# 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;
```

# Cast of Characters

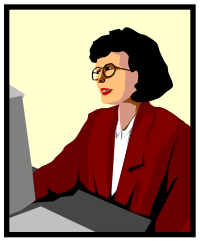
I use the following scenario to illustrate concepts of query parallelism. It does not imply a method of how you should implement your database design. Here is the cast of characters:

***PLAN_TABLE output from EXPLAIN***

| ACCESSTYPE | PREFETCH |
|---|---|

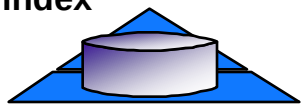
***Application***

***End User***

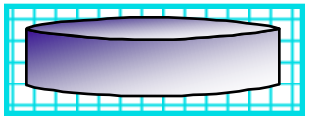


***Physical Design***

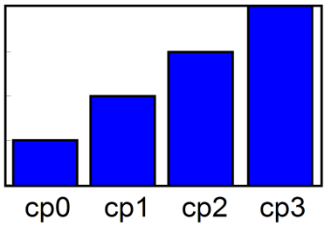
**index**



**table**



***Performance Data***



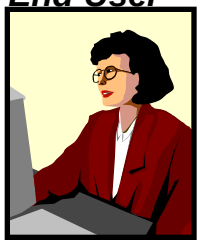

***DBA/SysProg View***

***Accounting Trace***

| CP Time: | xx mins |
|---|---|
| Elapsed Time: | yy mins |

# Sequential Processing

***End User***



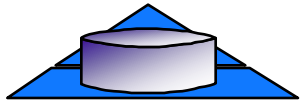
"Show me which customers I should send information to..."

***Application***

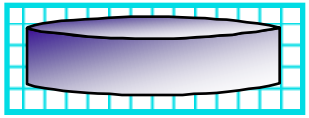
```
SELECT CUSTOMER, REGION
  FROM CUST
  WHERE LASTVISIT < (CURRENT DATE - 180 DAYS);
```

***Physical Design***

CUSTNO index (clustering)



CUST table



***DBA/SysProg View***

REORGs
RUNSTATS
Buffer pool tuning
Dataset placement
Work priority

***PLAN_TABLE output from EXPLAIN***

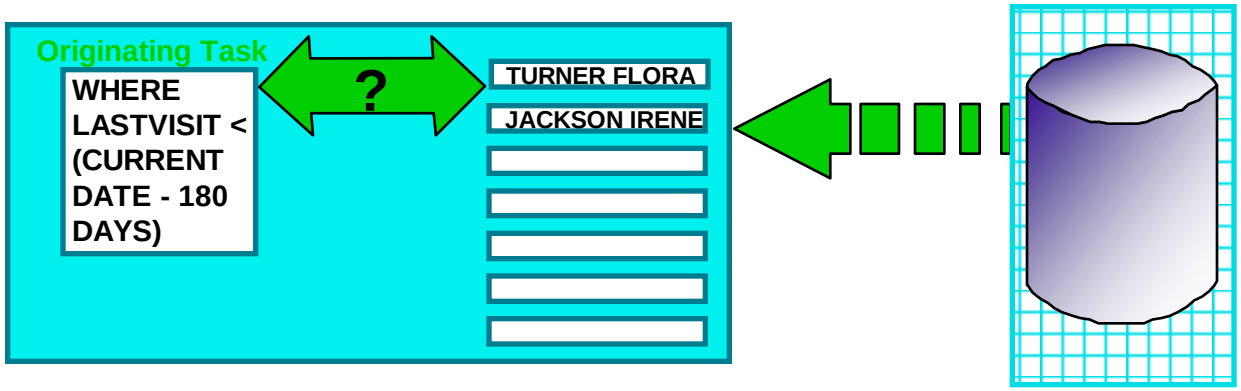
| ACCESSTYPE | PREFETCH |
|---|---|
| R | S |

***PLAN_TABLE output from EXPLAIN***

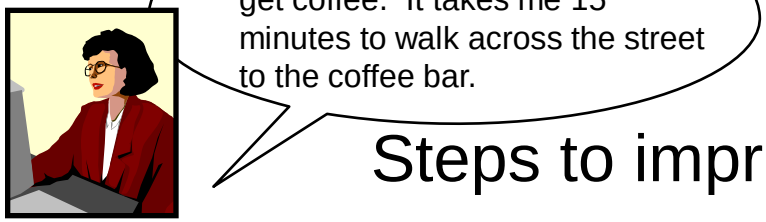
| ACCESSTYPE | PREFETCH |
|---|---|
| R | S |

Table space ("R"elational) scan

Sequential Prefetch - a form of parallelism that allows concurrent CPU and I/O processing



# "It's not fast enough!"




## 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***

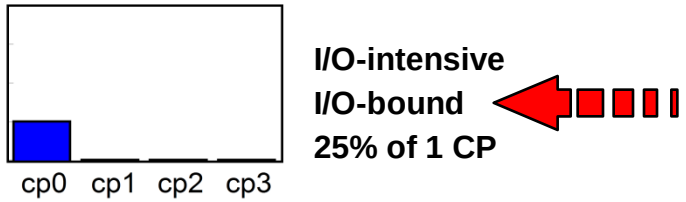
| ACCESSTYPE | PREFETCH |
|---|---|
| R | S |

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!):

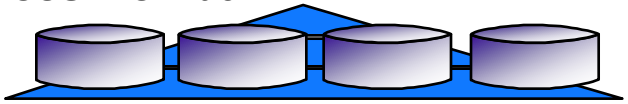



We should be able to reduce ET by 4x
60 mins -> 15 by opening up I/O bottleneck.

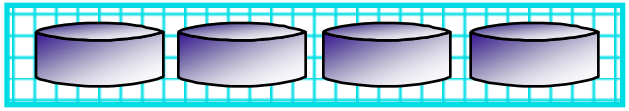
# I/O Parallelism (introduced in Version 3)

***Physical Design***

**CUSTNO index**



**CUST table**



***Application***

```
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***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | ACCESS_ PGROUP_ID |
|---|---|---|---|
| R | S | 4 | 1 |

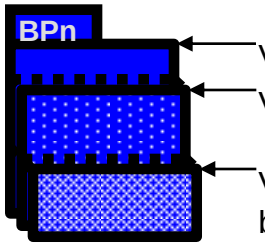
Parallel Degree

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.

***DBA's View***




VPSIZE = 1000 buffers

VPSEQT = 80% of VPSIZE (800 buffers)

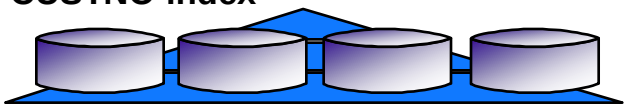
VPPSEQT = 50% of VPSEQT (400 buffers)

- 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.

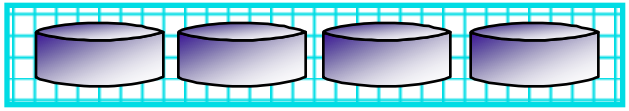
# I/O Parallelism (Version 3)

***Physical Design***

**CUSTNO index**



**CUST table**



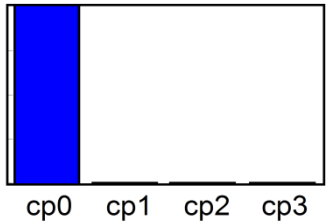
***Accounting Trace***

| | |
|---|---|
| CP Time: | 15.37 mins |
| Elapsed Time: | 15.5 mins |

**2.5% overhead**

***Performance Data***

No wait for processor →




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

***PLAN_TABLE output from EXPLAIN***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | ACCESS_ PGROUP_ID |
|---|---|---|---|
| R | S | 4 | 1 |

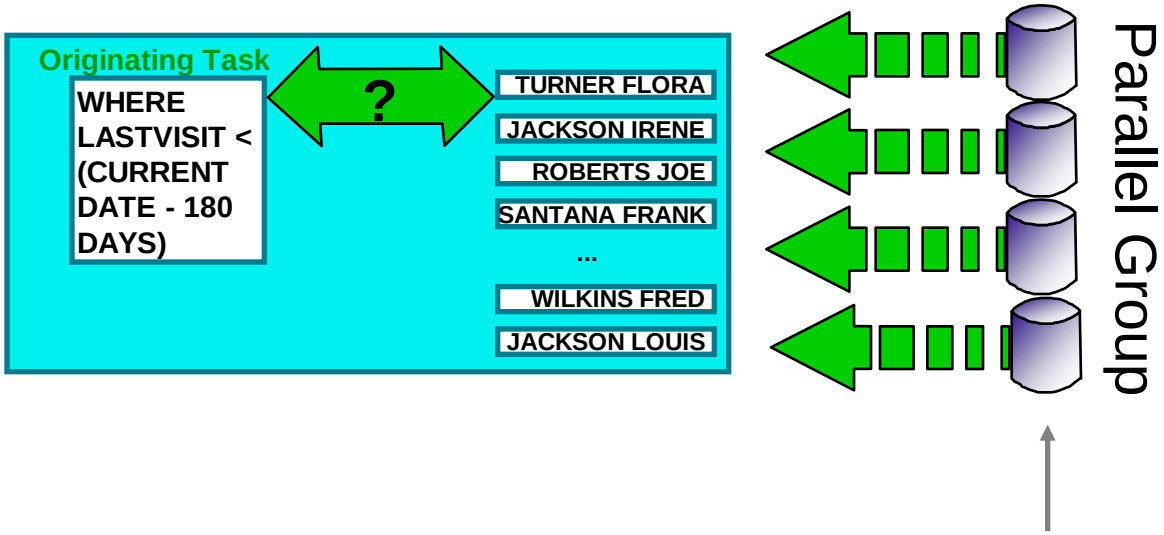
"Wow, it's done and my coffee is still warm!"

***End User***

# 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



# 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)

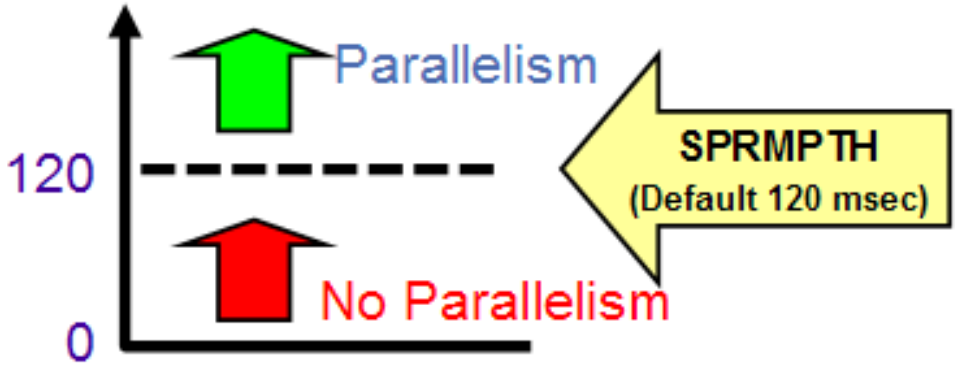
Optimization Strategy (prior to DB2 9… more on this later)

- 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)


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.

