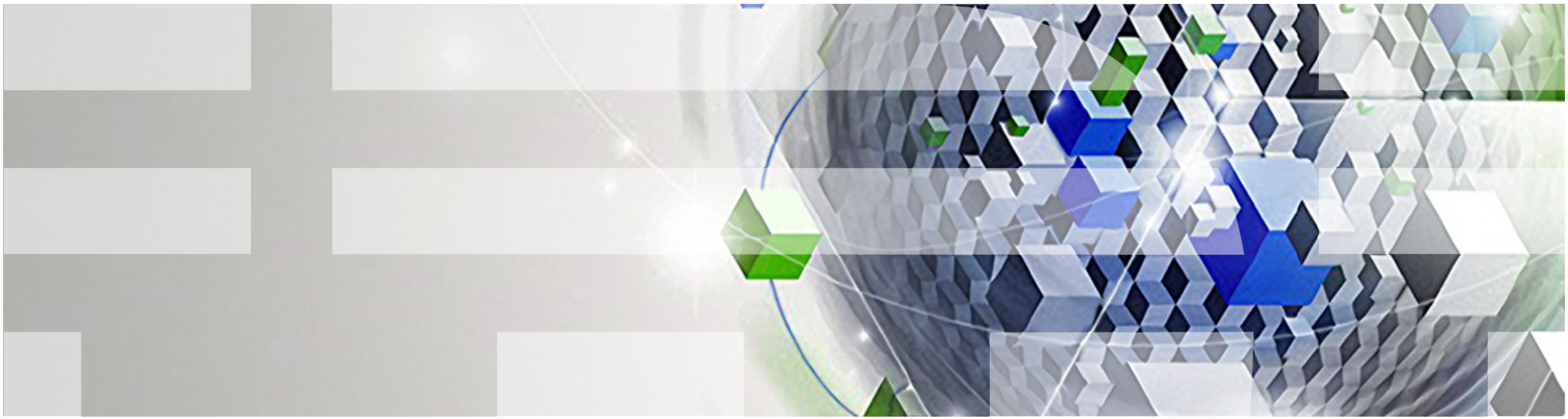
Kent Milligan
IBM Lab Services – Db2 for i
Rochester, MN USA
kmill@us.ibm.com

