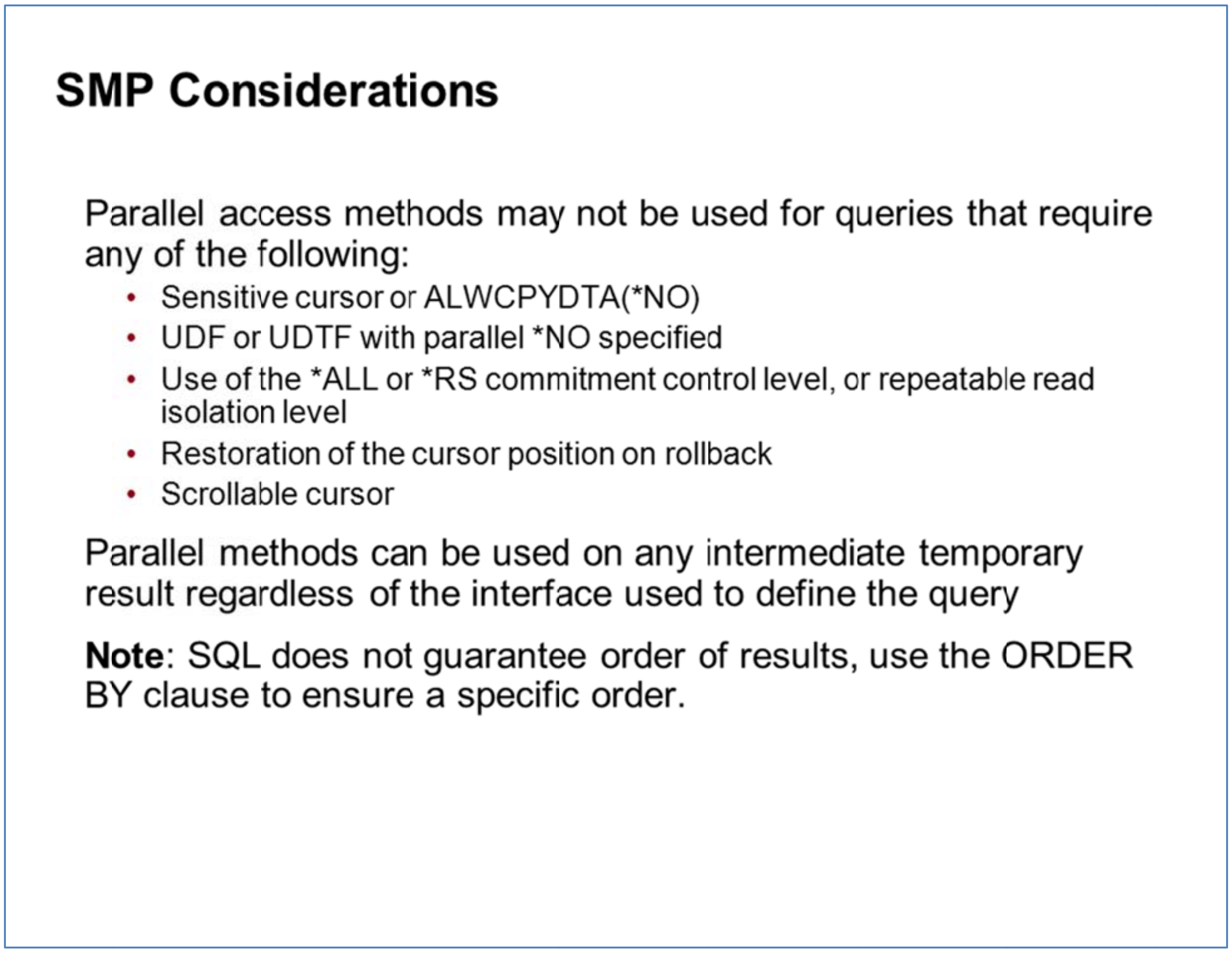
- Work Management is the same, and different
  - DB tasks or threads used to perform work on your job's behalf
  - Fair share of memory considered
  - Automatic preload, prefetch, prebring
  - Optimization affects use of resources
  - Available resources affects optimization

Work Management and SMP

With SMP-enabled workloads, IBM i Work Management processes are the same, and yet different. Work Management is the same given that the tasks and threads use the parent job's run priority and memory pool. Work Management is different given that the tasks and threads are working on behalf of the parent job.