## 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

***Accounting Trace***

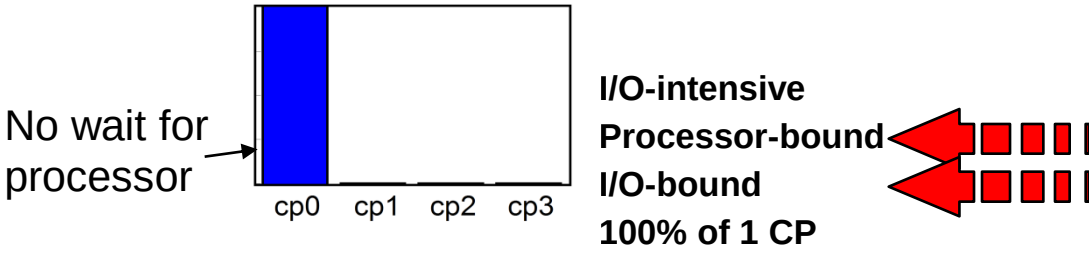
| | |
|---|---|
| CP Time: | 15.37 mins |
| Elapsed Time: | 15.5 mins |

4. Understand access path

***PLAN_TABLE output from EXPLAIN***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | ACCESS_ PGROUP_ID |
|---|---|---|---|
| R | S | 4 | 1 |

6. Determine bottleneck



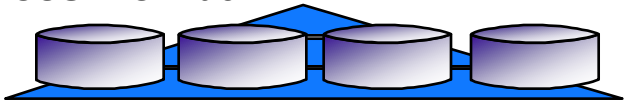

Processor- and I/O-bound!

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

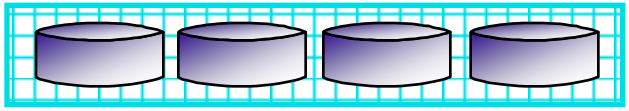
# CP Parallelism (Version 4)

***Physical Design***

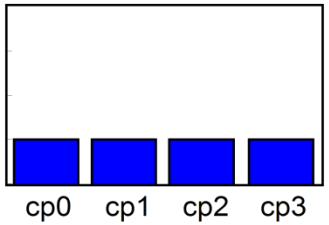
**CUSTNO index**



**CUST table**



***Performance Data***




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

**Added in Version 4:**
**'I' for I/O parallelism**
**'C' for CP parallelism**

***PLAN_TABLE output from EXPLAIN***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | ACCESS_ PGROUP_ID | PARALLEL_ MODE |
|---|---|---|---|---|
| R | S | 4 | 1 | 'C' |

***Accounting Trace***

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

Very little improvement?!?

Why?

We opened up the CP bottleneck,

but not the I/O bottleneck. If we open up the I/O...

Parallel Degree

The number of parallel **tasks** in a parallel group.

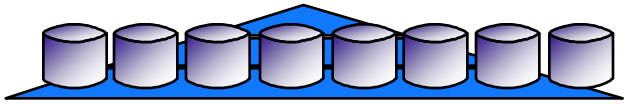
Parallel Group

A group of **tasks** dedicated to performing a specific portion of a query's access path.

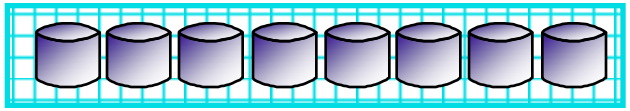
# CP Parallelism (Version 4)

***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***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | ACCESS_ PGROUP_ID | PARALLEL_ MODE |
|---|---|---|---|---|
| R | S | 8 | 1 | 'C' |

**DBA's View**

**Control priority of work**

**-DISPLAY THREAD:**

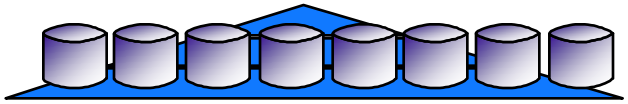
```
- 16.32.57           -DB1G display thread(*)
- 16.32.57 STC00090  DSNV401I -DB1G DISPLAY THREAD REPORT FOLLOWS -
- 16.32.57 STC00090  DSNV402I -DB1G ACTIVE THREADS -
- NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
- BATCH    T  *     1 PUPPYDML     ADMF001  DSNTEP3  0025    30
-           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)

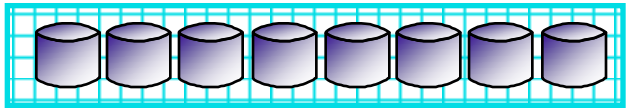
SHARE
Technology • Connections • Results

***Physical Design***

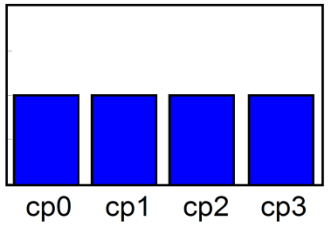
**CUSTNO index**



**CUST table**



***Performance Data***




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

***PLAN_TABLE output from EXPLAIN***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | ACCESS_ PGROUP_ID | PARALLEL_ MODE |
|---|---|---|---|---|
| R | S | 8 | 1 | 'C' |

***Accounting Trace***

| | |
|---|---|
| CP Time: | 16.1 mins |
| Elapsed Time: | 7.5 mins |

***End***

"That IT group is really responsive!"

# CP Parallelism (Version 4)

## System Requirements for CP Parallelism

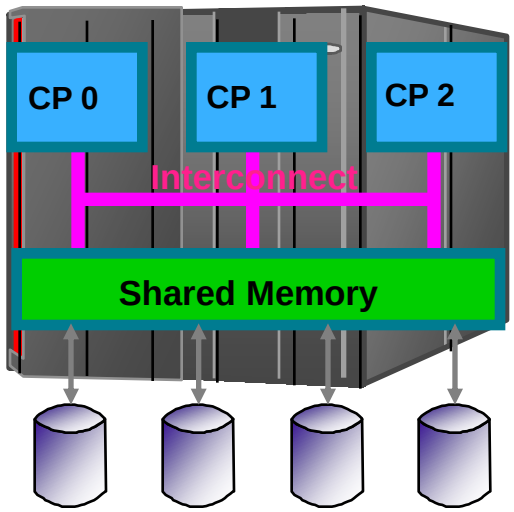
- Multiple CPs on-line
  - With only one CP available, I/O parallelism can achieve the same elapsed time with less CP overhead
  - Checked at BIND or PREPARE time
  - If unavailable, I/O parallelism considered

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




**Each CP can access the same memory and all of the data**

Advantages of Shared Memory Model:

Flexibility - CP not tied to a specific data partition

Simple inter-task communication

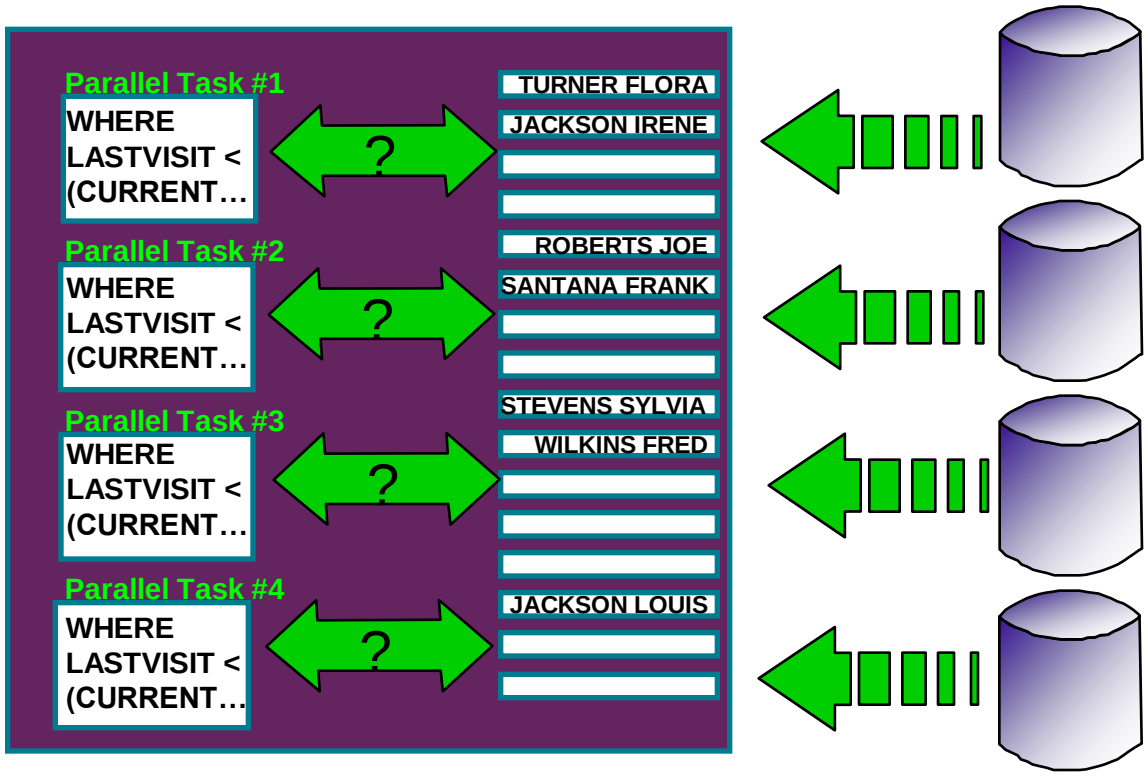
Disadvantage:

Upper bound to scalability or speedup - Ultimately the capacity of the machine will be reached

# CP Parallelism (Behind the Scenes)

## CP Parallelism - True Multiprocessing

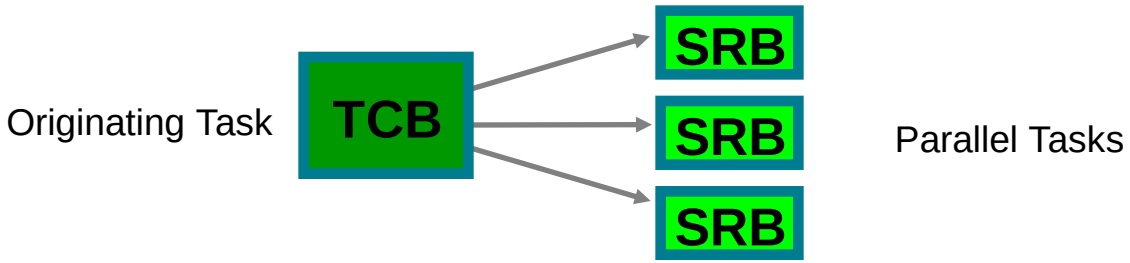
- Multiple CPs, each working on one of the I/O streams allows for even greater throughput
- CP-intensive sorts and joins can now be partitioned and performed in parallel
- I/O-intensive queries also benefit because I/O streams are managed by a single task



# 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.



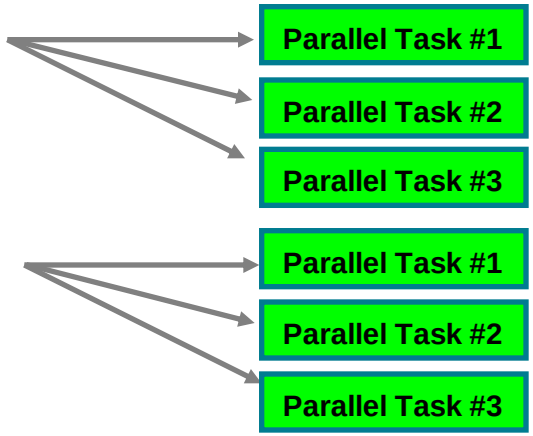

- Preemptable SRBs:
  - Synchronize originating and parallel tasks
  - Introduced with Enclave Services (MVS 5.2)
  - Inherit dispatching priority of allied address space. Therefore all work is done at the same priority (goodness)
- Originating task does not control scheduling or which CP an SRB is run 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:

```
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

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

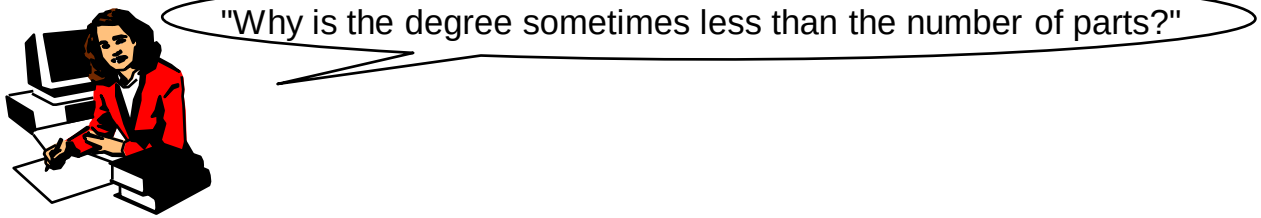





*Exception if RID sort, but no data sort, then //ism starts at first fetch (same as without //ism)

# CP Parallelism (Behind the Scenes)

## Parallel Degree Determination




- 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

# Determining the degree of parallelism

- 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

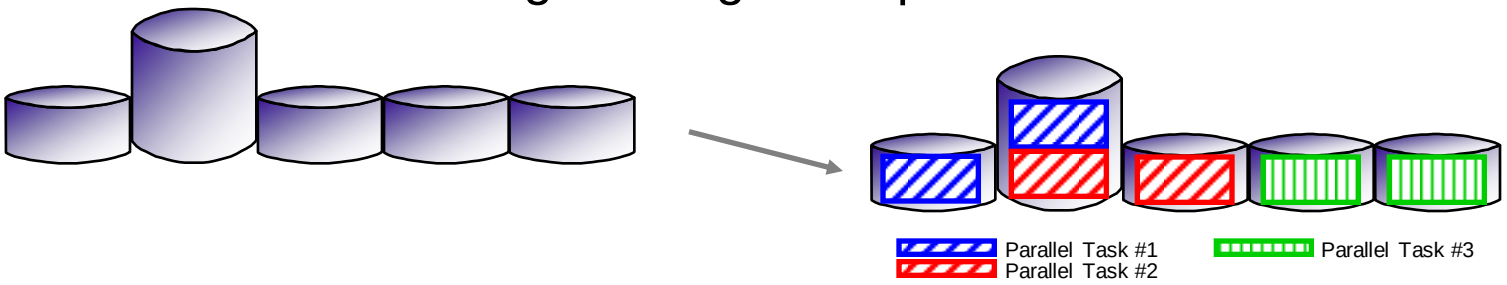
  I/O-intensive: Degree of parallelism approaches the number of partitions

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

  CP CP CP CP CP CP CP CP CP CP X 4 (as of DB2 9)

  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

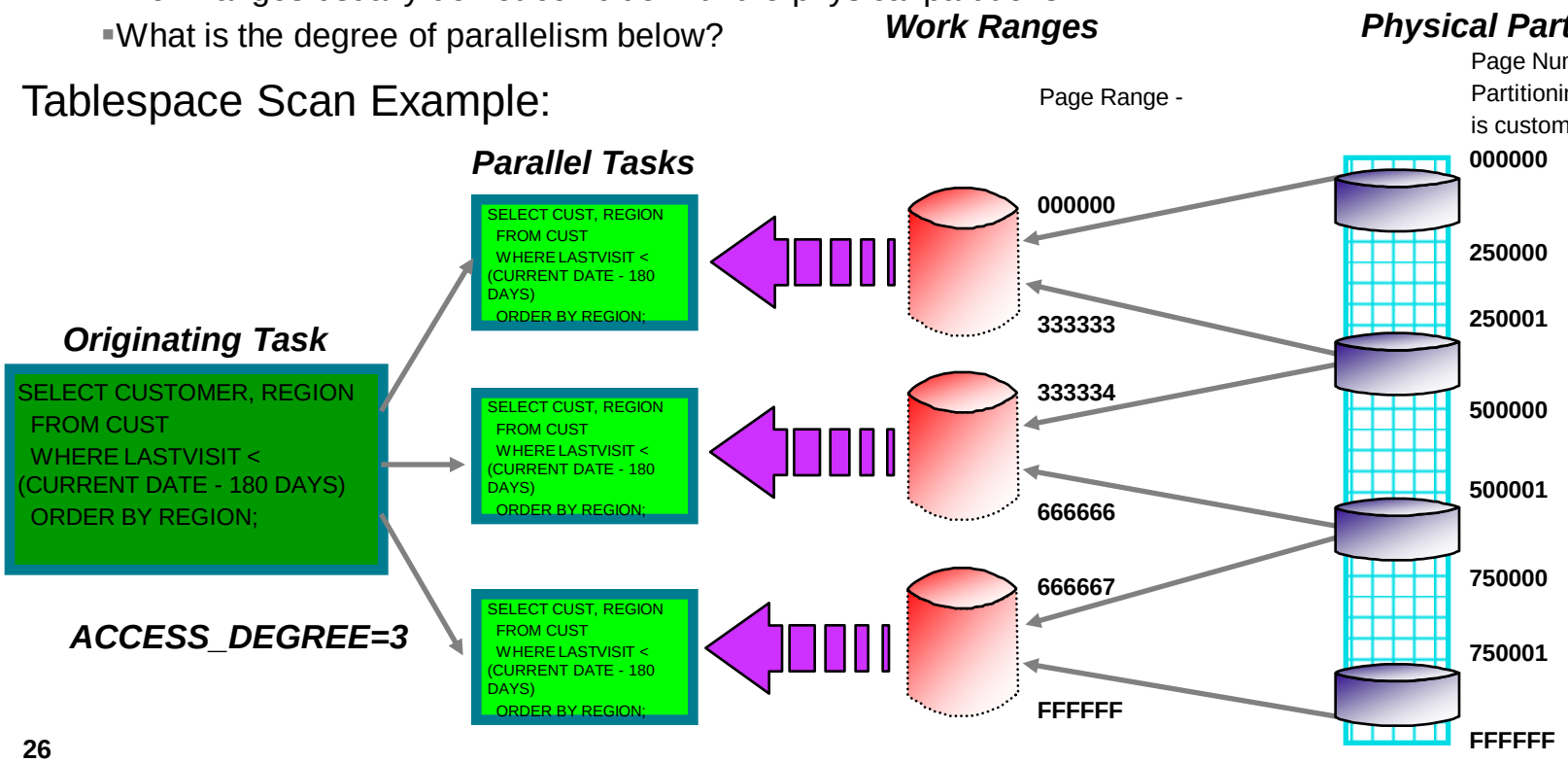
- Degree chosen at BIND time may be adjusted at run time:
  - Host variable case: Degree determination was deferred at BIND time
  - CPs on-line is rechecked (CP //ism only), resulting degree is known as "Planned Run Degree"
- Buffer pool resource availability
  - If system already flooded with parallel 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:

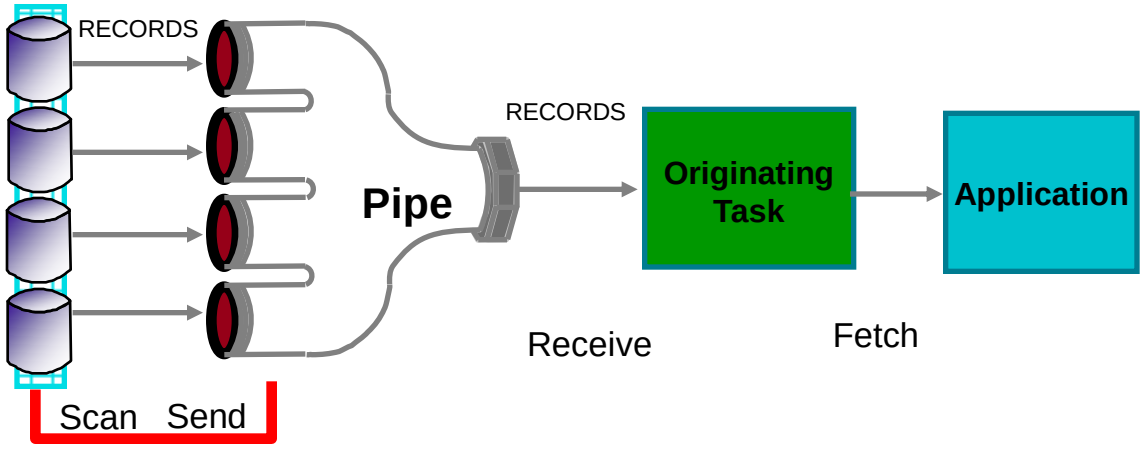


# CP Parallelism (Behind the Scenes)

## Query Decomposition Example #1

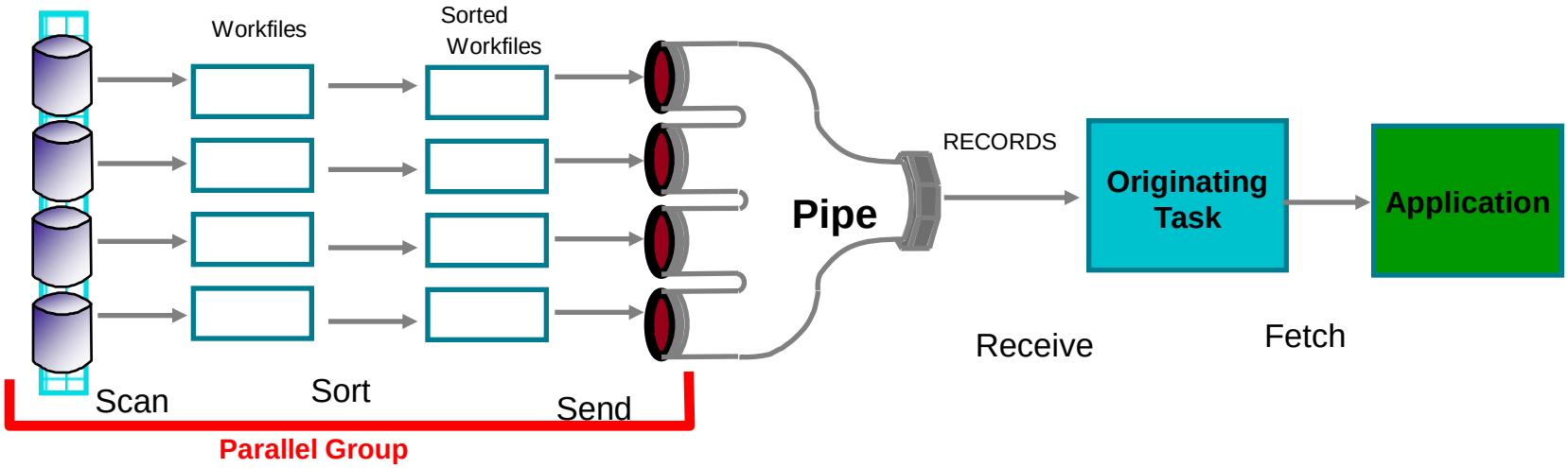
Tablespace Scan:

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

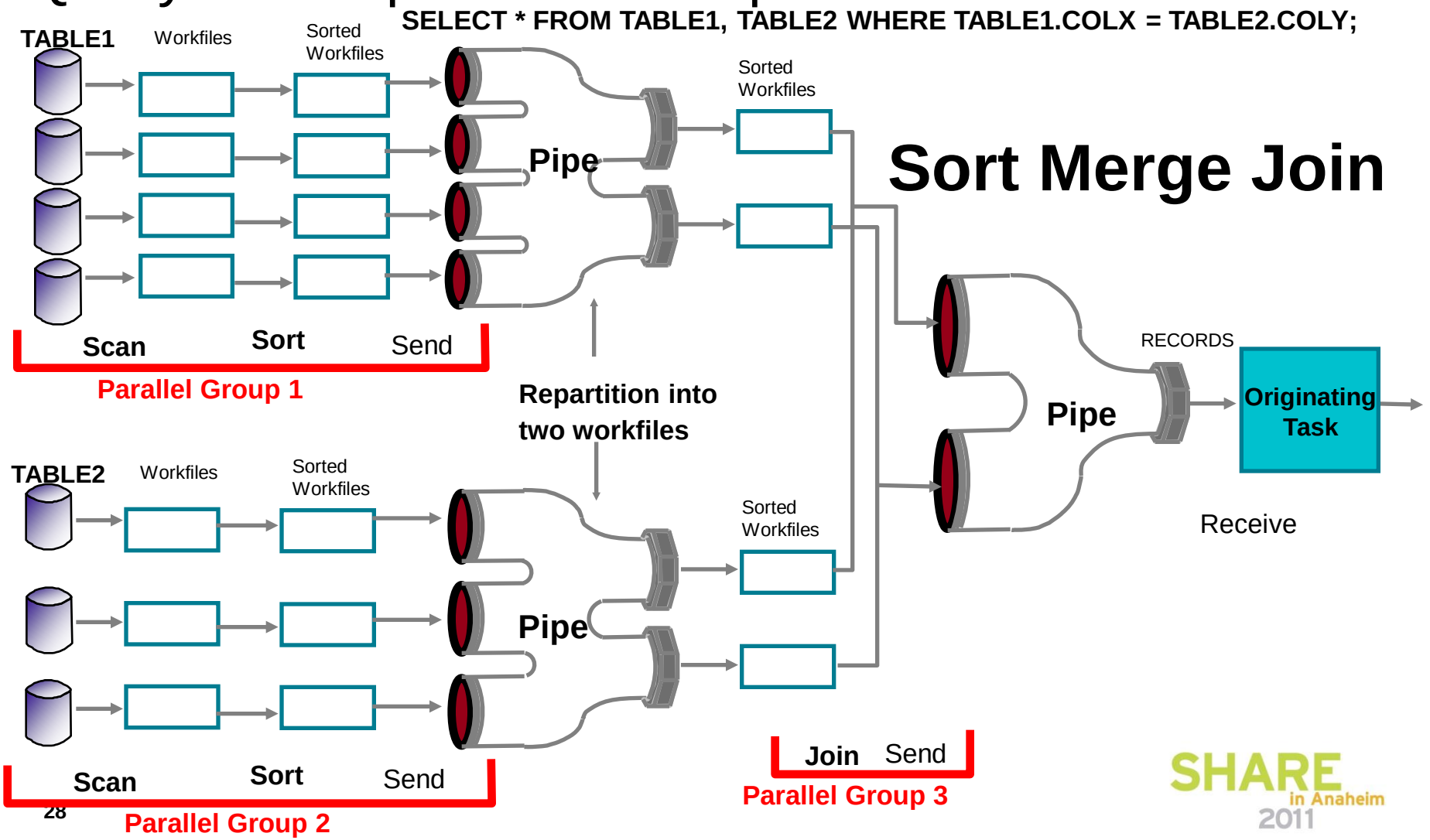



Single-table Sort:

**SELECT LASTNAME, FIRSTNAME, HIREDATE FROM EMP ORDER BY HIREDATE;**



# CP Parallelism (Behind the Scenes)
## Query Decomposition Example #2



# CP Parallelism (Behind the Scenes)

Tablespace Scan:

| QUERYNO | PLANNO | METHOD | ACCESSTYP | PREFETCH | ACCESS_ DEGREE | ACCESS_ PGROUPID |
|---|---|---|---|---|---|---|
| 39 | 1 | 0 | R | S | 4 | 1 |

Tablespace Scan with ORDER BY sort:

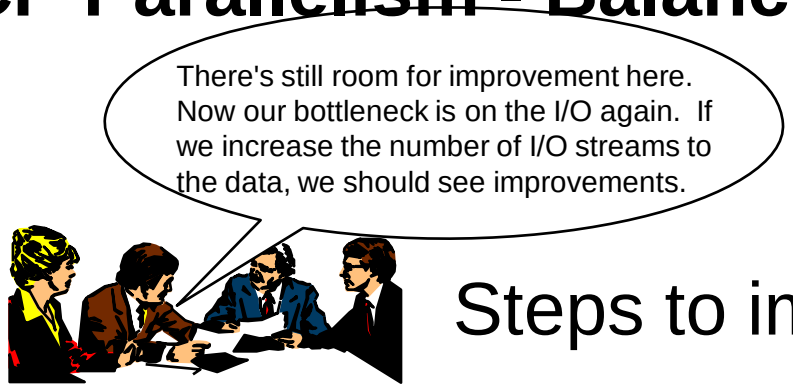
| QUERYNO | PLANNO | METHOD | ACCESS TYPE | SORTC_ ORDERBY | PRE-FETCH | ACCESS_ DEGREE | ACCESS_ PGROUPID | SORTC_ PGROUPID |
|---|---|---|---|---|---|---|---|---|
| 44 | 1 | 0 | R | N | S | 4 | 1 | ? |
| 44 | 2 | 3 | | Y | | ? | ? | 1 |

Sort-Merge Join:

| QRY# | PLAN# | TNAME | METH | ACC ESS TYP | PRE-FETCH | ACCESS_ DEGREE | ACCESS_ PGROUP | SORTN_ PGROUP | SORTC_ PGROUP | JOIN_ PGROUP | JOIN_ DEGREE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 1 | T1 | 0 | R | S | 4 | 1 | ? | ? | ? | ? |
| 50 | 2 | T2 | 2 | R | S | 3 | 2 | 2 | 1 | 3 | 2 |

Visual Explain / Access Plan Graph shows more clearly

# CP Parallelism - Balancing I/O and CP




## Steps to improve performance

1. Analyze accounting trace
   - CP and elapsed times

***Accounting Trace***

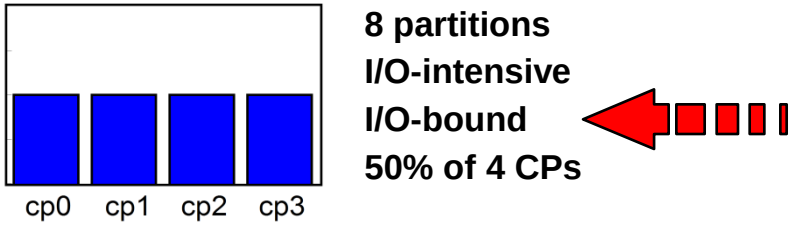
| | |
|---|---|
| CP Time: | 16.1 mins |
| Elapsed Time: | 7.5 mins |

2. Understand access path

***PLAN_TABLE output from EXPLAIN***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | PARALLEL MODE |
|---|---|---|---|
| R | S | 8 | 'C' |

3. Determine bottleneck




I/O-bound!

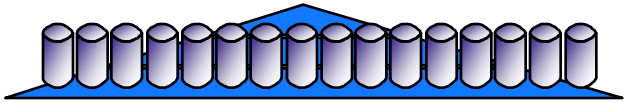
We should be able to reduce ET by 2x 7.5 mins -> 3.75 mins by opening up the I/O bottleneck and doubling # of partitions.

# CP Parallelism - Balancing I/O and CP

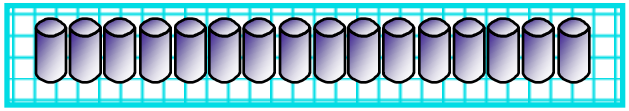
SHARE
Technology • Connections • Results

***Physical Design***

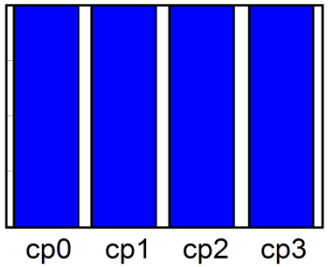
**CUSTNO index**



**CUST table**



***Performance Data***




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

***Application***

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

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

***PLAN_TABLE output from EXPLAIN***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | PARALLEL_ MODE |
|---|---|---|---|
| R | S | 16 | 'C' |

***Accounting Trace***

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

***DBA's View***

Resource Limit Facility (RLF) controls

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

| RLFFUNC | AUTHID | LU NAME | PLAN NAME | RLF COLLN | RLF PKG |
|---|---|---|---|---|---|
| 4 | QMFUSER | | QMF | | |
| 3 | HACKER | | | | |
| 4 | HACKER | | | | |

**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.** → **DSN6SYSP PTASKROL=YES**
- Originating task and parallel tasks have the same correlation header information:

**Originating Task Accounting Trace Record**

| ACE Address | Originating Task's ACE Address | Number of parallel tasks created | CP Time | Getpages | Sequential Prefetch |
|---|---|---|---|---|---|
| 03AA2320 | 03AA2320 | 2 | 1:47 | 110 | 0 |

...

**Parallel Task Accounting Trace Records**

| ACE Address | Originating Task's ACE Address | Number of parallel tasks created | CP Time | Getpages | Sequential Prefetch |
|---|---|---|---|---|---|
| 04B29440 | 03AA2320 | 0 | 12:35 | 29765 | 3159 |
| 05C83460 | 03AA2320 | 0 | 12:43 | 29431 | 3236 |

...

# Query Parallelism - Performance Traces

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

**Header Portion - shows information about degree determination**

| PROGRAM NAME | STATEMENT NUMBER | QUERY BLOCK NUMBER | PARALLEL GROUP NUMBER | PLANNED BIND DEGREE | PLANNED RUN DEGREE | ACTUAL DEGREE | REASON FOR RUNTIME DEGREE | PARALLEL MODE |
|---|---|---|---|---|---|---|---|---|
| INVT551 | 44 | 1 | 1 | 6 | 6 | 4 | 0 | C |

**Applies if host variables are used to determine degree**

| LOGICAL PARTITION | LOW PAGE RANGE | HIGH PAGE RANGE | STATUS |
|---|---|---|---|
| 1 | x'000000' | x'200000' | NORMAL |
| 2 | x'200001' | x'400000' | NORMAL |
| 3 | x'400001' | x'600000' | NORMAL |
| 4 | x'600001' | x'800000' | EMPTY |

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

**Indicates that DB2 determined no qualifying data at runtime**

# Query Parallelism - Performance Traces

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

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

| PROGRAM NAME | STATEMENT NUMBER | QUERY BLOCK NUMBER | PARALLEL GROUP NUMBER | PIPE CREATION TIME | PIPE TERMINATION TIME | ROWS PROCESSED |
|---|---|---|---|---|---|---|
| INVT551 | 44 | 1 | 1 | 16:21:41:33 | 16:23:59:44 | 1009345 |

| SUBPIPE | SUBPIPE CREATION TIME | SUBPIPE TERMINATION TIME | ROWS PROCESSED |
|---|---|---|---|
| 1 | 16:21:41:34 | 16:23:59:31 | 301002 |
| 2 | 16:21:41:34 | 16:23:59:37 | 399543 |
| 3 | 16:21:41:34 | 16:23:59:43 | 367889 |
| 4 | 16:21:41:34 | 16:23:59:49 | 340911 |