Db2 Symmetric Multiprocessing

June 2022

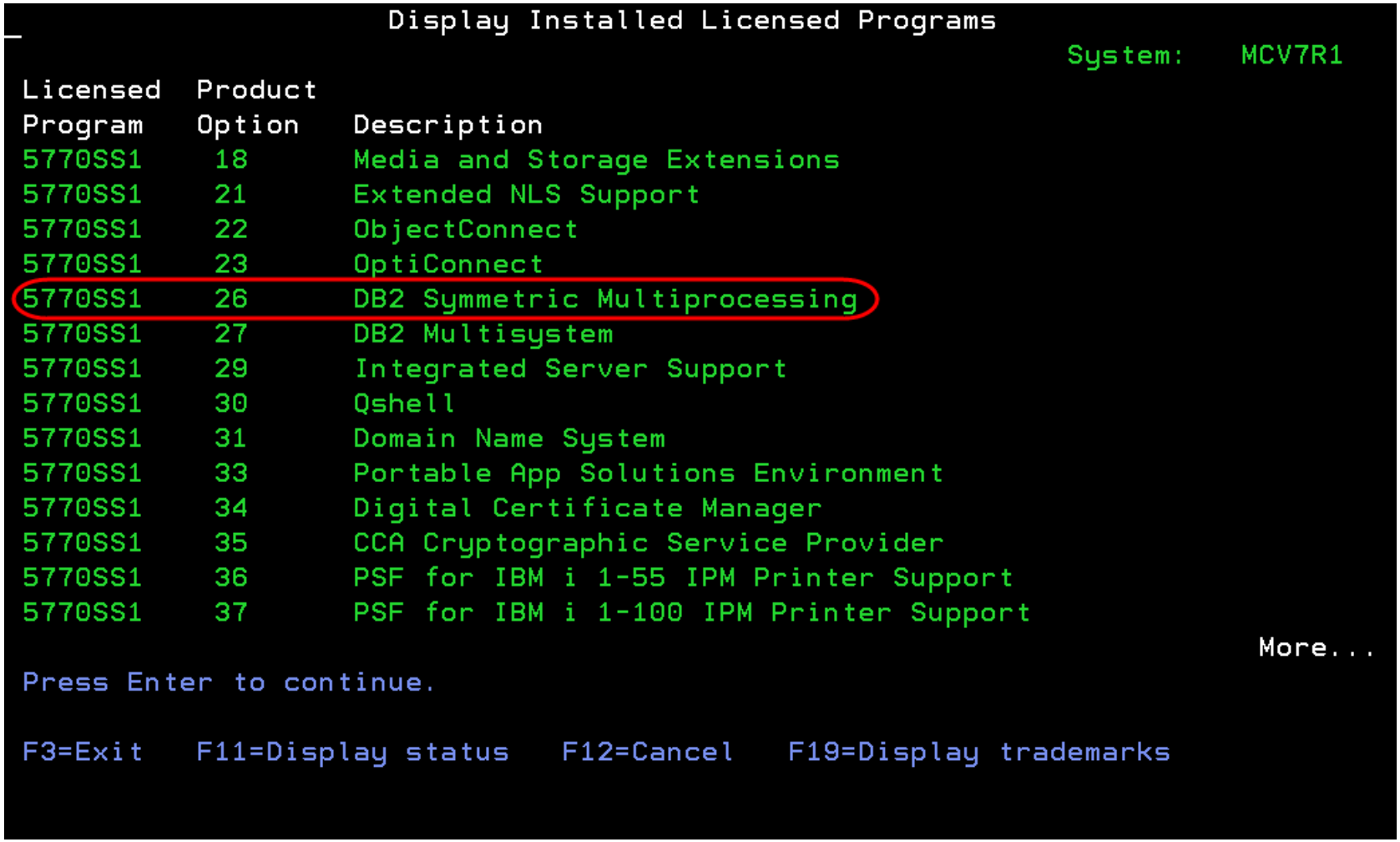


Db2 Symmetric Multiprocessing for IBM i

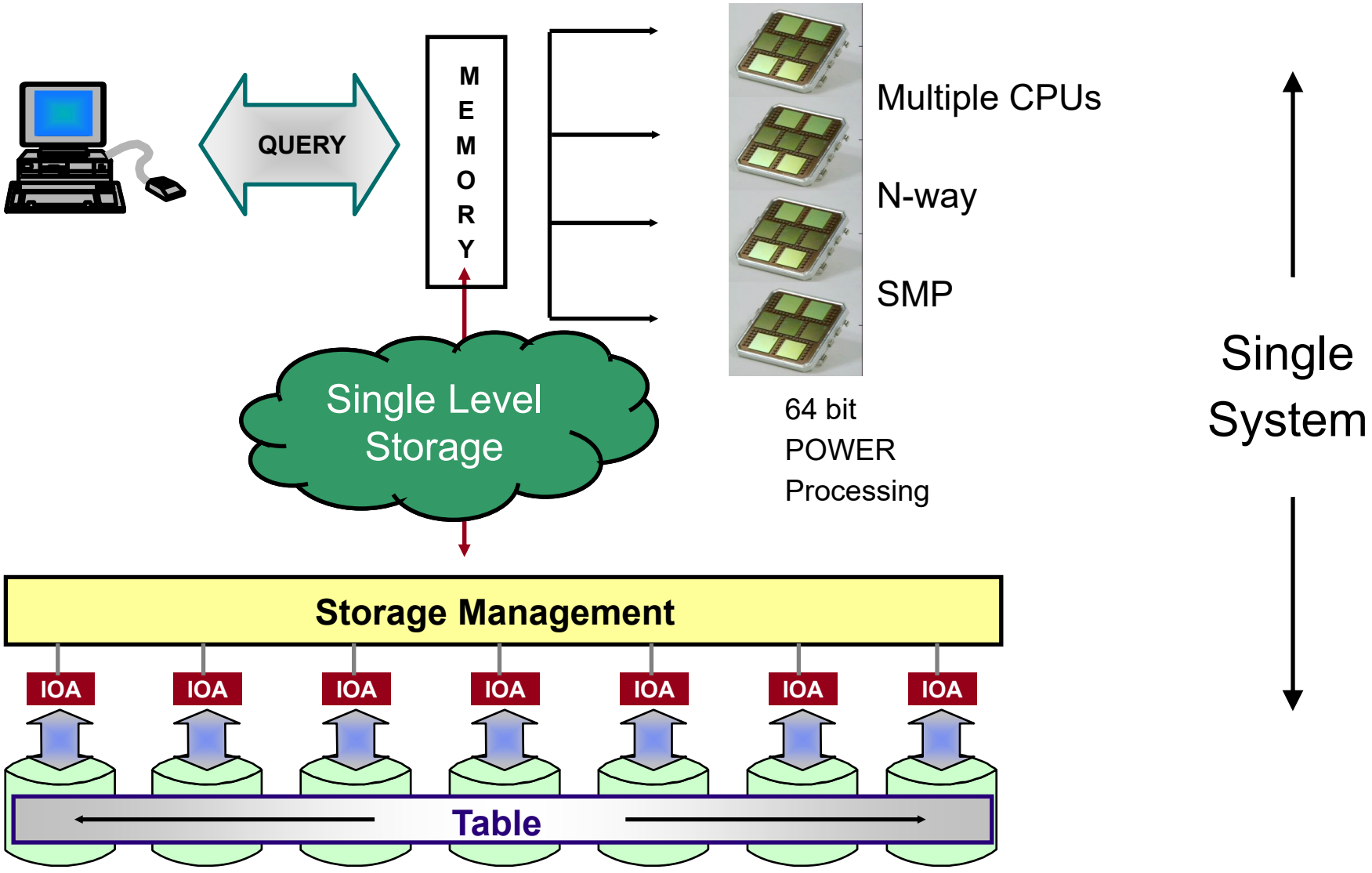
Database parallelism,
while inherently part of Db2 for i,
is enabled by installing the optional IBM i feature
"Db2 Symmetric Multiprocessing"