To prevent the database engine from consuming all the computing resources for parallelism, the SMP degree also controls the optimizer. During query optimization, the job's fair share of memory is calculated. This fair share helps to determine how aggressive the query plan will be, including the amount of parallelism. All of the parallel degree values except *MAX, cause the optimizer to use a fair share of the job's memory pool. The value *MAX allows the optimizer to consider using *all* of the job's memory pool, which results in a more aggressive query plan and possibly more use of SMP.

The Max-active value for the memory pool can be viewed and changed via the WRKSYSSTS command or System i Navigator (Work Management —> Memory Pools).



SMP Considerations

Parallel access methods may not be used for queries that require any of the following:

- Sensitive cursor or ALWCPYDTA(*NO)
- UDF or UDTF with parallel *NO specified
- Use of the *ALL or *RS commitment control level, or repeatable read isolation level
- Restoration of the cursor position on rollback
- Scrollable cursor

Parallel methods can be used on any intermediate temporary result regardless of the interface used to define the query

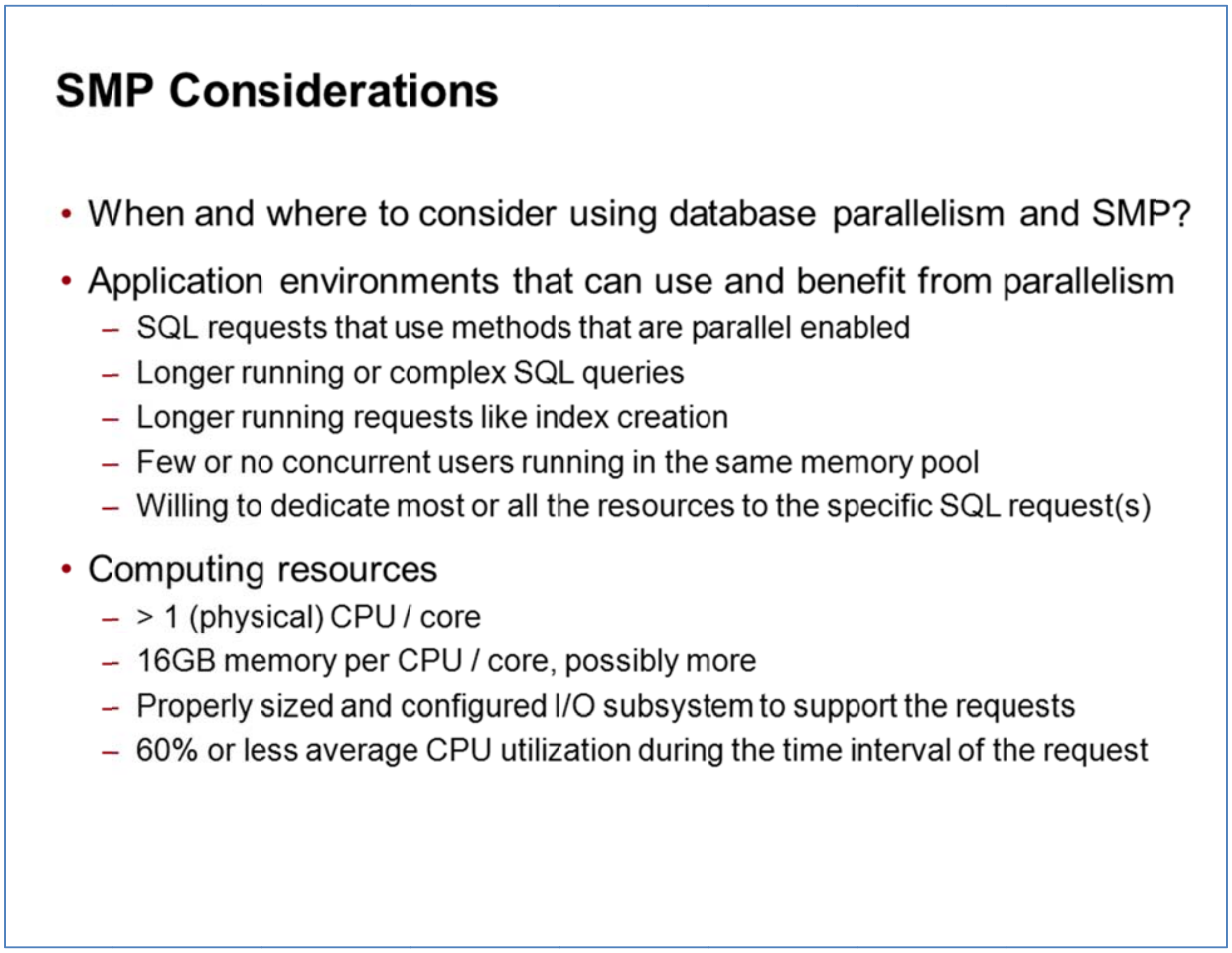
Note: SQL does not guarantee order of results, use the ORDER BY clause to ensure a specific order.

