Query Parallelism in DB2 for z/OS

Bryan F. Smith  bfsmith@us.ibm.com
IBM

Session Code: 8361
Thursday, March 3, 2011: 3:00 PM-4:00 PM
ACC, Room 211A
Abstract

• Query parallelism was implemented in stages.
  • DB2 Version 3 introduced query I/O parallelism, which enables a much greater I/O throughput for I/O-intensive queries.
  • DB2 Version 4 introduced query CP parallelism, which added the capability to parallelize the processing operations needed to service a query. Queries requiring large sorts, joins, or complex computations can benefit by exploiting the power of a multi-processor system.
  • DB2 Version 5 introduced Sysplex query parallelism, which extends query CP parallelism "parallel tasks" to run across all DB2 members in the data sharing group. Sysplex query parallelism can use the combined processing power available within a Parallel Sysplex.
  • DB2 9 and 10 continue to add additional functionality for query parallelism
Objective for Parallel Queries

- The target for Query Parallelism is long-running queries
  - I/O-intensive Queries - Tablespace Scans, Large Index Scans
  - Processor-intensive Queries - Joins, Sorts, Complex Expressions
- Objective: Reduce elapsed time by exploiting parallel resources:
  - I/O bandwidth
  - Processing power

```
SELECT DISTINCT(CUSTOMERS), PURCHASE_PRICE
FROM CUST_TAB C, ORDER_TAB O
WHERE C.CUST_NO = O.CUST_NO AND
  LAST_ORDER_DATE > :LASTMONTH
ORDERBY PURCHASE_PRICE;
```
I use the following scenario to illustrate concepts of query parallelism. It does not imply a method of how you should implement your database design. Here is the cast of characters:

**Application**

**End User**

**Physical Design**

**DBA/SysProg View**

**PLAN_TABLE output from EXPLAIN**

| ACCESSIONTYPE | PREFETCH |

**Performance Data**

| cp0 | cp1 | cp2 | cp3 |

**Accounting Trace**

| CP Time: | xx mins |
| Elapsed Time: | yy mins |
Sequential Processing

End User

"Show me which customers I should send information to..."

Application

SELECT CUSTOMER, REGION
FROM CUST
WHERE LASTVISIT < (CURRENT DATE - 180 DAYS);

Physical Design

CUSNO index (clustering)

CUST table

DBA/SysProg View

REORGs
RUNSTATS
Buffer pool tuning
Dataset placement
Work priority

PLAN_TABLE output from EXPLAIN

<table>
<thead>
<tr>
<th>ACCESTYPE</th>
<th>PREFETCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
</tbody>
</table>
Sequential Prefetch - a form of parallelism that allows concurrent CPU and I/O processing

Table space ("R"elational) scan

**PLAN_TABLE output from EXPLAIN**

<table>
<thead>
<tr>
<th>ACCESSSTYPE</th>
<th>PREFETCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
</tbody>
</table>
"It's not fast enough!"

End User:
I want this thing to run while I go get coffee. It takes me 15 minutes to walk across the street to the coffee bar.

Steps to improve performance

1. Empathize with the user
2. Understand requirement
   - Analyze accounting trace
     
     **Accounting Trace**
     
     | CP Time: | 15 mins |
     | Elapsed Time: | 60 mins |
   
   - Req: (reduce ET by 4x)
3. Understand access path
   
   **PLAN_TABLE output from EXPLAIN**

<table>
<thead>
<tr>
<th>ACCESTYPE</th>
<th>PREFETCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
</tr>
</tbody>
</table>
4. Would an index help?
   - Yes (on LASTVISIT)
   - Suppose other appls cannot afford this index -> not an option
5. Determine bottleneck
   
   I/O / CP ratio is 4! (assume no other significant class 3 wait time)

Assuming that a single processor can drive four I/O paths to 100% busy simultaneously (this was true for S/390 G-2 CMOS processors!):

We should be able to reduce ET by 4x 60 mins -> 15 by opening up I/O bottleneck.

I/O-intensive
I/O-bound
25% of 1 CP
I/O Parallelism (introduced in Version 3)

**Physical Design**

CUSTNO index

CUST table

**Application**

```sql
SET CURRENT DEGREE = 'ANY'; <- or, bind with DEGREE(ANY)
SELECT CUSTOMER, REGION
FROM CUST
WHERE LASTVISIT < (CURRENT DATE - 180 DAYS);
```

**PLAN_TABLE output from EXPLAIN**

<table>
<thead>
<tr>
<th>ACCESS TYPE</th>
<th>PRE FETCH</th>
<th>ACCESS DEGREE</th>
<th>ACCESS PGROUP_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

**DBA's View**

- VPSIZE = 1000 buffers
- VPSEQT = 80% of VPSIZE (800 buffers)
- VPPSEQT = 50% of VPSEQT (400 buffers)

- Amount of resources consumed by parallel tasks (both I/O and CP) is controlled by buffer pool allocation parameter VPPSEQT
- Prefetch I/O buffers for parallel tasks are cast out with MRU scheme instead of the usual LRU scheme.

Parallel Degree

The number of parallel operations in a parallel group.

Parallel Group

A group of operations dedicated to performing a specific portion of a query's access path.
I/O Parallelism (Version 3)

Physical Design

CUSTNO index

CUST table

Accounting Trace

<table>
<thead>
<tr>
<th>CP Time:</th>
<th>15.37 mins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed Time:</td>
<td>15.5 mins</td>
</tr>
</tbody>
</table>

Performance Data

I/O-intensive Processor-bound
I/O-bound
100% of 1 CP

"Wow, it's done and my coffee is still warm!"

PLAN_TABLE output from EXPLAIN

<table>
<thead>
<tr>
<th>ACCESS_TYPE</th>
<th>PRE_FETCH</th>
<th>ACCESS_DEGREE</th>
<th>ACCESS_PGROUP_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

No wait for processor
I/O Parallelism (Version 3)

What is I/O Parallelism doing?