Symmetric Multiprocessing

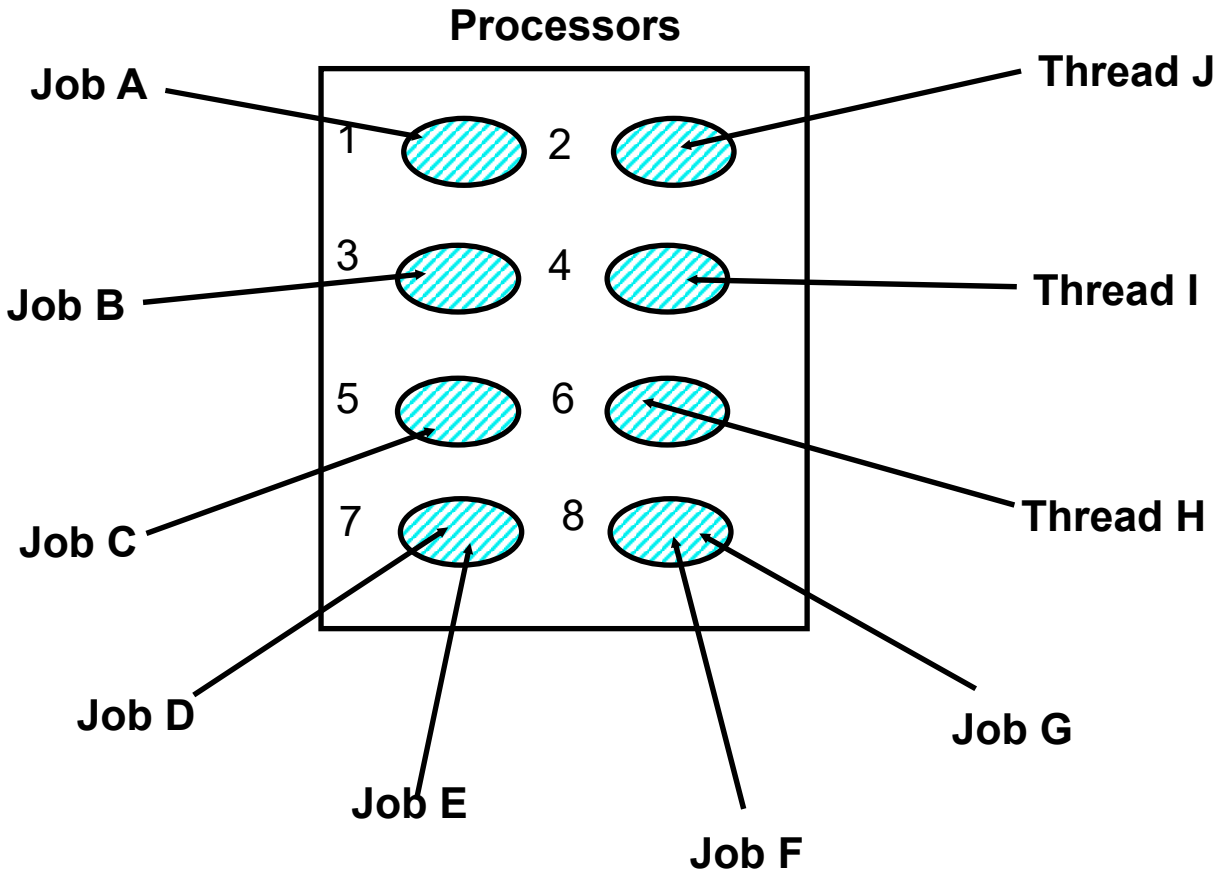
GO LICPGM, option 10 Display installed licensed programs