SMP Considerations: Conditions that restrict parallelism

There are some application environments in which SMP cannot be employed because parallel database techniques are not allowed, do not help, or are not available. When considering or evaluating the use of SMP, it is important to understand the application environment and attributes.

SMP may be used for some parts of the query plan and not for others. If the query plan uses temporary, intermediate results; these results may be processed with parallel methods. An example of this is the creation of a temporary index to support the query. To speed up the processing, the temporary index can be created in parallel, but accessed without parallelism.

It is important to remember that SQL does not guarantee the query results will be returned in any particular order. As a matter of fact, the same query executed multiple times may return the results in a different order each time. One reason for this phenomenon is the use of SMP. When running in parallel, the rows can be processed and returned in any order by the multiple tasks or threads. To ensure a consistent ordering of the query results, specify the SQL ORDER BY clause.



SMP Considerations

- When and where to consider using database parallelism and SMP?
- Application environments that can use and benefit from parallelism
  - SQL requests that use methods that are parallel enabled
  - Longer running or complex SQL queries
  - Longer running requests like index creation
  - Few or no concurrent users running in the same memory pool
  - Willing to dedicate most or all the resources to the specific SQL request(s)
- Computing resources
  - > 1 (physical) CPU / core
  - 16GB memory per CPU / core, possibly more
  - Properly sized and configured I/O subsystem to support the requests
  - 60% or less average CPU utilization during the time interval of the request