- I/O Parallelism exploits sequential prefetch to achieve parallelism
- Multiple prefetch I/O streams are started by a single DB2 agent

```
WHERE LASTVISIT < (CURRENT DATE - 180 DAYS)
```

Degree of Parallelism = 4
I/O Parallelism DEGREE

Application program considerations

- Controlling when parallelism can be used
  - DEGREE parameter on bind for *static* queries
  - CURRENT DEGREE special register for *dynamic* queries
    - Acceptable Values:
      - '1' - DB2 will not consider parallelism for queries
      - 'ANY' - DB2 will use parallelism for queries where possible
    - Default CURRENT DEGREE zparm in DSNTIP4 (CDSSRDEF = ANY)

- Migration: Rebind to have parallelism considered for static queries

- Query parallelism should be transparent to application:
  - same locking behavior
  - same error handling
I/O Parallelism (Behind the Scenes)

Optimization Strategy (prior to DB2 9... more on this later)

- Two-phase approach:
  1. Follow DB2's normal access path selection
  2. Parallelize any parts that would benefit -- identify parts of the access path to parallelize and determine degree of parallelism
    - Leaves parallel degree flexible to allow runtime adjustment

- Parallel degree determination to be covered later in this presentation

- I/O parallelism is not "easy" to achieve. More on this later.
I/O Parallelism (Behind the Scenes)

Influencing the Parallel Degree

- Redefine table space with more partitions
- Adjust partition ranges to achieve more balanced partitions
- Adjust buffer pool allocation thresholds - no effect on degree chosen at bind time, but might affect actual degree at run time.
- Maximum Degree ZPARM (MDEG), caps the degree of parallelism per group
  - Install panel or DSNTIJUZ: DSN6SPRM PARAMDEG=40
- SPRMPTH to disable parallelism for short running queries

For the daring -- Modify statistics from RUNSTATS - DB2 uses these catalog tables to determine degree:

- SYSTABLESPACE, SYSTABLES, SYSTABSTATS, SYSCOLUMNS, SYSINDEXES, SYSINDEXSTATS
CP Parallelism (Version 4)

Steps to improve performance

1. Empathize with the user
2. Understand requirement (reduce ET by half (2x))
3. Analyze accounting trace - CP and elapsed times
4. Understand access path

<table>
<thead>
<tr>
<th>ACCOUNTING TRACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP Time:</td>
</tr>
<tr>
<td>Elapsed Time:</td>
</tr>
</tbody>
</table>

We just got some new vending machines with fresh brewed coffee. I want this thing to run while I walk to the vending machines. It takes me 7-8 mins.

6. Determine bottleneck

I/O-intensive
Processor-bound
100% of 1 CP

No wait for processor

Processor- and I/O-bound!
Reduce ET by 2x: 15 mins -> 7.5 mins by opening up CP and I/O bottleneck (by doubling the number of partitions).
CP Parallelism (Version 4)

Physical Design

- CUSTNO index
- CUST table

Performance Data

- 4 partitions
- I/O-intensive
- I/O-bound
- 25% of 4 CPs

PLAN TABLE output from EXPLAIN

<table>
<thead>
<tr>
<th>ACCESS TYPE</th>
<th>PRE FETCH</th>
<th>ACCESS DEGREE</th>
<th>ACCESS_PGROUP_ID</th>
<th>PARALLEL_MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
<td>4</td>
<td>1</td>
<td>'C'</td>
</tr>
</tbody>
</table>

Performance Data

- CP Time: 15.9 mins
- Elapsed Time: 14.8 mins

Accounting Trace

Very little improvement?!?

Why?

We opened up the CP bottleneck, but not the I/O bottleneck. If we open up the I/O...
CP Parallelism (Version 4)

Physical Design

CUSTNO index

CUST table

Application

Need to rebind static plans/packages

Be aware that parallel tasks are created at OPEN CURSOR time

OPEN C1;

application stuff...

FETCH FROM C1;

PLAN_TABLE output from EXPLAIN

<table>
<thead>
<tr>
<th>ACCESS</th>
<th>PRE</th>
<th>ACCESS_</th>
<th>ACCESS_</th>
<th>PARALLEL_</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>FETCH</td>
<td>DEGREE</td>
<td>PGROUP_ID</td>
<td>MODE</td>
</tr>
<tr>
<td>R</td>
<td>S</td>
<td>8</td>
<td>1</td>
<td>'C'</td>
</tr>
</tbody>
</table>

DBA's View

Control priority of work

-DISPLAY THREAD:

- 16.32.57 -DB1G display thread(*)
- 16.32.57 STC00090 DSNV401I -DB1G DISPLAY THREAD REPORT FOLLOWS -
- 16.32.57 STC00090 DSNV402I -DB1G ACTIVE THREADS -

- NAME ST A REQ ID AUTHID PLAN ASID TOKEN
- BATCH T * 1 PUPPYDML ADMF001 DSNTEP3 0025 30
- PT * 549 PUPPYDML ADMF001 DSNTEP3 002A 38
- PT * 892 PUPPYDML ADMF001 DSNTEP3 002A 37
- PT * 47 PUPPYDML ADMF001 DSNTEP3 002A 36
- PT * 612 PUPPYDML ADMF001 DSNTEP3 002A 35
- PT * 545 PUPPYDML ADMF001 DSNTEP3 002A 34
- PT * 432 PUPPYDML ADMF001 DSNTEP3 002A 33
- PT * 443 PUPPYDML ADMF001 DSNTEP3 002A 32
- PT * 252 PUPPYDML ADMF001 DSNTEP3 002A 31

- DISPLAY ACTIVE REPORT COMPLETE
- 16.32.58 STC00090 DSN9022I -DB1G DSNVDT 'DISPLAY THREAD' NORMAL COMPLETION
CP Parallelism (Version 4)