IBM i Architecture

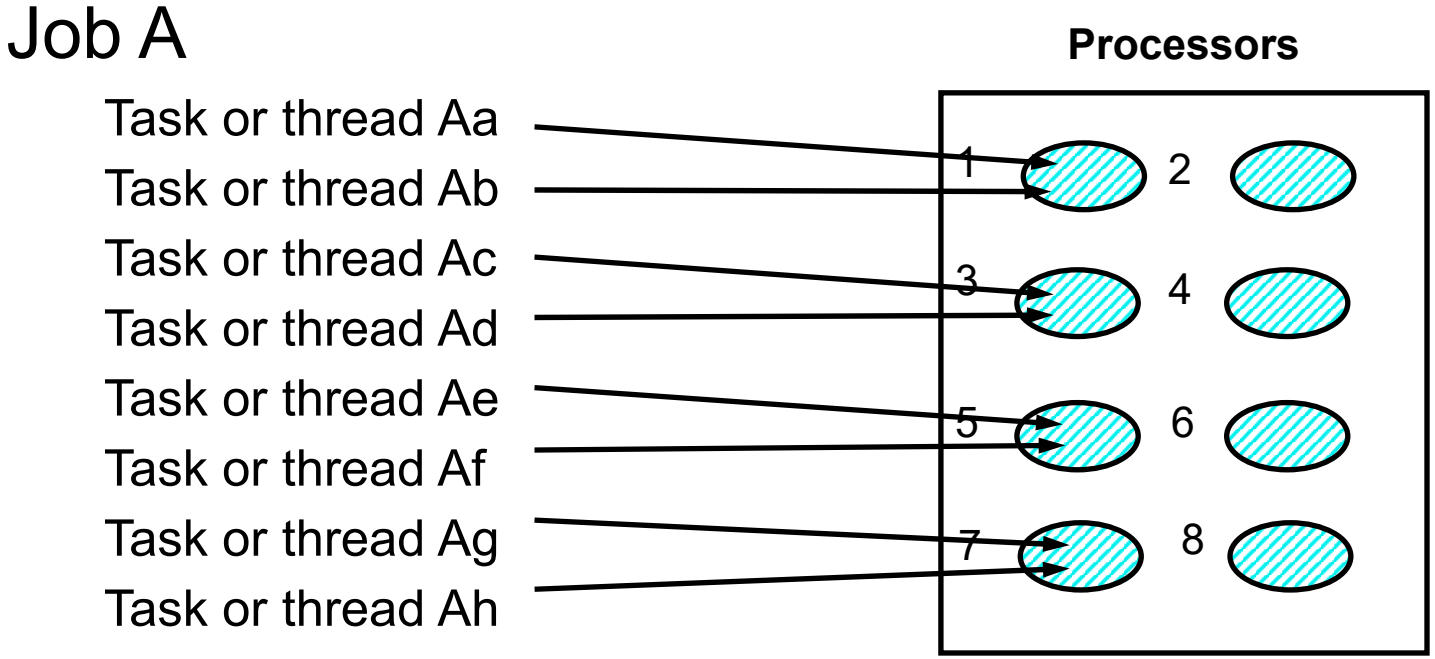


N-way processing



- ✓ 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

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)**

- ✓ **Db2 for i parallelism does <u>not</u> require table space partitioning**

Symmetric Multiprocessing

- 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

- 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)

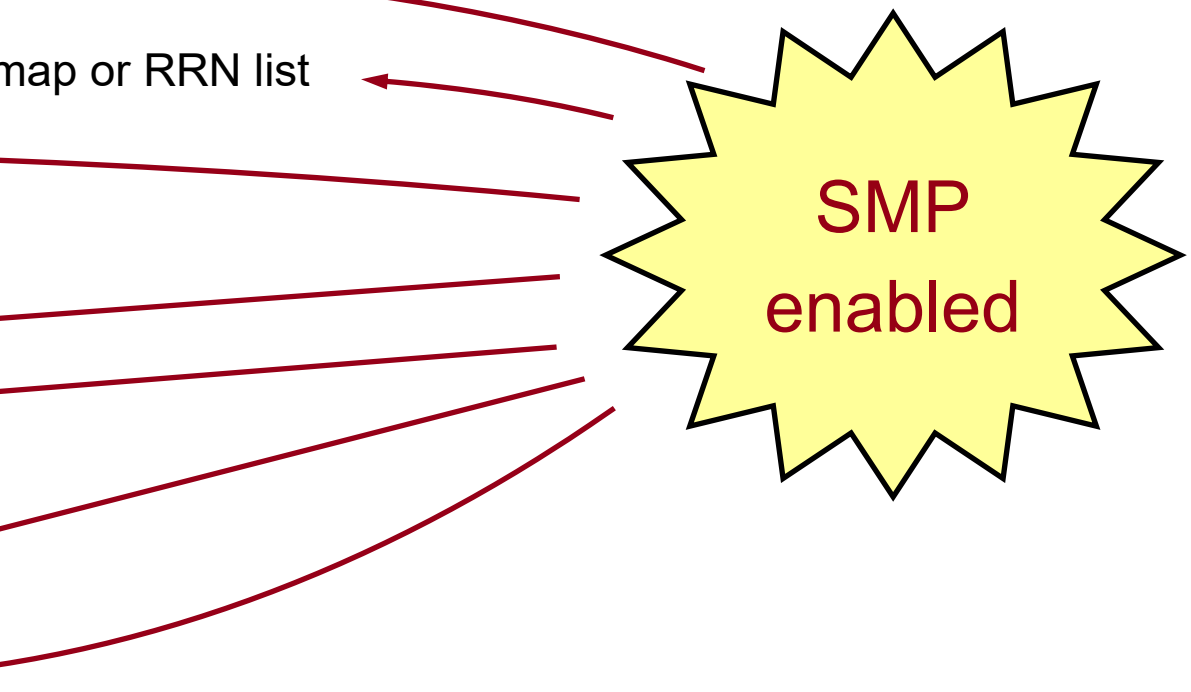
Symmetric Multiprocessing

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)

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

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, but limited benefits with a single processor system

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: *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 (only available via QAQQINI)
    - *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 (only available via QAQQINI
    - *MAX 50%