SMP Considerations: Application environments, computing resources

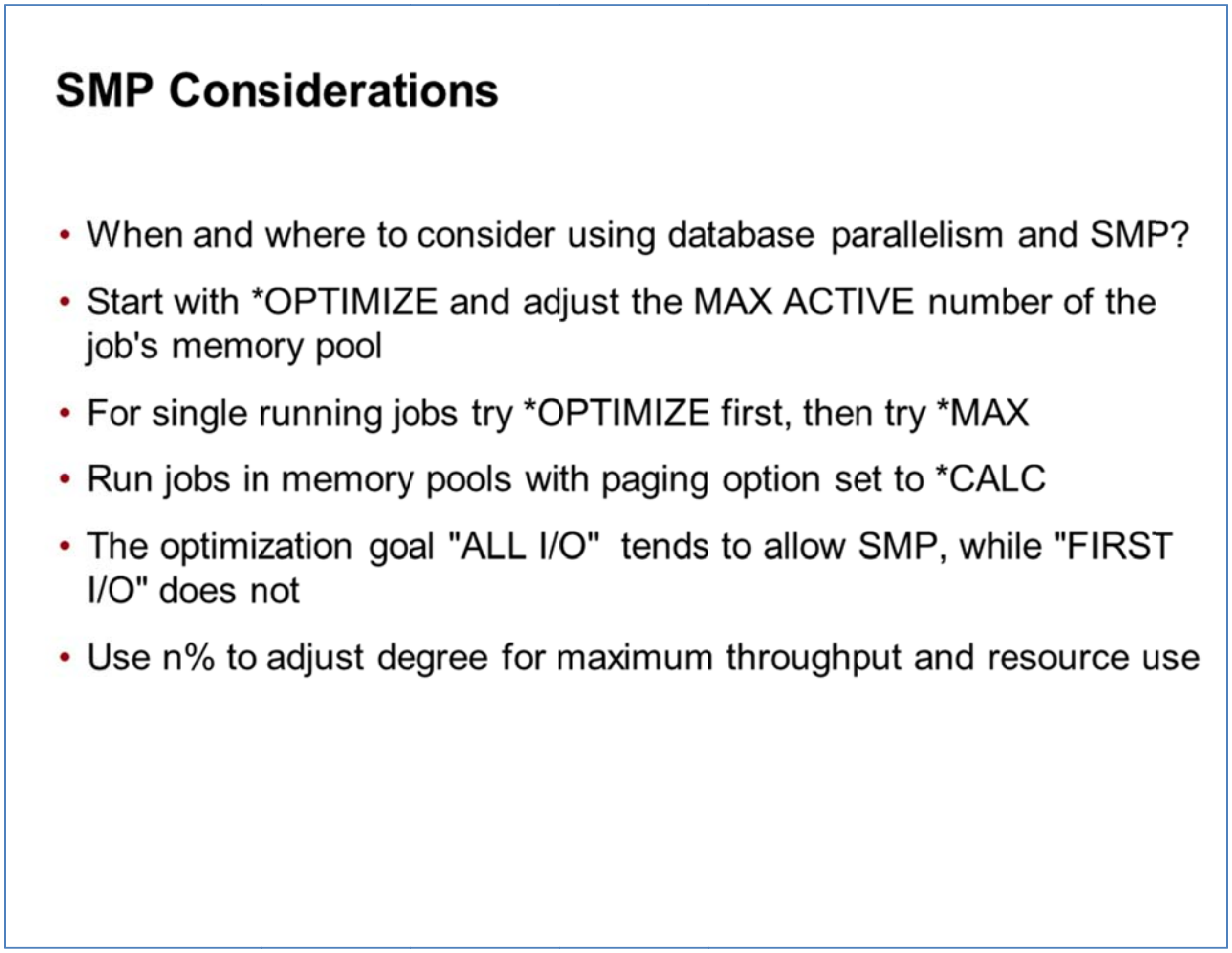
When considering the use and benefits of SMP, remember that some query requests may benefit from SMP and others may not. Similarly, some environments can support parallel database techniques while others may not. For example, if a query request is accessing a very small set of rows via an index, then that particular query will not benefit from SMP. On the other hand, if a query request is accessing a majority of the rows via a table scan, then that particular query might benefit from SMP by allowing the optimizer to use SMP to run the table scan in parallel.

If the query request is expected to access and process many rows, then allowing the request to run in parallel can increase the response time of the query. This increase in response time is not magical or free—it is the result of using more resources to do work in parallel.

The additional resources must be available during query optimization and query execution. The optimizer uses the *static* resources that are in place at the time of query optimization. These resources are: the number and type of CPUs or cores as well as their relative power, memory pool size, and number of disk units that contain the database objects being accessed. If the resources are not balanced or not sufficient to support SMP, then the optimizer will not use parallel methods and strategies.

When a parallel query plan is executed, this plan will adhere to the rules of work management. The parallel tasks or threads running on behalf of the job will compete for resources just like any other job. If there are not sufficient resources available to support the increase in workload, then the resources will become saturated and the overall throughput may decrease. If no other jobs are competing for resources, then the parallel query plan will be free to use all the available resources and gain the benefits of SMP – namely increased throughput and response time.