**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**

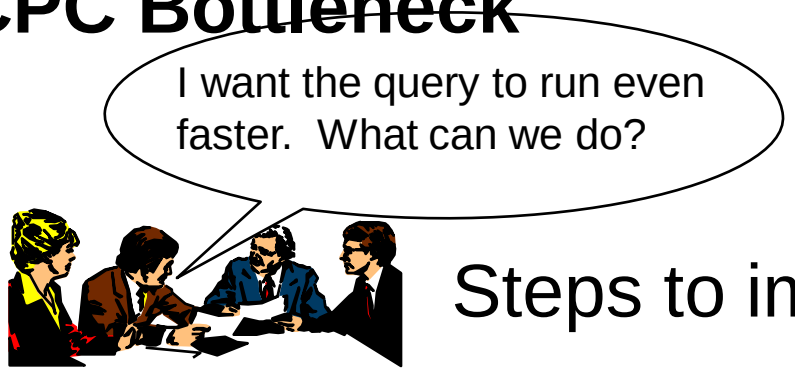
| PROGRAM NAME | STATEMENT NUMBER | QUERY BLOCK NUMBER | PARALLEL GROUP NUMBER | GROUP CREATION TIME | GROUP TERMINATION TIME |
|---|---|---|---|---|---|
| INVT551 | 44 | 1 | 1 | 16:21:41:32 | 16:23:59:51 |

Added for Version 5

| PARALLEL TASK | TASK CREATION TIME | TASK TERMINATION TIME | CP SECONDS | DB2 MEMBER | SERVICE UNITS |
|---|---|---|---|---|---|
| 1 | 16:21:41.45 | 16:23:57.29 | 134.57 | DB2A | 2346894376 |
| 2 | 16:21:41.51 | 16:22:51.32 | 72.03 | DB2A | 2344762845 |
| 3 | 16:21:41.59 | 16:23:21.52 | 101.27 | DB2A | 2346831853 |
| 4 | 16:21:41.88 | 16:22:58.38 | 78.10 | DB2A | 2344843243 |

**Repeating Portion -shows information about each parallel task**

# CPC Bottleneck




## Steps to improve performance

1. Analyze accounting trace
   - CP and elapsed times

***Accounting Trace***

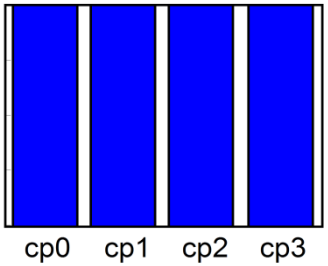
| | |
|---|---|
| CP Time: | 16.3 mins |
| Elapsed Time: | 3.75 mins |

2. Understand access path

***PLAN_TABLE output from EXPLAIN***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | PARALLEL_ MODE |
|---|---|---|---|
| R | S | 16 | 'C' |

3. Determine bottleneck



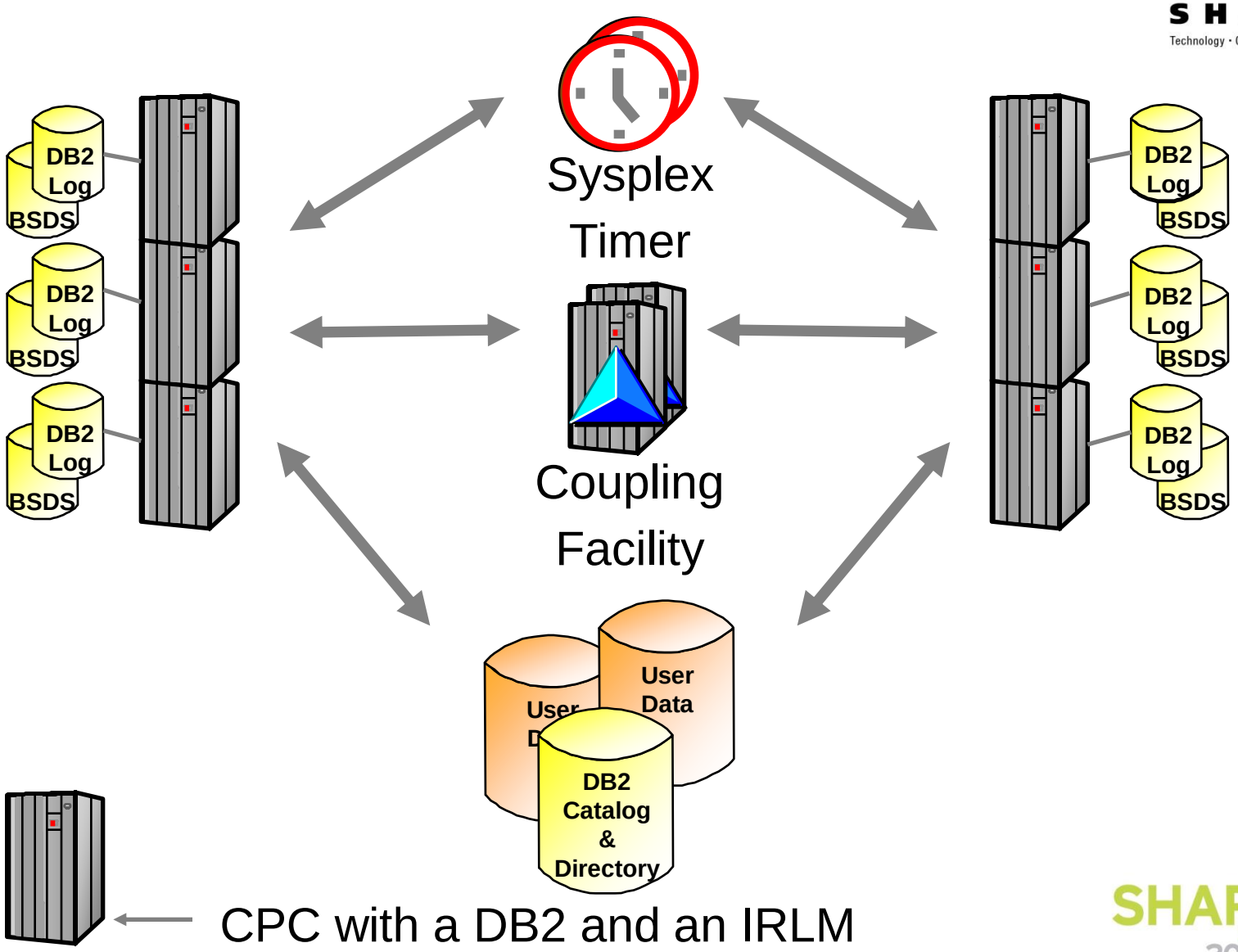

**16 partitions**
**I/O-intensive**
**Processor-bound**
**I/O-bound**
**100% of 4 CPs**
**Min of 16 lower paths**

CP- and I/O-bound!