Degree Parameter Values: *SYSVAL and *NBRTASKS

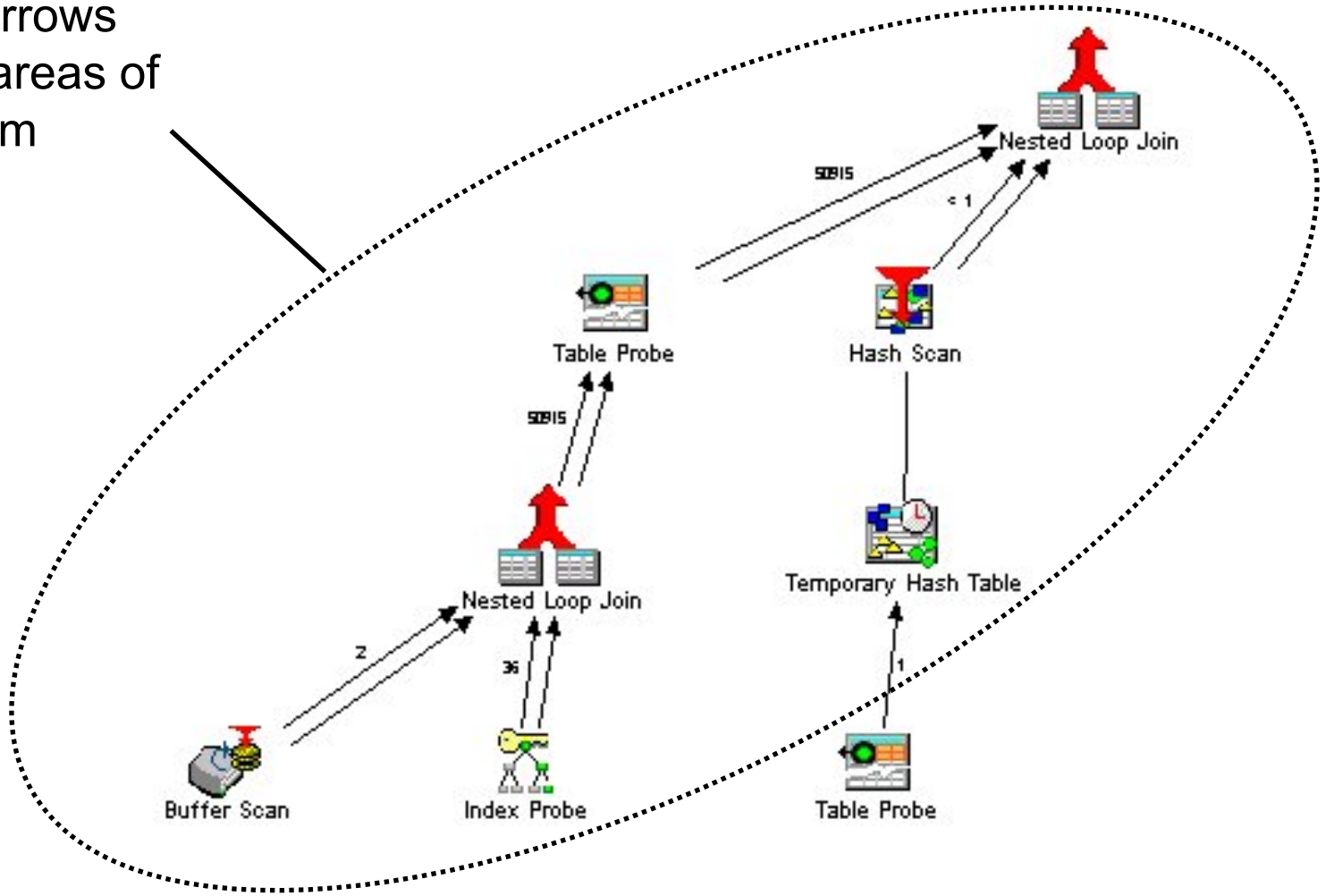
- *SYSVAL
  - Specifies that the processing option used should be set to the current value of the system value, QQRYDEGREE
  - Used with CHGQRYA command to “reset” the degree value
- *NBRTASKS nn
  - Specifies the number of tasks or threads to be used when the query optimizer chooses to use CPU parallel 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