Physical Design

CUSTNO index

CUST table

Performance Data

I/O-intensive
I/O-bound
50% of 4 CPs

PLAN_TABLE output from EXPLAIN

<table>
<thead>
<tr>
<th>ACCESS TYPE</th>
<th>PRE.FETCH</th>
<th>ACCESS DEGREE</th>
<th>ACCESS PGROUP_ID</th>
<th>PARALLEL MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
<td>8</td>
<td>1</td>
<td>'C'</td>
</tr>
</tbody>
</table>

Accounting Trace

CP Time: 16.1 mins
Elapsed Time: 7.5 mins

"That IT group is really responsive!"
CP Parallelism (Version 4)

System Requirements for CP Parallelism

- Multiple CPs on-line
  - With only one CP available, I/O parallelism can achieve the same elapsed time with less CP overhead
  - Checked at BIND or PREPARE time
  - If unavailable, I/O parallelism considered

CP parallelism is much easier to achieve since CP-intensive queries are now candidates for parallelism. CP parallelism is used for both I/O- and CP-intensive queries.
CP Parallelism (Behind the Scenes)

What kind of parallelism?

- Common Industry Terms: Symmetric MultiProcessing (SMP), Shared Memory Model

![Diagram of CP Parallelism]

Advantages of Shared Memory Model:
- Flexibility - CP not tied to a specific data partition
- Simple inter-task communication

Disadvantage:
- Upper bound to scalability or speedup - Ultimately the capacity of the machine will be reached
CP Parallelism (Behind the Scenes)

CP Parallelism - True Multiprocessing

- Multiple CPs, each working on one of the I/O streams allows for even greater throughput
- CP-intensive sorts and joins can now be partitioned and performed in parallel
- I/O-intensive queries also benefit because I/O streams are managed by a single task
CP Parallelism (Behind the Scenes)
Multi-Tasking - How does DB2 do it?

- Spawning parallel tasks: z/OS preemptable SRBs are used for work done in parallel. Originating Task (TCB) handles SRB creation, cleanup and data merging.

- Preemptable SRBs:
  - Synchronize originating and parallel tasks
  - Introduced with Enclave Services (MVS 5.2)
  - Inherit dispatching priority of allied address space. Therefore all work is done at the same priority (goodness)

- Originating task does not control scheduling or which CP an SRB is run on – z/OS handles scheduling.
- DB2 handles synchronization through suspending and resuming tasks
CP Parallelism (Behind the Scenes)

- Parallel tasks are started at OPEN CURSOR*

  Application might be able to take advantage of this to achieve inter-query parallelism:

  ```sql
  DECLARE CURSOR C1 FOR SELECT COUNT(*) FROM ORDERS WHERE INVOICE_AMT > 4000.00
  DECLARE CURSOR C2 FOR SELECT PARTNAME FROM PARTS WHERE INVENTORY_AMT > 200
  ```

  ![Diagram of parallel tasks](image)

  ```sql
  OPEN CURSOR C1
  OPEN CURSOR C2
  ... 
  ... 
  FETCH C1
  FETCH C2
  ```

  *Exception if RID sort, but no data sort, then //ism starts at first fetch (same as without //ism)
CP Parallelism (Behind the Scenes)

Parallel Degree Determination

"Why is the degree sometimes less than the number of parts?"

- Optimal degree for parallel group is determined at BIND/PREPARE time
  - Also called "Planned BIND degree" - shown in EXPLAIN output
- Optimal degree determined by considering:
  - Number of table space partitions
  - Estimated I/O cost of largest partition
  - Estimated CP cost considering:
    - Processing cost
    - MIPS rating of machine
    - Number of CPs on-line (used for CP parallelism only)
- Degree determination deferred if access path dependent on host variable
DB2 chooses the smallest degree that will still deliver the best possible elapsed time. With the shared data model, DB2 has the flexibility to choose the degree of parallelism.

I/O-intensive: Degree of parallelism approaches the number of partitions.

Processor-intensive: Degree of parallelism approaches the number of processors (as of DB2 9, times 4).

Additionally, skews in the data organization can be detected and compensated for in choosing the degree of parallelism.
CP Parallelism (Behind the Scenes)

Parallel Degree - Runtime Adjustment

- Degree chosen at BIND time may be adjusted at run time:
  - Host variable case: Degree determination was deferred at BIND time
  - CPs on-line is rechecked (CP //ism only), resulting degree is known as "Planned Run Degree"

- Buffer pool resource availability
  - If system already flooded with parallel tasks, run with a lower degree of parallelism

- Resulting degree is known as "Actual Degree"

- Planned Bind Degree, Planned Run Degree, and Actual Degree are shown in Performance Trace

- Statistics and accounting traces shows how often parallel degree is downgraded
CP Parallelism (Behind the Scenes)
Query Decomposition into Work Ranges

- DB2 uses "work ranges" to split a query access path into pieces to achieve parallelism on that access path.
  - By page range for a tablespace scan
  - By key range for an index scan
- Work ranges usually do not coincide with the physical partitions
- What is the degree of parallelism below?

Tablespace Scan Example:

**Originating Task**
SELECT CUSTOMER, REGION FROM CUST WHERE LASTVISIT < (CURRENT DATE - 180 DAYS) ORDER BY REGION;

**Parallel Tasks**
- SELECT CUST, REGION FROM CUST WHERE LASTVISIT < (CURRENT DATE - 180 DAYS) ORDER BY REGION;
- SELECT CUST, REGION FROM CUST WHERE LASTVISIT < (CURRENT DATE - 180 DAYS) ORDER BY REGION;
- SELECT CUST, REGION FROM CUST WHERE LASTVISIT < (CURRENT DATE - 180 DAYS) ORDER BY REGION;

**Work Ranges**
- Page Range -

**Physical Partitions**
- Page Numbers - Partitioning index is customer number:
CP Parallelism (Behind the Scenes)
Query Decomposition Example #1