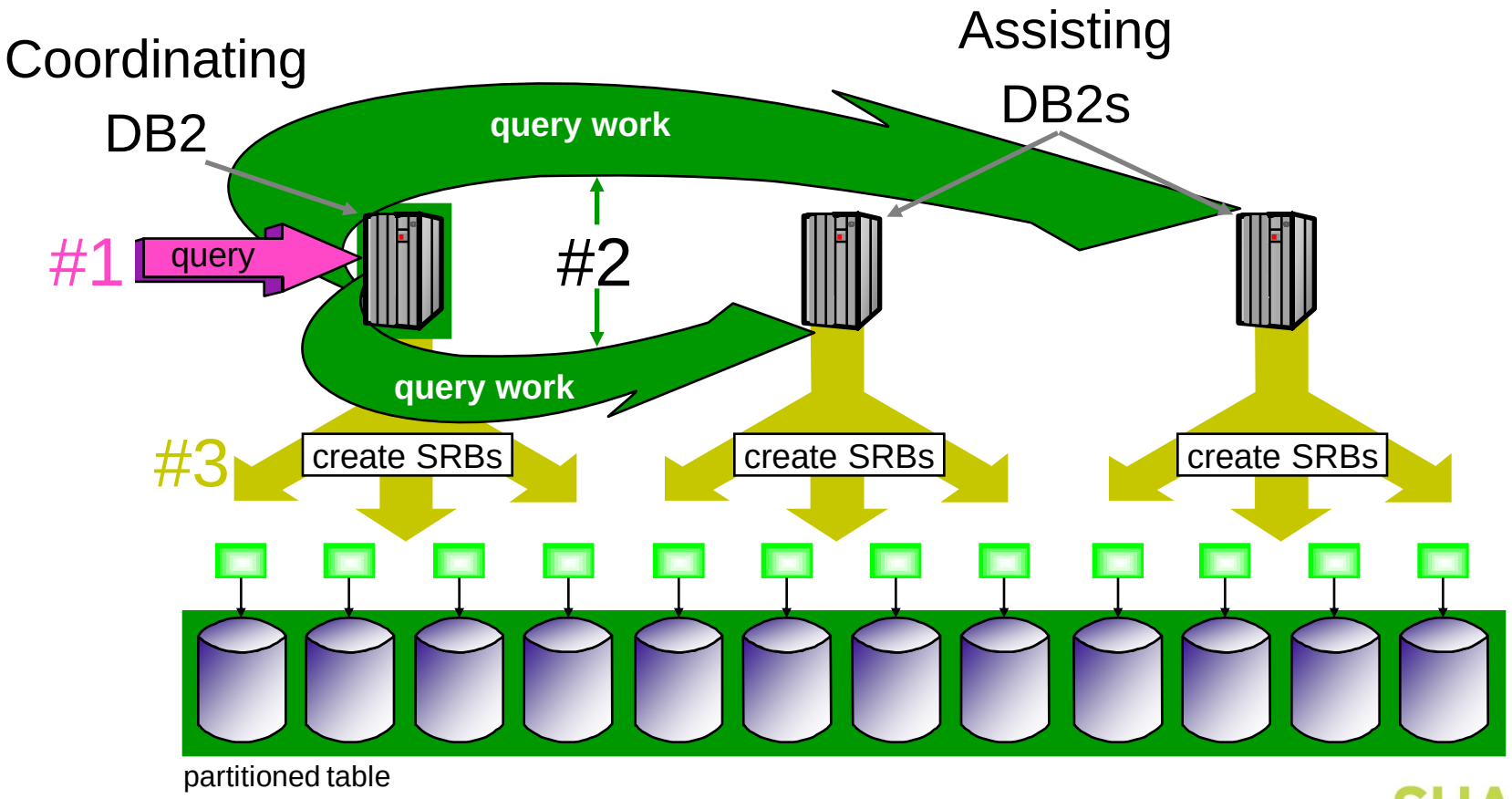
We should be able to reduce ET by 2x

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

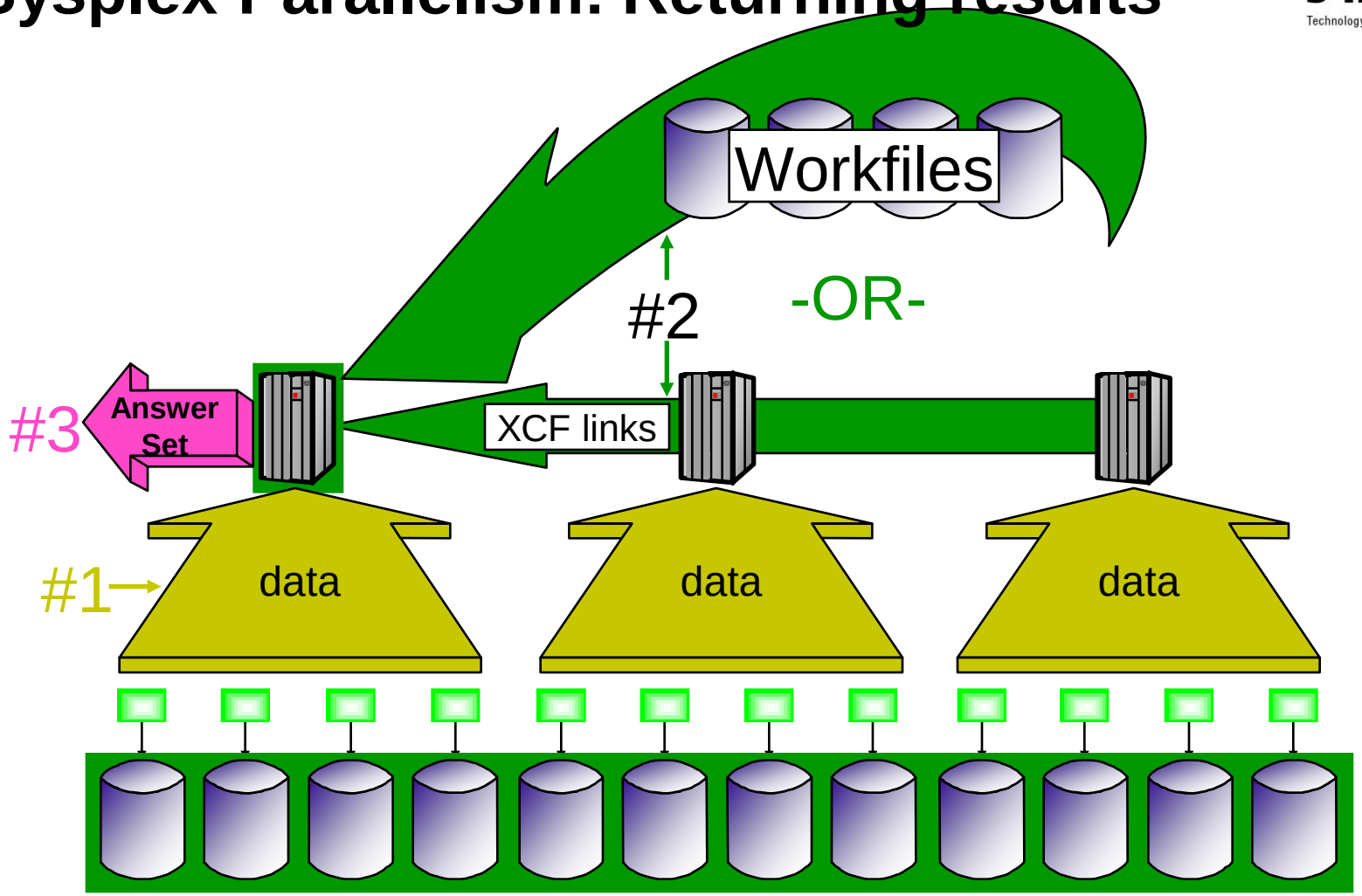
# Data Sharing Environment



# Sysplex Parallelism: Splitting the work



# Sysplex Parallelism: Returning results

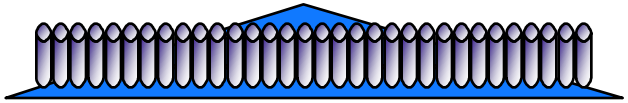


# Sysplex Parallelism (Version 5)

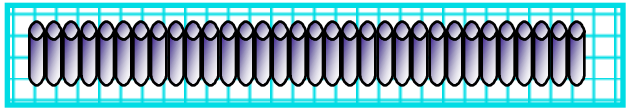
SHARE
Technology · Connections · Results

***Physical Design***

**CUSTNO index**



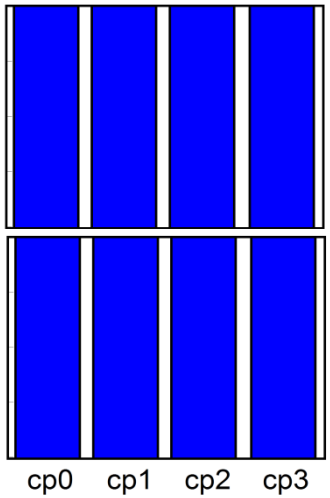
**CUST table**



***PLAN_TABLE output from EXPLAIN***

| ACCESS TYPE | PRE FETCH | ACCESS_ DEGREE | PARALLEL_ MODE |
|---|---|---|---|
| R | S | 32 | 'X' |

***Performance Data***




**32 partitions**
**Balanced**
**Processor-bound**
**if 2 CPCs**
**Not bound if**
**3 CPCs**

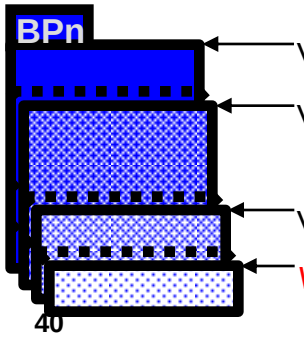
***Application***

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




VPSIZE = 1000 buffers

VPSEQT = 80% of VPSIZE (800 buffers)

VPPSEQT = 50% of VPSEQT (400 buffers)

*VPXPSEQT = 50% of VPPSEQT (200 buffers)*

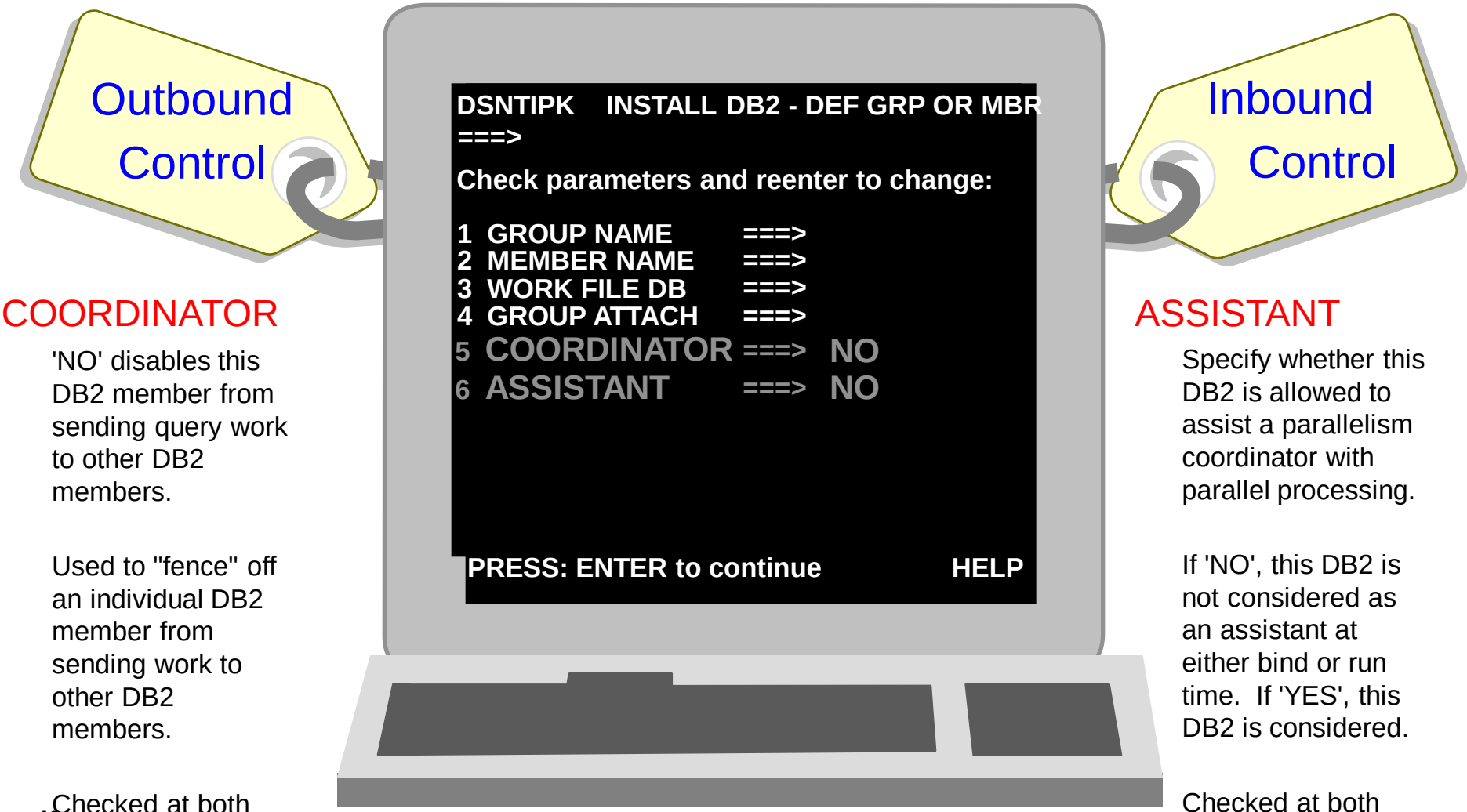
***Accounting Trace***

| | |
|---|---|
| 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




## 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.

Checked at both bind and at runtime.

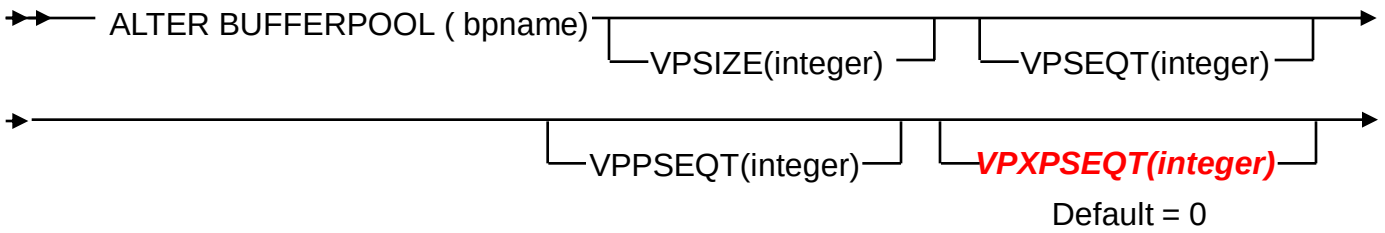
## ASSISTANT

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.

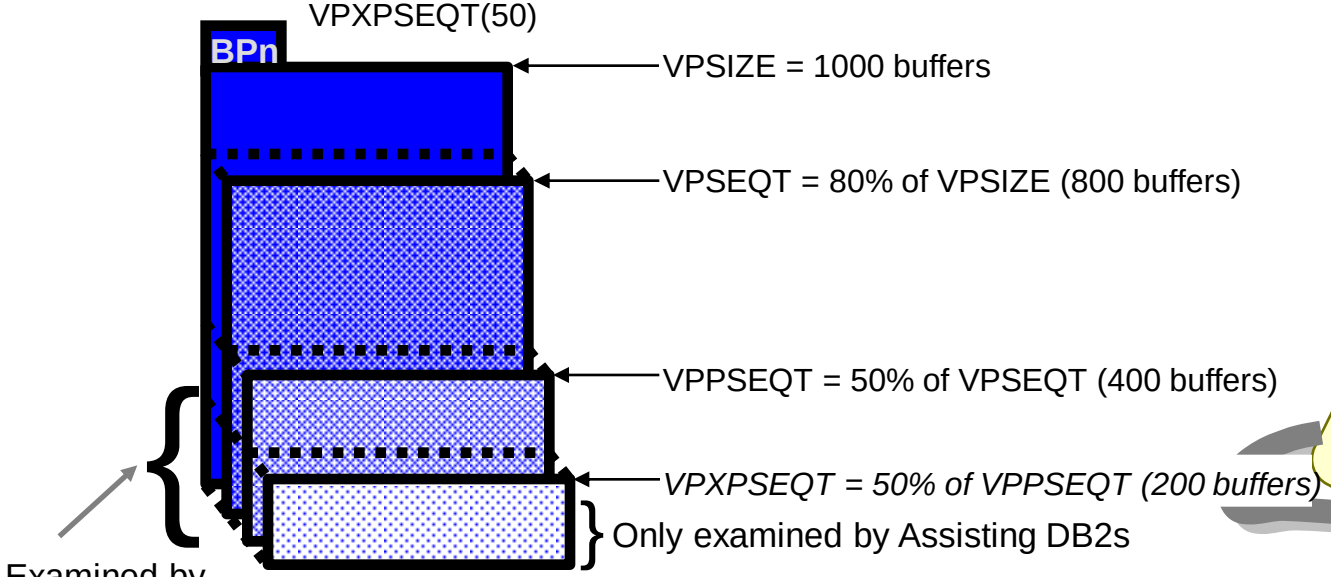
Checked at both bind and run time.