Background Database Server Jobs

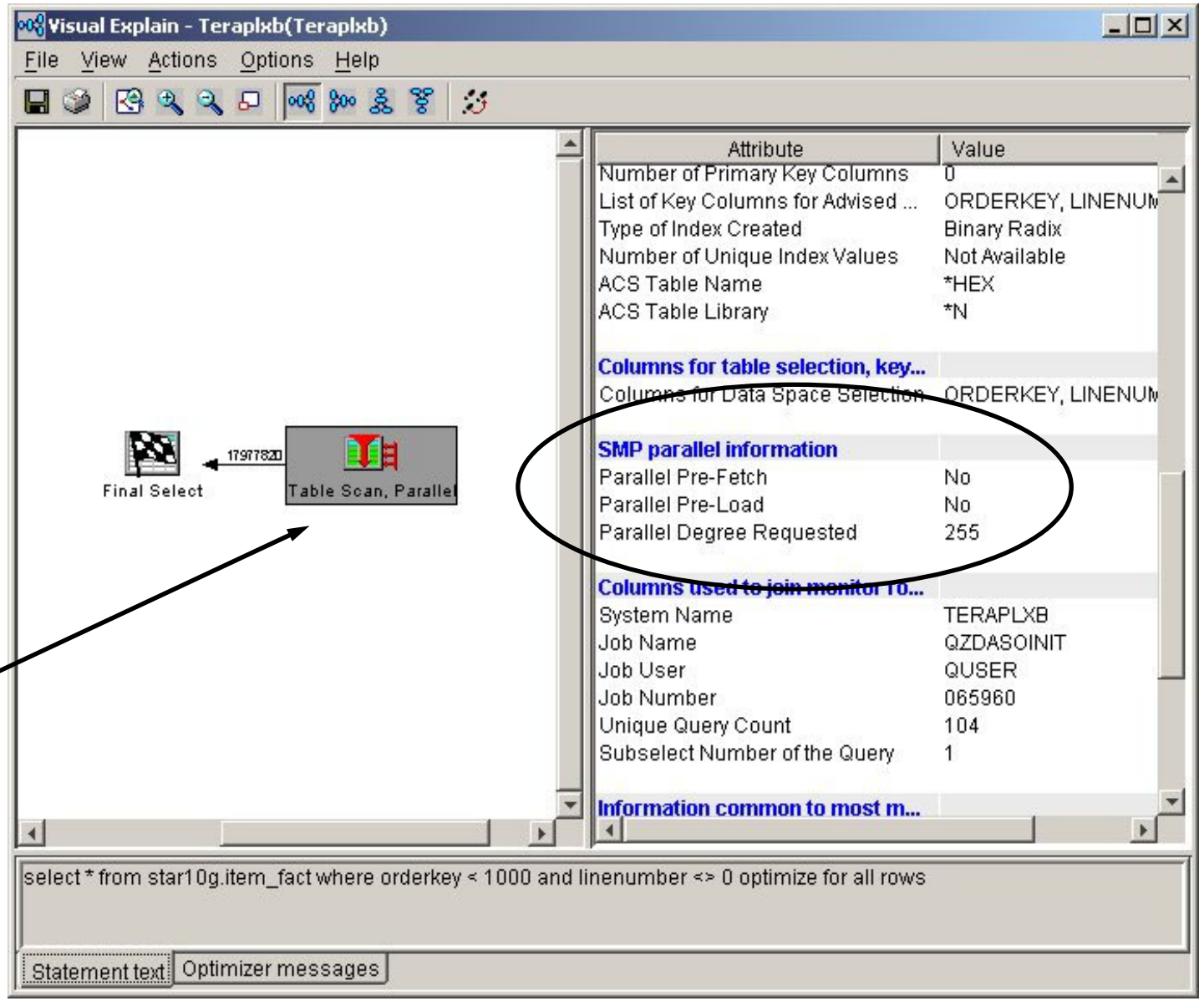
- QDBSRVxx jobs handle asynchronous requests such as rebuilding or refreshing indexes after restore or alter operations
  - 2 jobs per CPU + 1
  - Jobs are started at IPL
- QDBSRVxx jobs get assigned their parallel degree prior to handling each request
- To change the degree, issue a CHGQRYA DEGREE(...) 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