Tablespace Scan:

```
SELECT EMPNO, LASTNAME FROM EMP WHERE DEPTNO = 'M92';
```

Single-table Sort:

```
SELECT LASTNAME, FIRSTNAME, HIREDATE FROM EMP ORDER BY HIREDATE;
```
CP Parallelism (Behind the Scenes)
Query Decomposition Example #2

```
SELECT * FROM TABLE1, TABLE2 WHERE TABLE1.COLX = TABLE2.COLY;
```

Diagram showing the process of parallelism with tables, workfiles, scanning, sorting, repartitioning into two workfiles, merging and joining.
CP Parallelism (Behind the Scenes)

**Tablespace Scan:**

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>ACCESTATYP</th>
<th>PREFETCH</th>
<th>ACCESSTYP</th>
<th>ACCESS DEGREE</th>
<th>ACCESS PGROUPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>1</td>
<td>0</td>
<td>R</td>
<td>S</td>
<td></td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

**Tablespace Scan with ORDER BY sort:**

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>ACCESS TYPE</th>
<th>SORTC ORDERBY</th>
<th>PREFETCH</th>
<th>ACCESS DEGREE</th>
<th>ACCESS PGROUPID</th>
<th>SORTC PGROUPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>1</td>
<td>0</td>
<td>R</td>
<td>N</td>
<td>S</td>
<td>4</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>44</td>
<td>2</td>
<td>3</td>
<td>Y</td>
<td></td>
<td></td>
<td>?</td>
<td>?</td>
<td>1</td>
</tr>
</tbody>
</table>

**Sort-Merge Join:**

<table>
<thead>
<tr>
<th>QRY#</th>
<th>PLAN#</th>
<th>TNAME</th>
<th>METH</th>
<th>ACCESTATYP</th>
<th>PRE-FETCH</th>
<th>ACCESS DEGREE</th>
<th>ACCESS PGROUP</th>
<th>SORTN PGROUP</th>
<th>OUTP GROUP</th>
<th>JOIN DEGREE</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1</td>
<td>T1</td>
<td>0</td>
<td>R</td>
<td>S</td>
<td>4</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>T2</td>
<td>2</td>
<td>R</td>
<td>S</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>

Visual Explain / Access Plan Graph shows more clearly.
CP Parallelism - Balancing I/O and CP

There's still room for improvement here. Now our bottleneck is on the I/O again. If we increase the number of I/O streams to the data, we should see improvements.

Steps to improve performance

1. Analyze accounting trace
   - CP and elapsed times

   **Accounting Trace**

<table>
<thead>
<tr>
<th>CP Time:</th>
<th>16.1 mins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed Time:</td>
<td>7.5 mins</td>
</tr>
</tbody>
</table>

2. Understand access path

   **PLAN TABLE output from EXPLAIN**

<table>
<thead>
<tr>
<th>ACCESS TYPE</th>
<th>PRE FETCH</th>
<th>ACCESS DEGREE</th>
<th>PARALLEL MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
<td>8</td>
<td>'C'</td>
</tr>
</tbody>
</table>

3. Determine bottleneck

   ![Diagram showing 8 partitions with I/O-intensive and I/O-bound]

   - 8 partitions
   - I/O-intensive
   - I/O-bound
   - 50% of 4 CPs

   I/O-bound!
   We should be able to reduce ET by 2x
   7.5 mins -> 3.75 mins by opening up the I/O bottleneck and doubling # of partitions.
CP Parallelism - Balancing I/O and CP

**Physical Design**

- CUSTNO index
- CUST table

**Application**

Bind or rebinding necessary for DB2 to consider a different degree of parallelism

For dynamic statements -- **NO CHANGES NECESSARY**

**DBA’s View**

Resource Limit Facility (RLF) controls

New RLFFUNC values of 3 (disable I/O //ism) and 4 (disable CP //ism):

<table>
<thead>
<tr>
<th>RLFFUNC</th>
<th>AUTHID</th>
<th>LU NAME</th>
<th>PLAN NAME</th>
<th>RLF COLLN</th>
<th>RLF PKG</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>QMFUSER</td>
<td></td>
<td>QMF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>HACKER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>HACKER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Performance Data**

- 16 partitions
- I/O-intensive Processor-bound
- I/O-bound
- 100% of 4 CPs

**PLAN_TABLE output from EXPLAIN**

<table>
<thead>
<tr>
<th>ACCESS TYPE</th>
<th>PRE_FETCH</th>
<th>ACCESS DEGREE</th>
<th>PARALLEL MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
<td>16</td>
<td>'C'</td>
</tr>
</tbody>
</table>

**Accounting Trace**

- CP Time: 16.3 mins
- Elapsed Time: 3.75 mins

QMFUSER is disabled from using CP Query Parallelism with PLAN QMF

HACKER is disabled from using any Query Parallelism

**End**

"I think it's time to give those IS guys a raise!"
CP Parallelism - Accounting Trace

- Separate accounting trace records cut for each parallel task, **-OR- you can tell DB2 to roll-up the information into one acctg trace record for the originating task.**

- Originating task and parallel tasks have the same correlation header information:

  **Originating Task Accounting Trace Record**

<table>
<thead>
<tr>
<th>ACE Address</th>
<th>Originating Task’s ACE Address</th>
<th>Number of parallel tasks created</th>
<th>CP Time</th>
<th>Getpages</th>
<th>Sequential Prefetch</th>
</tr>
</thead>
<tbody>
<tr>
<td>03AA2320</td>
<td>03AA2320</td>
<td>2</td>
<td>1:47</td>
<td>110</td>
<td>0</td>
</tr>
</tbody>
</table>

  **Parallel Task Accounting Trace Records**

<table>
<thead>
<tr>
<th>ACE Address</th>
<th>Originating Task’s ACE Address</th>
<th>Number of parallel tasks created</th>
<th>CP Time</th>
<th>Getpages</th>
<th>Sequential Prefetch</th>
</tr>
</thead>
<tbody>
<tr>
<td>04B29440</td>
<td>03AA2320</td>
<td>0</td>
<td>12:35</td>
<td>29765</td>
<td>3159</td>
</tr>
<tr>
<td>05C83460</td>
<td>03AA2320</td>
<td>0</td>
<td>12:43</td>
<td>29431</td>
<td>3236</td>
</tr>
</tbody>
</table>
# Query Parallelism - Performance Traces