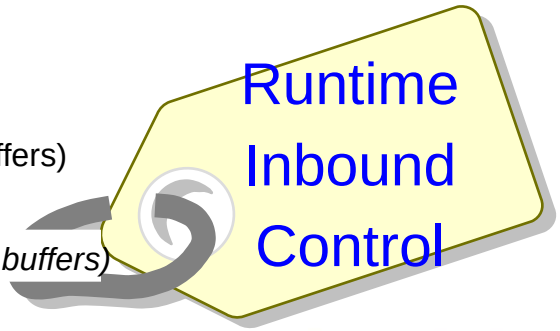
# Sysplex Parallelism BPool




Example:

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



# Sysplex parallelism

## Monitoring and Tuning

- Improving Response Time (affected by the following)
  - CP Contention
  - Buffer Pool Availability
  - I/O Contention
  - ***XCF Links availability***
    - Defined wth 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
- BATCH    PT *    47 PUPPYDML     ADMF001  DSNTEP3  002D     6
-  V443-QUERY COORDINATING DB2=V42A, ORIGINATING TOKEN=30
- DISPLAY ACTIVE REPORT COMPLETE
- 17.10.12 STC00044  DSN9022I <V42D DSNVDT '-DISPLAY THREAD' NORMAL
-  COMPLETION
```

## -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
  - ------------------------------------------------------------------------
...
  - 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**

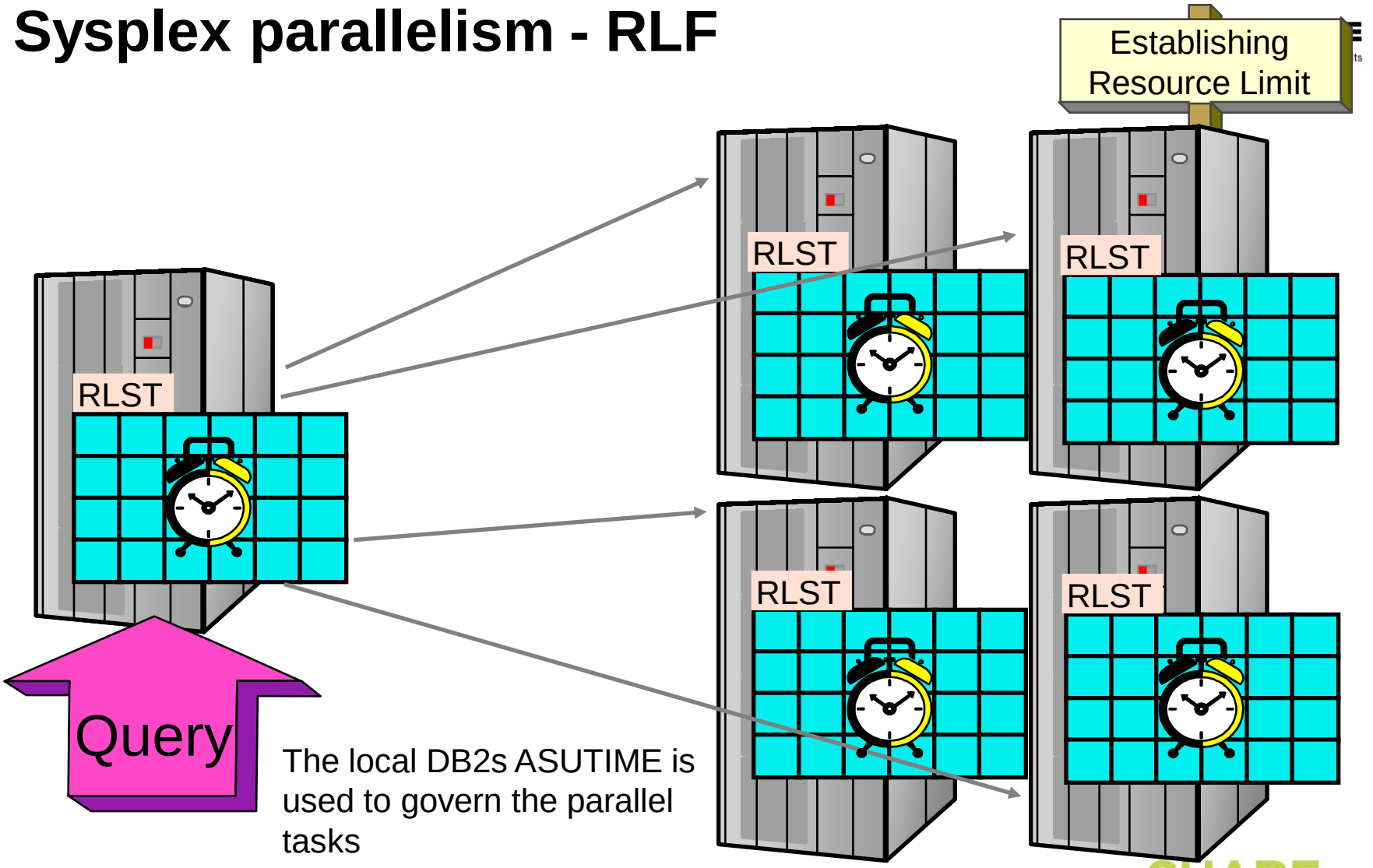
| Member Name | ACE Address | *Query Originating Member* | Originating Task's ACE Address | Number of parallel tasks created | LUWID |
|---|---|---|---|---|---|
| DB2A | 03AA2320 | DB2A | 03AA2320 | 4 | 'NETID.LUName.x.y' |

**Parallel Task Accounting Trace Records**

| Member Name | ACE Address | *Query Originating Member* | Originating Task's ACE Address | Number of parallel tasks created | LUWID |
|---|---|---|---|---|---|
| DB2A | 04B29440 | DB2A | 03AA2320 | 0 | 'NETID.LUName.x.y' |
| DB2A | 05C83460 | DB2A | 03AA2320 | 0 | 'NETID.LUName.x.y' |
| DB2B | 06D29480 | DB2A | 03AA2320 | 0 | 'NETID.LUName.x.y' |
| DB2C | 07E200C0 | DB2A | 03AA2320 | 0 | 'NETID.LUName.x.y' |

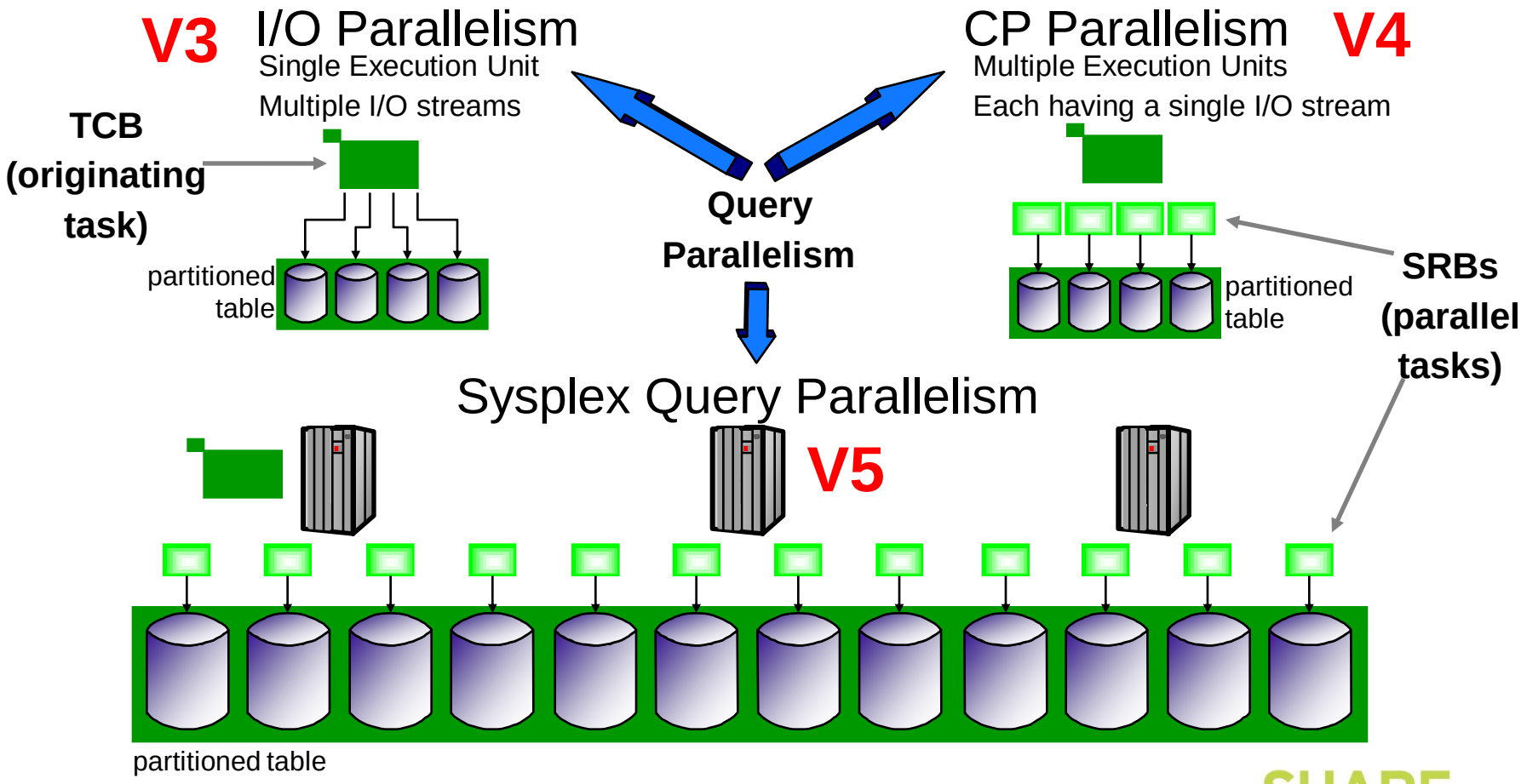
OMEGAMON for DB2 will gather information across the Sysplex in order to give a "one thread" view of the accounting trace data (for batch processing only)

# Sysplex parallelism - RLF




The local DB2s ASUTIME is used to govern the parallel tasks

# Query Parallelism Modes



# 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)

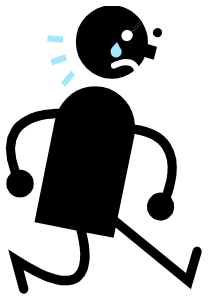
CP:

- 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"

| Bottle neck | Problem | Solution |
|---|---|---|
| **I/O** | 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. | Query I/O Parallelism |