The best practice to understand the benefits of using SMP is to study the particular workload and environment, or run a benchmark / proof of concept. Short of that, some general guidelines can be used to assess whether or not SMP may be beneficial. A candidate SMP environment should have more than one physical (dedicated) CPU or at least 2 cores, 16 gigabytes of memory per CPU or core (POWER6 and above), and a properly sized and configured I/O subsystem. The CPU utilization should be below 60% for the interval of time when SMP will be considered. For example, assuming an SQL query workload, a nightly batch environment has four cores available and the average (total) CPU utilization is 25% during the nightly processing. To make use of the remaining CPU resources, SMP can be enabled to allow parallel database processing. On the other hand, the daily transaction environment has an average (total) CPU utilization of 85% during the daily processing. During this time interval, the extra computing resources are not available and using SMP will not increase the throughput.



SMP Considerations

- When and where to consider using database parallelism and SMP?
- Start with *OPTIMIZE and adjust the MAX ACTIVE number of the job's memory pool
- For single running jobs try *OPTIMIZE first, then try *MAX
- Run jobs in memory pools with paging option set to *CALC
- The optimization goal "ALL I/O" tends to allow SMP, while "FIRST I/O" does not
- Use n% to adjust degree for maximum throughput and resource use

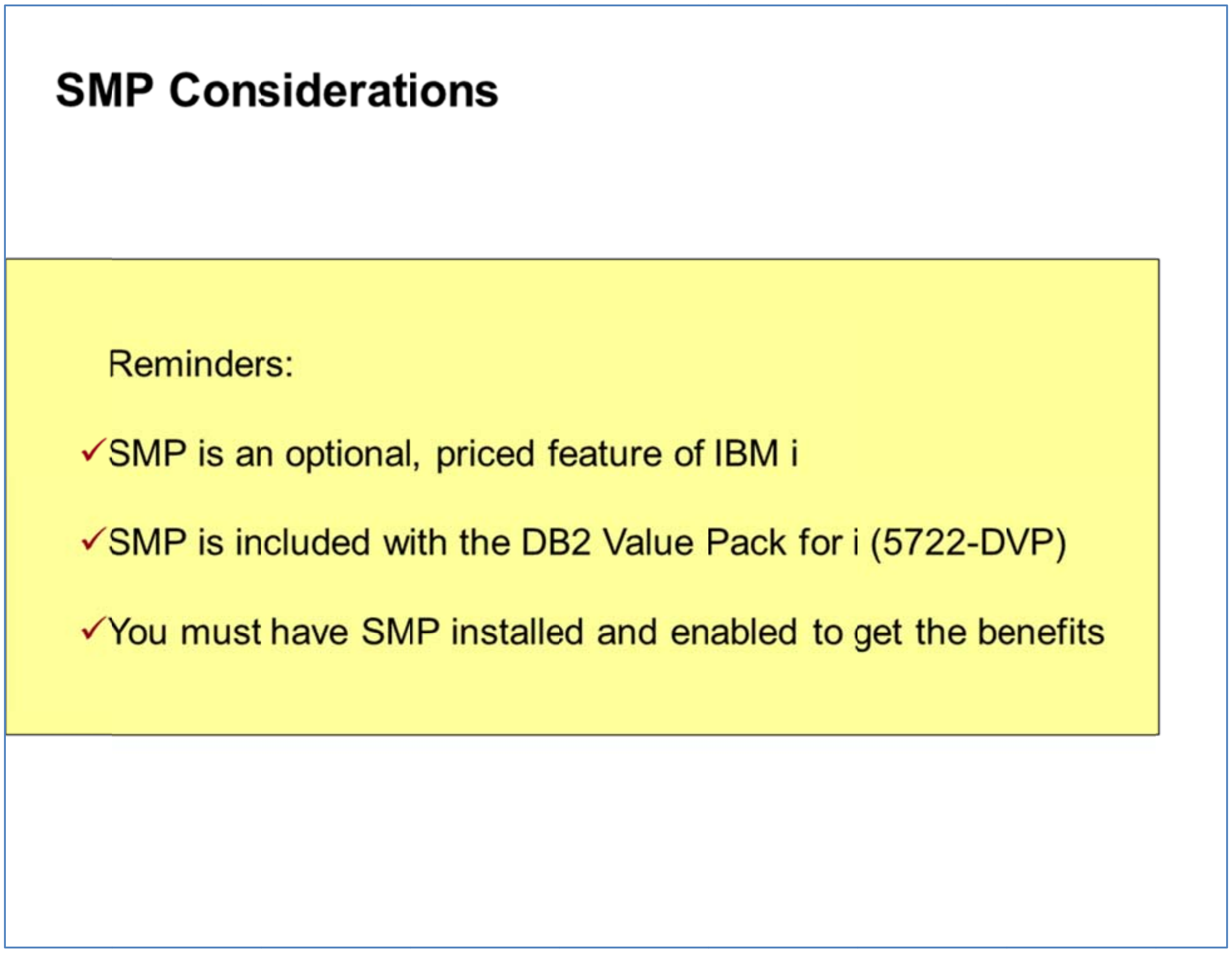
SMP Considerations: First time tips

When using or experimenting with SMP for the first time, start with the parallel degree set to *OPTIMIZE and analyze the query plan and run time. Set the MAX ACTIVE value for the memory pool to a number that represents the true number of *concurrent*, active jobs or threads. This will allow the optimizer to determine a realistic fair share of the memory pool for the jobs.

For jobs that are running alone in a memory pool, start by using *OPTIMIZE, and then move to *MAX. *MAX will allow the optimizer to consider using all the memory in the pool, but will also let the database engine use more CPU resources.

Setting the memory pool’s paging option to *CALC will allow the database engine to be more intuitive and more aggressive with I/O requests. The value of *CALC allows more parallel, asynchronous I/O.