IFCID 0221 (Class 8) - Issued for each parallel group

## Header Portion - shows information about degree determination

<table>
<thead>
<tr>
<th>PROGRAM NAME</th>
<th>STATEMENT NUMBER</th>
<th>QUERY BLOCK NUMBER</th>
<th>PARALLEL GROUP NUMBER</th>
<th>PLANNED BIND DEGREE</th>
<th>PLANNED RUN DEGREE</th>
<th>ACTUAL DEGREE</th>
<th>REASON FOR RUNTIME DEGREE</th>
<th>PARALLEL MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVT551</td>
<td>44</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>C</td>
</tr>
</tbody>
</table>

## Repeating Portion - shows logical partition info for each task

<table>
<thead>
<tr>
<th>LOGICAL PARTITION</th>
<th>LOW PAGE RANGE</th>
<th>HIGH PAGE RANGE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x'000000'</td>
<td>x'2000000'</td>
<td>NORMAL</td>
</tr>
<tr>
<td>2</td>
<td>x'200001'</td>
<td>x'4000000'</td>
<td>NORMAL</td>
</tr>
<tr>
<td>3</td>
<td>x'400001'</td>
<td>x'6000000'</td>
<td>NORMAL</td>
</tr>
<tr>
<td>4</td>
<td>x'600001'</td>
<td>x'8000000'</td>
<td>EMPTY</td>
</tr>
</tbody>
</table>

- Applies if host variables are used to determine degree
- Indicates that DB2 determined no qualifying data at runtime
- Repeating Portion - shows logical partition info for each task
### Query Parallelism - Performance Traces

IFCID 0222 (Class 8) - Issued for each parallel group

#### Header Portion - shows information about the pipe for this parallel group

<table>
<thead>
<tr>
<th>PROGRAM NAME</th>
<th>STATEMENT NUMBER</th>
<th>QUERY BLOCK NUMBER</th>
<th>PARALLEL GROUP NUMBER</th>
<th>PIPE CREATION TIME</th>
<th>PIPE TERMINATION TIME</th>
<th>ROWS PROCESSED</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVT551</td>
<td>44</td>
<td>1</td>
<td>1</td>
<td>16:21:41:33</td>
<td>16:23:59:44</td>
<td>1009345</td>
</tr>
</tbody>
</table>

#### Repeating Portion - shows information about each subpipe

<table>
<thead>
<tr>
<th>SUBPIPE</th>
<th>SUBPIPE CREATION TIME</th>
<th>SUBPIPE TERMINATION TIME</th>
<th>ROWS PROCESSED</th>
</tr>
</thead>
</table>

Repeating Portion - shows information about each subpipe
# Query Parallelism - Performance Traces

IFCID 0231 (Class 8) - Issued for each parallel group (not issued for I/O Parallelism)

Header Portion - shows information about the tasks in this parallel group

<table>
<thead>
<tr>
<th>Program Name</th>
<th>Statement Number</th>
<th>Query Block Number</th>
<th>Parallel Group Number</th>
<th>Group Creation Time</th>
<th>Group Termination Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV551</td>
<td>44</td>
<td>1</td>
<td>1</td>
<td>16:21:41:32</td>
<td>16:23:59:51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parallel Task</th>
<th>Task Creation Time</th>
<th>Task Termination Time</th>
<th>CP Seconds</th>
<th>DB2 Member</th>
<th>Service Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16:21:41.45</td>
<td>16:23:57.29</td>
<td>134.57</td>
<td>DB2A</td>
<td>2346894376</td>
</tr>
<tr>
<td>2</td>
<td>16:21:41.51</td>
<td>16:22:51.32</td>
<td>72.03</td>
<td>DB2A</td>
<td>2344762845</td>
</tr>
<tr>
<td>4</td>
<td>16:21:41.88</td>
<td>16:22:58.38</td>
<td>78.10</td>
<td>DB2A</td>
<td>2344843243</td>
</tr>
</tbody>
</table>

Added for Version 5

Repeating Portion - shows information about each parallel task
CPC Bottleneck

I want the query to run even faster. What can we do?

Steps to improve performance

1. Analyze accounting trace
   - CP and elapsed times

   **Accounting Trace**
   - CP Time: 16.3 mins
   - Elapsed Time: 3.75 mins

2. Understand access path

   **PLAN_TABLE output from EXPLAIN**
   - ACCESS TYPE: R
   - PRE FETCH: S
   - ACCESS DEGREE: 16
   - PARALLELMODE: 'C'

3. Determine bottleneck

   - 16 partitions
   - I/O-intensive
   - Processor-bound
   - I/O-bound
   - 100% of 4 CPs
   - Min of 16 lower paths

CP- and I/O-bound!

We should be able to reduce ET by 2x

3.75 mins -> 1.8 mins by opening up
the CP and the I/O bottlenecks.
Data Sharing Environment

Sysplex Timer

Coupling Facility

CPC with a DB2 and an IRLM
Sysplex Parallelism: Splitting the work

Coordinating DB2

-Assisting DB2s

#1 query

#2 query work

#3 create SRBs

partitioned table
Sysplex Parallelism: Returning results

-OR-

Workfiles

#1 data

#2 XCF links

#3 Answer Set

partitioned table
Sysplex Parallelism (Version 5)

**Physical Design**

- CUSTNO index
- CUST table

**Application**
If data affinity routing is used, be aware of creating inter-system read/write interest
For dynamic statements -- **NO CHANGES NECESSARY**

**DBA’s View**
See requirements and display gbpool slides

**Performance Data**

<table>
<thead>
<tr>
<th>PLAN_TABLE output from EXPLAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS TYPE</td>
</tr>
<tr>
<td>R</td>
</tr>
</tbody>
</table>

- VPSIZE = 1000 buffers
- VPSEQT = 80% of VPSIZE (800 buffers)
- VPPSEQT = 50% of VPSEQT (400 buffers)
- VPXPSEQT = 50% of VPPSEQT (200 buffers)

**Accounting Trace**

- CP Time: 17.2 mins
- Elapsed Time: 1.8 mins

**End User**
"I don't even have time to go get my coffee!"
Sysplex Parallelism Install parms

Outbound Control

Inbound Control

COORDINATOR

'NO' disables this DB2 member from sending query work to other DB2 members.