| **CP** | 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. | Query CP Parallelism |
| **CPC** | 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. | Sysplex Query Parallelism |

# 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

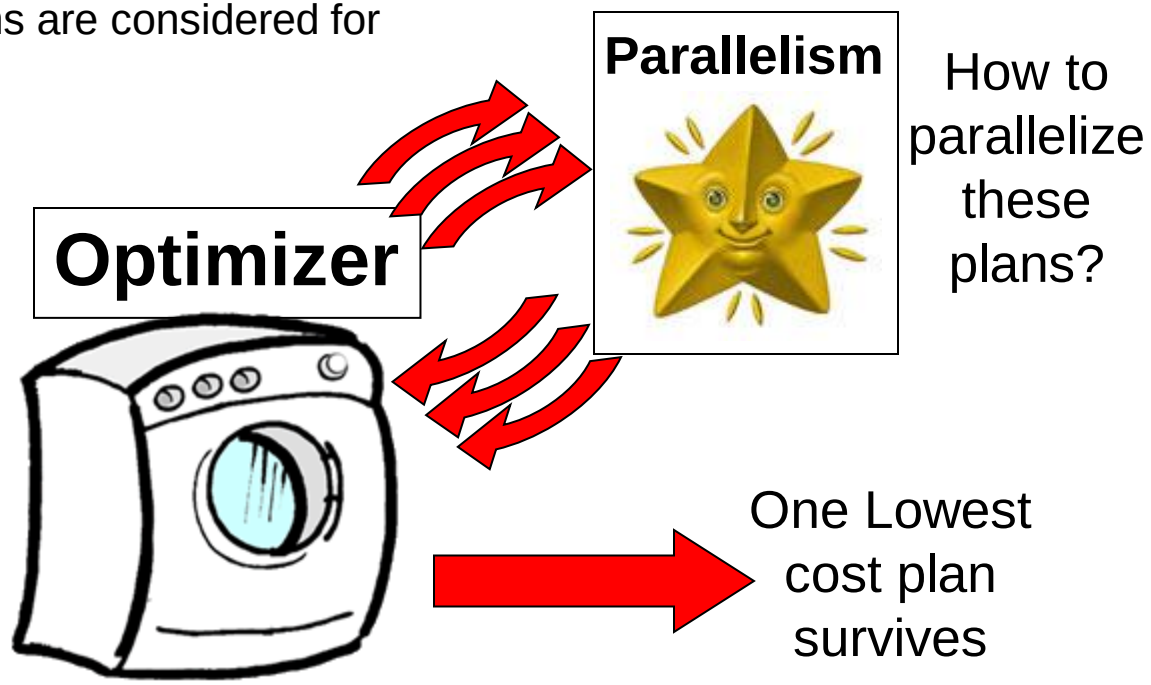
| | DB2 V8 | DB2 9 | Improvement |
|---|---|---|---|
| Total Elapse time (seconds) | ***21320*** | ***12620*** | ***41%*** |
| Total CPU time (seconds) | ***6793*** | ***5351*** | ***21 %*** |
| CPU time eligible for zIIP | ***4756 (70%)*** | ***4676 (87%)*** | |

  - New complex query workload increased to 90% zIIP eligible

| | DB2 V8 | DB2 9 | Improvement |
|---|---|---|---|
| Total Elapse Time (seconds) | ***71660*** | ***8544*** | ***88%*** |
| Total CPU time (seconds) | ***7400*** | ***7514*** | ***-1.5 %*** |
| CPU time eligible for zIIP | ***2924 (39.5%)*** | ***6775 (90%)*** | |

# Parallelism plan determination chaged 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

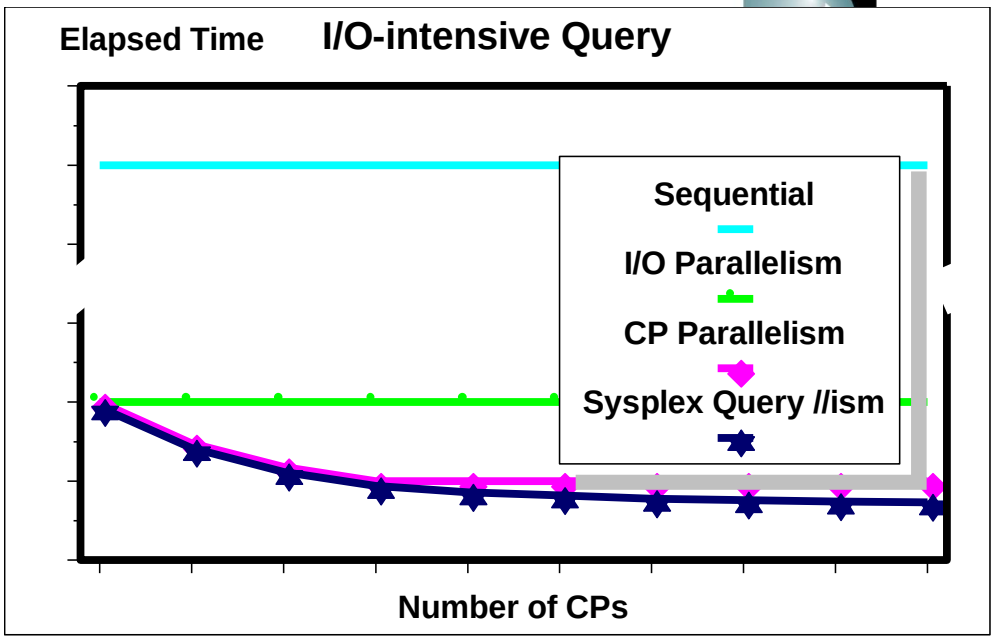


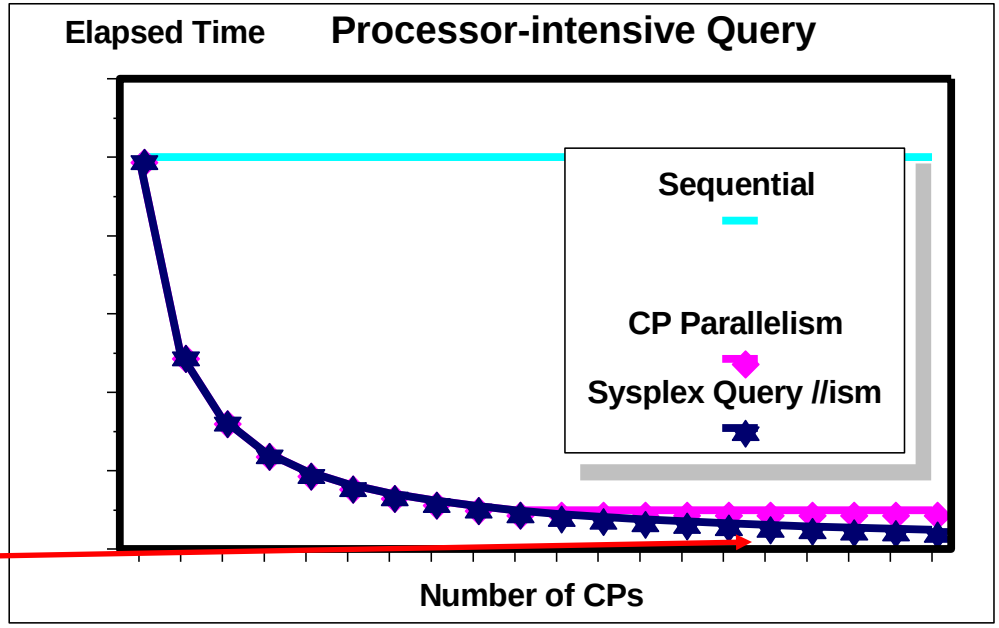
# Performance

- 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!***

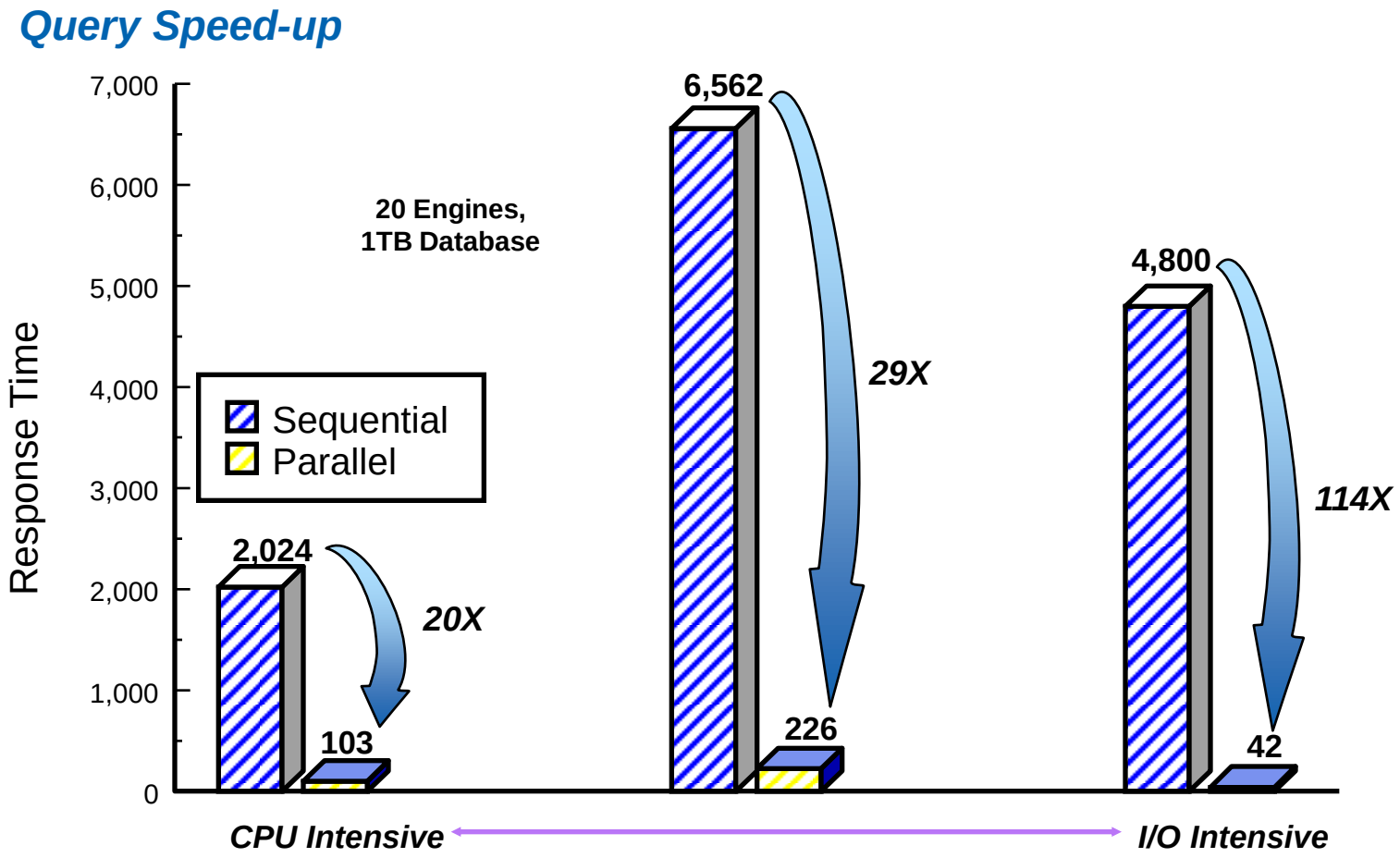
- CP-intensive queries
  - Only multi-tasking significantly reduces the elapsed time of processor-bound queries
  - Once the processing power of a single-CPC is fully utilized, elapsed time cannot be reduced further




**Sysplex Query Parallelism!**

# Parallelism Performance




Speed-up is not limited to the number of CP engines

# Summary

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.

# Disclaimer



***THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.***

***The information on any new products in this presentation is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information on new products is for informational purposes only and may not be incorporated into any contract. The information on new products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. The development, release, and timing of any features or functionality described for our products remains at our sole discretion.***

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

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