Feedback - SQE

Double arrows indicate areas of parallelism



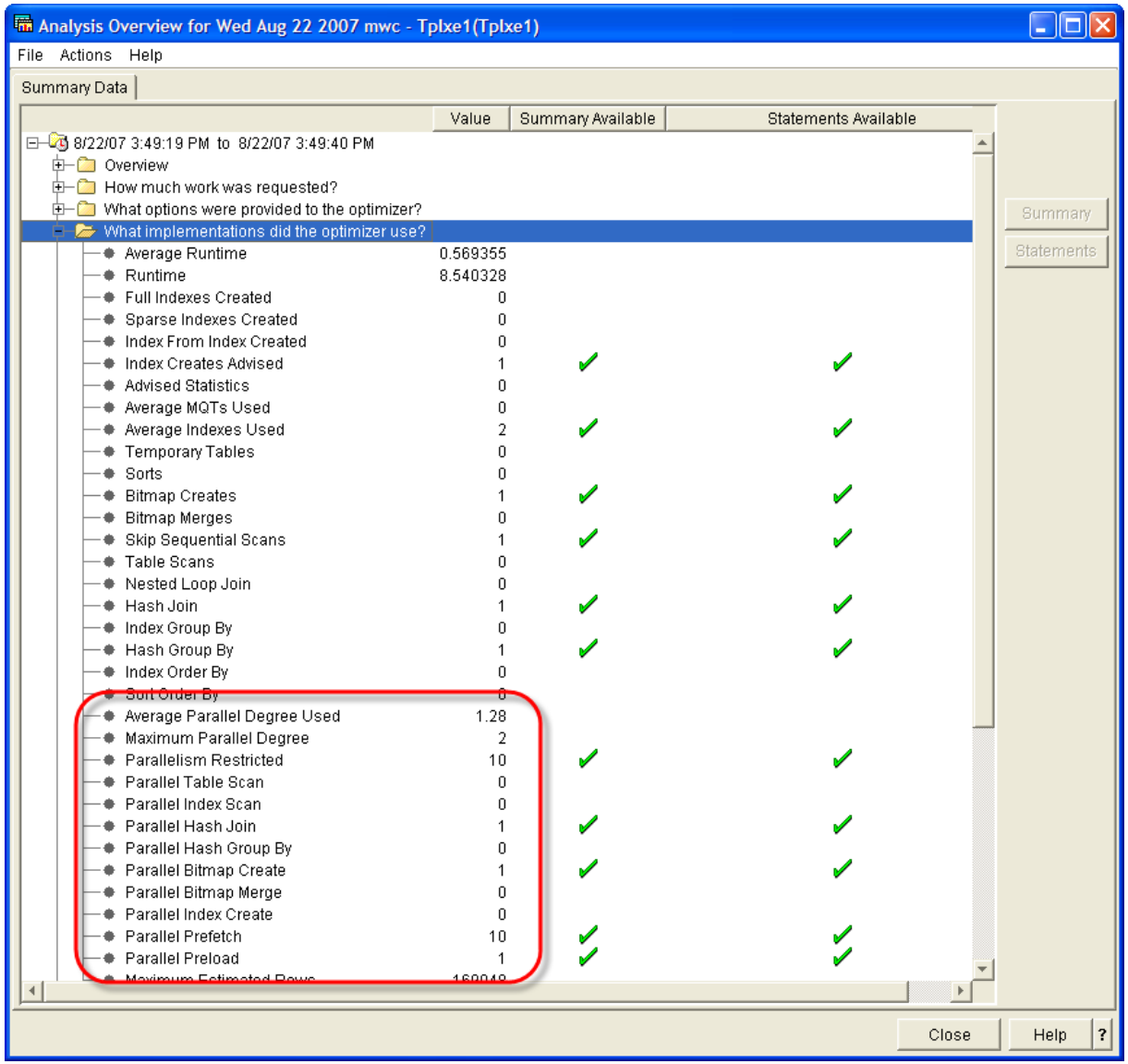
Feedback - CQE

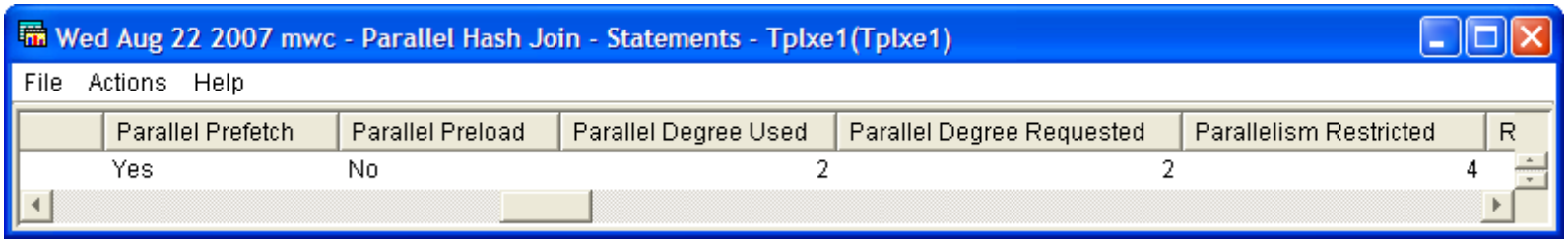


Icon indicates areas of parallelism

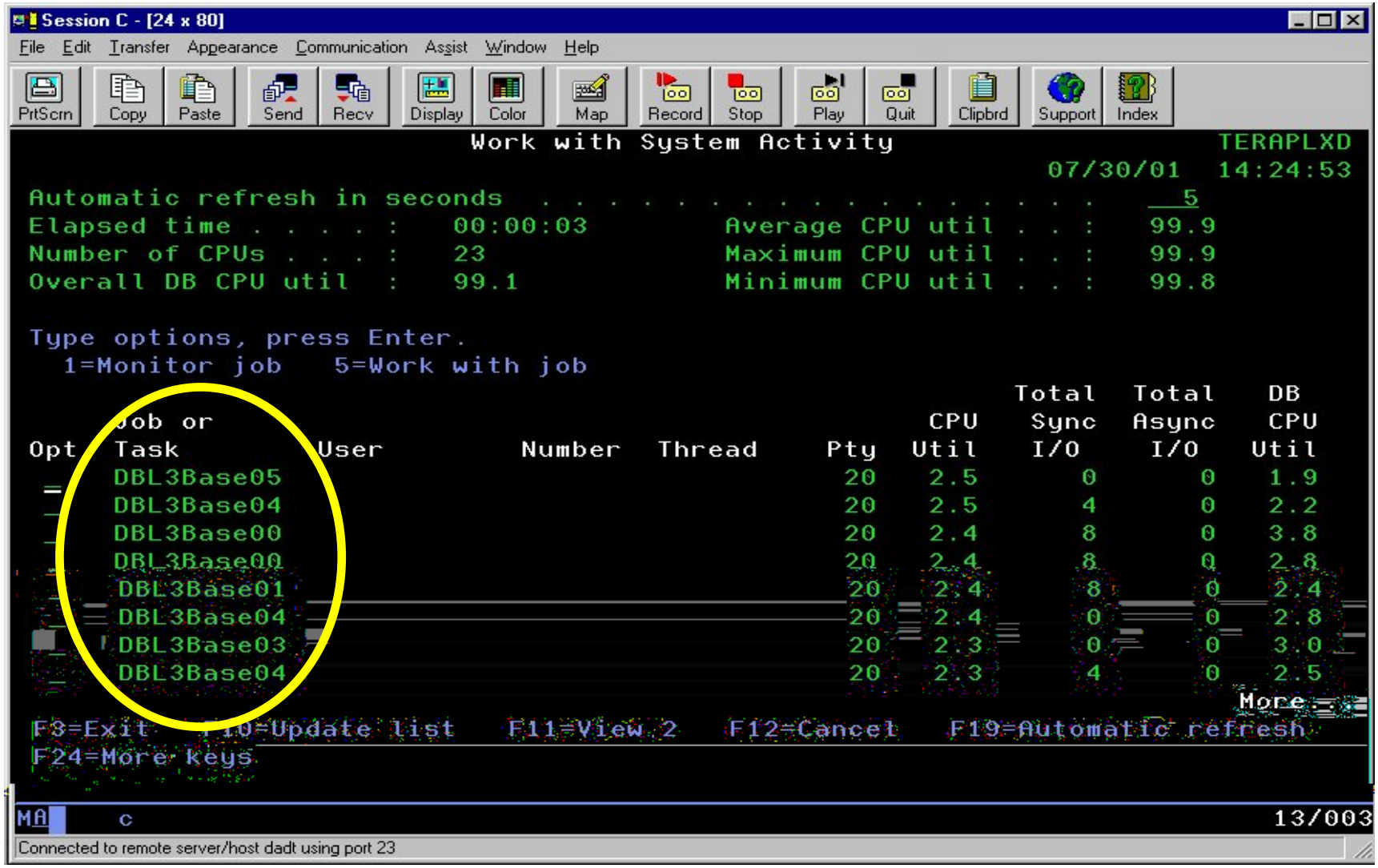
Feedback

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



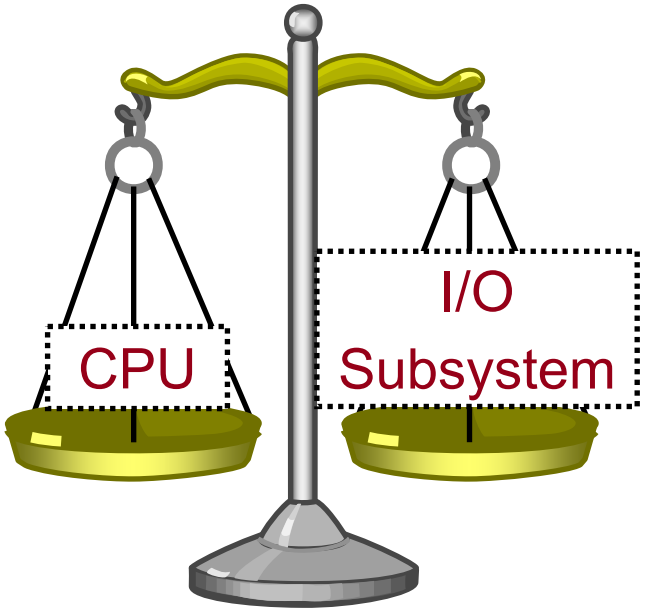


Work Management Feedback - WRKSYSACT



Symmetric Multiprocessing

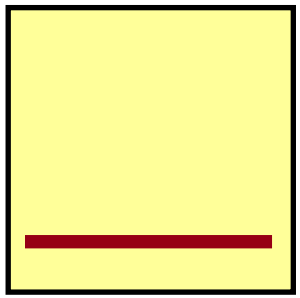
To make use of SMP, resources must be:
Available and Balanced



- 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

Symmetric Multiprocessing

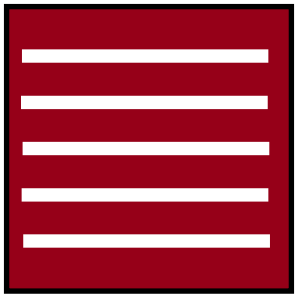
Job / Query
without
SMP



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

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

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

- 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 (limited benefit with single CPU)
  - 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

- When and where to consider using database parallelism and SMP?
  - Start with *OPTIMIZE on longer running queries
  - 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

Reminders:

- ✓Db2 SMP is a no charge feature as of June 1, 2022 (Announcement letter)
- ✓Db2 SMP must be installed and enabled to get the benefits