Used to "fence" off an individual DB2 member from sending work to other DB2 members.

ASSISTANT

Specify whether this DB2 is allowed to assist a parallelism coordinator with parallel processing.

If 'NO', this DB2 is not considered as an assistant at either bind or run time. If 'YES', this DB2 is considered.

Checked at both bind and run time.

Press: ENTER to continue            HELP

DSNTIPK   INSTALL DB2 - DEF GRP OR MBR

Copyright by IBM Corp.

Check parameters and reenter to change:

1  GROUP NAME   ===>
2  MEMBER NAME  ===>
3  WORK FILE DB ===>
4  GROUP ATTACH ===>
5  COORDINATOR  ====> NO
6  ASSISTANT    ==> NO

'NO' disables this DB2 member from sending query work to other DB2 members.

Used to "fence" off an individual DB2 member from sending work to other DB2 members.

Checked at both bind and at runtime.
Sysplex Parallelism BPool

Example:
- ALTER BUFFERPOOL(BPn) VPSIZE(1000) VPSEQT(80) VPPSEQT(50) VPXPSEQT(50)

VPSIZE = 1000 buffers
VPSEQT = 80% of VPSIZE (800 buffers)
VPPSEQT = 50% of VPSEQT (400 buffers)
VPXPSEQT = 50% of VPPSEQT (200 buffers)

Examined by Coordinating DB2
Only examined by Assisting DB2s

Runtime Inbound Control
Sysplex parallelism

Monitoring and Tuning

- Improving Response Time (affected by the following)
  - CP Contention
  - Buffer Pool Availability
  - I/O Contention
  - **XCF Links availability**
    - Defined with CTC (channel-to-channel) or CF Links
    - Defining both to XCF is recommended
Sysplex parallelism monitoring & tuning

-DISPLAY THREAD with Sysplex Query Parallelism:

- 17.08.44 .display thread(*)
- 17.08.44 STC00090 DSNV401I . DISPLAY THREAD REPORT FOLLOWS -
- 17.08.44 STC00090 DSNV402I . ACTIVE THREADS -
- NAME ST A REQ ID  AUTHID PLAN ASID TOKEN
- BATCH T 1 PUPPYDML ADMF001 DSNTEP3 0025 30
- PT 612 PUPPYDML ADMF001 DSNTEP3 002A 35
- PT 545 PUPPYDML ADMF001 DSNTEP3 002A 34
- PT 432 PUPPYDML ADMF001 DSNTEP3 002A 33
- PT 443 PUPPYDML ADMF001 DSNTEP3 002A 32
- PT 252 PUPPYDML ADMF001 DSNTEP3 002A 31
- DISPLAY ACTIVE REPORT COMPLETE
- 17.08.45 STC00090 DSN9022I . DSNVDT 'DISPLAY THREAD' NORMAL COMPLETION
- 17.10.12 <v42d display thread(*)
- 17.10.12 STC00044 DSNV401I <V42D DISPLAY THREAD REPORT FOLLOWS -
- 17.10.12 STC00044 DSNV402I <V42D ACTIVE THREADS -
- NAME ST A REQ ID  AUTHID PLAN ASID TOKEN
- BATCH PT 641 PUPPYDML ADMF001 DSNTEP3 002D 10
- V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- BATCH PT 72 PUPPYDML ADMF001 DSNTEP3 002D 9
- V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- BATCH PT 549 PUPPYDML ADMF001 DSNTEP3 002D 8
- V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- BATCH PT 892 PUPPYDML ADMF001 DSNTEP3 002D 7
- V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- DISPLAY ACTIVE REPORT COMPLETE

-DISPLAY GROUP DETAIL:

00- 12.55.36 .display group detail
- 12.55.36 STC00042 DSN7100I . DSN7GCMD
- *** BEGIN DISPLAY OF GROUP(DSNCAT ) GROUPLEVEL(420)
- DB2 SYSTEM IRLM
- MEMBER ID SUBSYS CMDPREF STATUS NAME LVL SUBSYS IRLMPROC
- V42A 1 V42A  .  ACTIVE MVSA 420 AR21 ARLM21
- V42B 2 V42B <V42B  ACTIVE MVSB 420 BR21 BRLM21
- V41C 3 V41C <V41C  ACTIVE MVSC 410 CRLM CRLM21
- V42D 4 V42D <V42D  FAILED MVSD 420 DR21 DRLM21
- V42E 5 V42E <V42E  QUIESCED MVSE 420 ER21 ERLM21
- V42F 6 V42F <V42F  ACTIVE MVSF 420 FR21 FRLM21
- V42G 7 V42G <V42G  ACTIVE MVSG 420 GR21 GRLM21
- DB2 PARALLEL PARALLEL
- MEMBER COORDINATOR ASSISTANT
- V42A YES NO
- V42B YES YES
- V41C ***** *****
- V42D ***** *****
- V42E ***** *****
- V42F NO YES
- V42G NO NO
- DISPLAY GROUP DETAIL COMPLETE
- 12.55.36 STC00042 DSN9022I . DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION

---

DISPLAY THREAD with Sysplex Query Parallelism:

- 17.08.44 .display thread(*)
- 17.08.44 STC00090 DSNV401I . DISPLAY THREAD REPORT FOLLOWS -
- 17.08.44 STC00090 DSNV402I . ACTIVE THREADS -
- NAME ST A REQ ID  AUTHID PLAN ASID TOKEN
- BATCH T 1 PUPPYDML ADMF001 DSNTEP3 0025 30
- PT 612 PUPPYDML ADMF001 DSNTEP3 002A 35
- PT 545 PUPPYDML ADMF001 DSNTEP3 002A 34
- PT 432 PUPPYDML ADMF001 DSNTEP3 002A 33
- PT 443 PUPPYDML ADMF001 DSNTEP3 002A 32
- PT 252 PUPPYDML ADMF001 DSNTEP3 002A 31
- DISPLAY ACTIVE REPORT COMPLETE
- 17.08.45 STC00090 DSN9022I . DSNVDT 'DISPLAY THREAD' NORMAL COMPLETION
- 17.10.12 <v42d display thread(*)
- 17.10.12 STC00044 DSNV401I <V42D DISPLAY THREAD REPORT FOLLOWS -
- 17.10.12 STC00044 DSNV402I <V42D ACTIVE THREADS -
- NAME ST A REQ ID  AUTHID PLAN ASID TOKEN
- BATCH PT 641 PUPPYDML ADMF001 DSNTEP3 002D 10
- V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- BATCH PT 72 PUPPYDML ADMF001 DSNTEP3 002D 9
- V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- BATCH PT 549 PUPPYDML ADMF001 DSNTEP3 002D 8
- V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- BATCH PT 892 PUPPYDML ADMF001 DSNTEP3 002D 7
- V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- DISPLAY ACTIVE REPORT COMPLETE

-DISPLAY GROUP DETAIL:

00- 12.55.36 .display group detail
- 12.55.36 STC00042 DSN7100I . DSN7GCMD
- *** BEGIN DISPLAY OF GROUP(DSNCAT ) GROUPLEVEL(420)
- DB2 SYSTEM IRLM
- MEMBER ID SUBSYS CMDPREF STATUS NAME LVL SUBSYS IRLMPROC
- V42A 1 V42A  .  ACTIVE MVSA 420 AR21 ARLM21
- V42B 2 V42B <V42B  ACTIVE MVSB 420 BR21 BRLM21
- V41C 3 V41C <V41C  ACTIVE MVSC 410 CRLM CRLM21
- V42D 4 V42D <V42D  FAILED MVSD 420 DR21 DRLM21
- V42E 5 V42E <V42E  QUIESCED MVSE 420 ER21 ERLM21
- V42F 6 V42F <V42F  ACTIVE MVSF 420 FR21 FRLM21
- V42G 7 V42G <V42G  ACTIVE MVSG 420 GR21 GRLM21
- DB2 PARALLEL PARALLEL
- MEMBER COORDINATOR ASSISTANT
- V42A YES NO
- V42B YES YES
- V41C ***** *****
- V42D ***** *****
- V42E ***** *****
- V42F NO YES
- V42G NO NO
- DISPLAY GROUP DETAIL COMPLETE
- 12.55.36 STC00042 DSN9022I . DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION

---
Sysplex Parallelism - Accounting Trace

- Unlike query CP parallelism, these trace records will be cut on different DB2 members
- Correlation provided by way of LUWID

### Originating Task Accounting Trace Record

<table>
<thead>
<tr>
<th>Member Name</th>
<th>ACE Address</th>
<th>Query Originating Member</th>
<th>Originating Task’s ACE Address</th>
<th>Number of parallel tasks created</th>
<th>LUWID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2A</td>
<td>03AA2320</td>
<td>DB2A</td>
<td>03AA2320</td>
<td>4</td>
<td>‘NETID.LUNAME.x.y’</td>
</tr>
</tbody>
</table>

### Parallel Task Accounting Trace Records

<table>
<thead>
<tr>
<th>Member Name</th>
<th>ACE Address</th>
<th>Query Originating Member</th>
<th>Originating Task’s ACE Address</th>
<th>Number of parallel tasks created</th>
<th>LUWID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2A</td>
<td>04B29440</td>
<td>DB2A</td>
<td>03AA2320</td>
<td>0</td>
<td>‘NETID.LUNAME.x.y’</td>
</tr>
<tr>
<td>DB2A</td>
<td>05C83460</td>
<td>DB2A</td>
<td>03AA2320</td>
<td>0</td>
<td>‘NETID.LUNAME.x.y’</td>
</tr>
<tr>
<td>DB2B</td>
<td>06D29480</td>
<td>DB2A</td>
<td>03AA2320</td>
<td>0</td>
<td>‘NETID.LUNAME.x.y’</td>
</tr>
<tr>
<td>DB2C</td>
<td>07E200C0</td>
<td>DB2A</td>
<td>03AA2320</td>
<td>0</td>
<td>‘NETID.LUNAME.x.y’</td>
</tr>
</tbody>
</table>

OMEGAMON for DB2 will gather information across the Sysplex in order to give a "one thread" view of the accounting trace data (for batch processing only)
Sysplex parallelism - RLF

The local DB2s ASUTIME is used to govern the parallel tasks
Query Parallelism Modes

**V3**
I/O Parallelism
Single Execution Unit
Multiple I/O streams

TCB (originating task)
partitioned table

**V4**
CP Parallelism
Multiple Execution Units
Each having a single I/O stream

SRBs (parallel tasks)
partitioned table

**V5**
Sysplex Query Parallelism

**V5**

[Diagram showing various components and their interactions related to query parallelism]
What happens to I/O & CP parallelism?