Based on the application’s behavior and SQL interface, the optimizer can use an optimization goal of *First I/O* or *All I/O*. The First I/O optimization goal tends to avoid parallel methods and strategies given that these plans are built to deliver the first n rows of the result set as fast as possible, and a faster query startup time is required. The All I/O optimization goal tends to allow parallel methods and strategies, given that (1) these plans are built to deliver the entire result set as fast as possible, and (2) a slower query startup time is acceptable. Using parallel methods and strategies may require some additional initialization time during query startup in favor of a faster overall query run time.



SMP Considerations

Reminders:

✓SMP is an optional, priced feature of IBM i

✓SMP is included with the DB2 Value Pack for i (5722-DVP)

✓You must have SMP installed and enabled to get the benefits

Conclusion

To receive the benefits of parallel database processing, the DB2 Symmetric Multiprocessing feature must be installed and enabled. This feature is included in the DB2 Value Pack for i (5722-DVP) or can be ordered separately.

Db2 Symmetric Multiprocessing provides high performance database processing and in and excellent way in which to take advantage of all the available resources within the POWER system running IBM i. This feature relies on the integrated technologies found in IBM i and DB2 for i, and is an example of the outstanding value provided by one of the best business systems available today.

Links

More information regarding DB2 for i and the data-centric technologies within IBM i can be found at the following Web sites:

DB2 for i home page:

ibm.com/systems/i/software/db2/

IBM DeveloperWorks article: **Access your DB2 for iSeries Indexes in Parallel**, by Kent Milligan

ibm.com/developerworks/db2/library/techarticle/0301milligan/0301milligan.html

White paper: **IBM DB2 for i indexing methods and strategies**, by Mike Cain and Kent Millgan

ibm.com/partnerworld/wps/servlet/ContentHandler/stg_ast_sys_wp_db2_i_indexing_methods_strategies

For additional assistance with getting the most out of DB2 for i, send an e-mail to Mike Cain at: mcain@us.ibm.com

Trademarks and Special Notices