I/O:
- DB2 will always prefer CP parallelism rather than I/O parallelism
- I/O Parallelism is not used "underneath" CP parallel tasks
- No mixture of I/O parallel and CP Parallel groups under the same statement
- Cases where I/O parallelism is still used:
  - Running on single CP system
  - Dynamic Queries - Use Resource Limit Facility (RLF) function
  - Ambiguous cursor with CURRENTDATA(YES) and ISOLATION(CS)

CP:
- DB2 will always prefer Sysplex parallelism rather than CP parallelism
- To force DB2 to choose CP parallelism instead of Sysplex parallelism:
  - Set COORDINATOR = "N", or
  - Set all ASSISTANT = "N"
  - Set VPXPSEQT on all other DB2s to zero
  - Use the new RLF function (dynamic queries only)
- Cases where CP parallelism is still used:
  - Static queries in migrated plans
  - ISOLATION(RR) or (RS)
  - Star join query
  - RID access or IN-list parallelism
  - Sparse index used
"The moving bottleneck"

<table>
<thead>
<tr>
<th>Bottle neck</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I/O</strong></td>
<td>Typically, the greatest bottleneck for long-running queries has been the time needed to simple get the data from the storage device. Despite advances such as prefetch I/O streams and various flavors of caching, the CP processing speed has still been much faster than the I/O path.</td>
<td>Query I/O Parallelism</td>
</tr>
<tr>
<td><strong>CP</strong></td>
<td>With parallel I/O streams, the portion of the total elapsed time that is devoted to I/O activities is reduced to such an extent that the CP cost becomes the next concern. Parallelizing these operations further reduces the elapsed time.</td>
<td>Query CP Parallelism</td>
</tr>
<tr>
<td><strong>CPC</strong></td>
<td>With parallel execution units each processing their own I/O stream, the CP resource again can become the bottleneck for queries that are very CP intensive (joins, sorts, etc.). By scheduling execution units across multiple DB2 (CPCs), CP intensive queries can see dramatic elapsed time reductions.</td>
<td>Sysplex Query Parallelism</td>
</tr>
</tbody>
</table>
So you’ve installed a zIIP

• And you aren’t utilizing it fully

• So how do you get more work to run on zIIP?
  • You could execute more distributed transactions
  • You could run more REORGs

• OR………. 
How to fully utilize your zIIP

- You could tune your SQL to increase parallelism

- Parallel child tasks obtain higher % redirect than DRDA
  - Applies to local or distributed
    - Local non-parallel obtains 0% redirect
    - Distributed non-parallel obtains x% redirect
    - Parallel obtains x++% redirect
      - *Except first “x” milliseconds*
Parallel Query zIIP Redirect Processing

- Applicable to the complex parallel queries
  - Portion of the child task processing will be redirected after certain CPU usage threshold has exceeded
    - Main tasks coming in via DRDA via TCP/IP can take advantage of the DRDA use of zIIP.

- The combined child & main tasks coming in through DRDA via TCP/IP is expected to yield additional processing eligible for zIIP.

- Longer running queries see higher benefit.

- The possible benefit to a data warehousing application may vary significantly depending on the characteristics of the application.
IBM Benchmarks – V8 vs 9

- Two internal Star Schema workloads
  - Existing Star Join workload increased to 87% zIIP eligible

<table>
<thead>
<tr>
<th></th>
<th>DB2 V8</th>
<th>DB2 9</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Elapse time (seconds)</td>
<td>21320</td>
<td>12620</td>
<td>41%</td>
</tr>
<tr>
<td>Total CPU time (seconds)</td>
<td>6793</td>
<td>5351</td>
<td>21%</td>
</tr>
<tr>
<td>CPU time eligible for zIIP</td>
<td>4756 (70%)</td>
<td>4676 (87%)</td>
<td></td>
</tr>
</tbody>
</table>

- New complex query workload increased to 90% zIIP eligible

<table>
<thead>
<tr>
<th></th>
<th>DB2 V8</th>
<th>DB2 9</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Elapse Time (seconds)</td>
<td>71660</td>
<td>8544</td>
<td>88%</td>
</tr>
<tr>
<td>Total CPU time (seconds)</td>
<td>7400</td>
<td>7514</td>
<td>-1.5%</td>
</tr>
<tr>
<td>CPU time eligible for zIIP</td>
<td>2924 (39.5%)</td>
<td>6775 (90%)</td>
<td></td>
</tr>
</tbody>
</table>
Parallelism plan determination changed in DB2 9

- In V8
  - Lowest cost is BEFORE parallelism
- In DB2 9
  - Lowest cost is AFTER parallelism
    - Only a subset of plans are considered for parallelism

How to parallelize these plans?

One Lowest cost plan survives
Performance

- I/O-intensive queries
  - All parallelism types significantly reduce the elapsed time of I/O-intensive queries
  - Additional processing power does not significantly decrease elapsed time of I/O-intensive queries

**Single-thread Only!**

- CP-intensive queries
  - Only multi-tasking significantly reduces the elapsed time of processor-bound queries
  - Once the processing power of a single-CPC is fully utilized, elapsed time cannot be reduced further

**Sysplex Query Parallelism!**

<table>
<thead>
<tr>
<th></th>
<th>Number of CPs</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O Parallelism</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CP Parallelism</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sysplex Query //ism</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Number of CPs</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CP Parallelism</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sysplex Query //ism</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parallelism Performance

**Query Speed-up**

<table>
<thead>
<tr>
<th>Response Time</th>
<th>Sequential</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>103</td>
<td>42</td>
</tr>
<tr>
<td>1,000</td>
<td>2,024</td>
<td>226</td>
</tr>
<tr>
<td>2,000</td>
<td>6,562</td>
<td>4,800</td>
</tr>
<tr>
<td>3,000</td>
<td>29X</td>
<td>114X</td>
</tr>
<tr>
<td>4,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7,000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 20 Engines, 1TB Database

Speed-up is not limited to the number of CP engines
Query parallelism has been implemented in stages across releases of DB2.

Query parallelism provides parallel processing to both processor and I/O-intensive read-only queries within a single DB2 and within the DB2 data sharing group while incurring minimal system overhead.

It reduces the elapsed time of long running queries by taking advantage of resources available.
Bryan F. Smith  bfsmith@us.ibm.com
IBM

Session Code: 8361
Query Parallelism in DB2 for z/OS