Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Jürgen Glag
Düsseldorf, Germany

juergen_glag@compuserve.com

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Car Insurance: Invoice processing

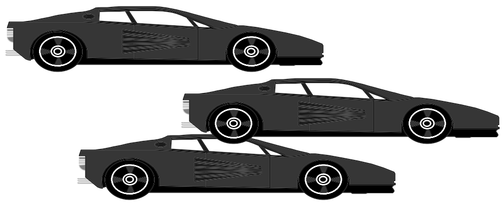
Example #1: once a year for all policies

assume 10 million policies = 10 million units-of-work in one job

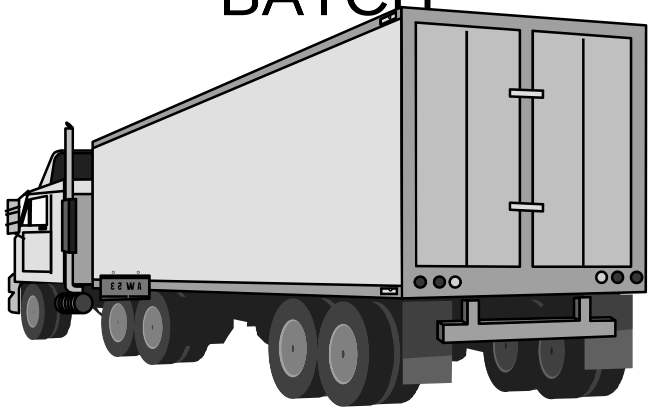
elapsed time = 10.000.000 * 50 ms = 500.000 s
139 hours
ca. 6 days

Probably 10 parallel batch-streams needed
Not viable in parallel to online processing

ONLINE



BATCH



Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

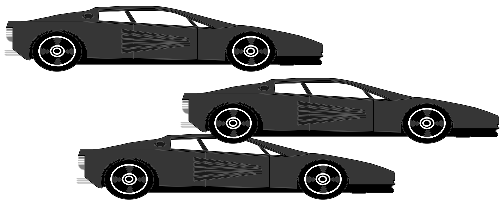
Car Insurance: Invoice processing

Example #2: date of contract = date of invoice

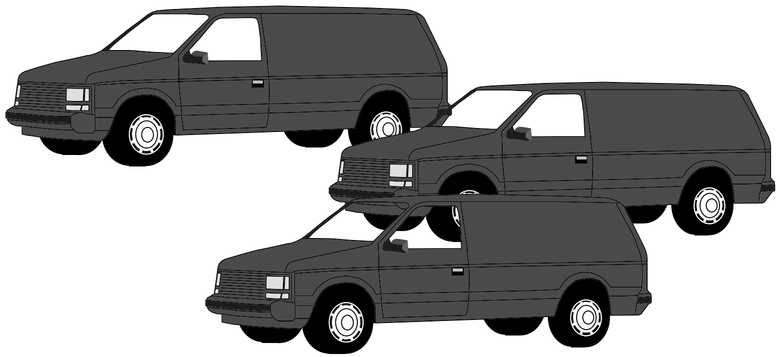
assume 10 million policies
even distribution of date of contact
200 working days/yr = 50.000 units-of-work per day

elapsed time = 50.000 * 50 ms = 2.500 s = 42 minutes
Can be processed in one job
Concurrent processing may have trouble

ONLINE



BATCH



Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

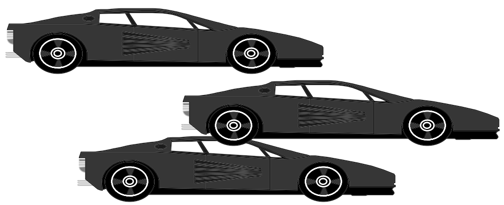
Car Insurance: Invoice processing

Example #3: date of contract = date of invoice
AND batch designed as transactional processing

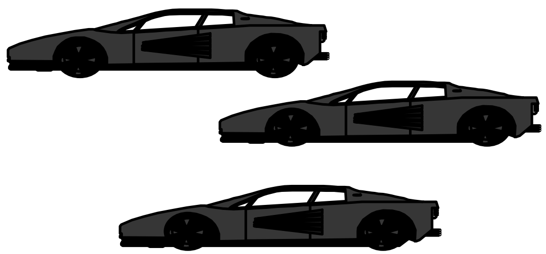
assume 10 million policies,even distribution of date of contact
50.000 units-of-work per day, 1 batch job = 1 unit-of-work

elapsed time = 50.000 * 50 ms = 2.500 s = 42 minutes
Batch jobs started as asynchronous task or by MQ
Solution may be prohibited by organizational restrictions or related job streams

ONLINE



BATCH



Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

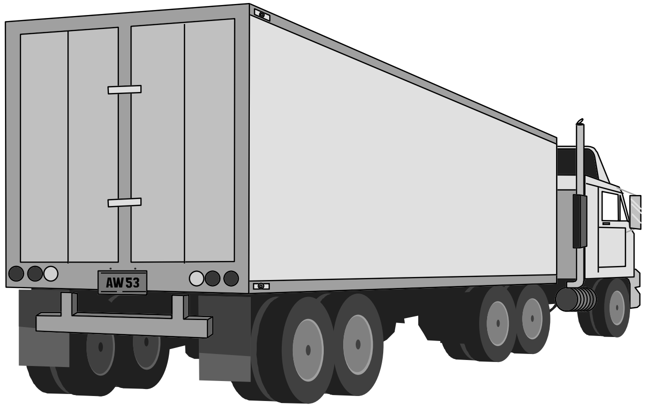
Car Insurance: Invoice processing

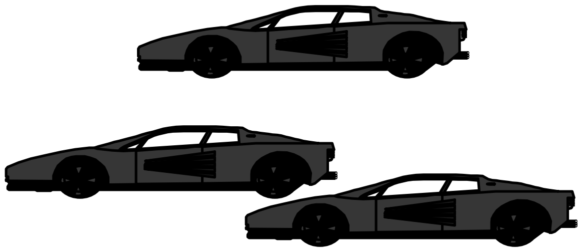
from

Large-Scale Processing

to

non-critical BATCHlets





Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Batch-window is smaller than elapsed time of a "big iron"-job

==> elapsed time of jobs must be reduced

Concurrent Online and Batch processing

No possibility for maintaining a batch-window
demand for short commit intervals
Online and Batch must be compatible in terms of locking

==> many short-running batch jobs
"transactional" batch
process only one logical unit of work
independent processing of independent objects

Why not parallelize additionally?

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

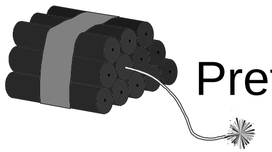
Summary

Design Guidelines for Large-Scale Batch

Prefetch flavors

==> information sources for prefetch
==> sequential prefetch
==> list prefetch
==> dynamic prefetch/sequential detection

Batch profits from dedicated design


Prefetch can kill your online performance


Prefetch is one of the key performance factors for batch

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Prefetch flavors: information sources for prefetch

PLAN_TABLE.PREFETCH:

S =	sequential prefetch
L =	prefetch with page list
blank =	unknown or no prefetch

Cursors will switch on prefetch

```
DECLARE CURSOR
        SELECT ...
        ...
        ORDER BY ...
```

Cursors should always contain the ORDER BY clause

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Prefetch flavors: Sequential Prefetch

=> ORDER BY clause without physical sort, i.e., RID's are in matching sequence

=> First FETCH transmits 32 pages from DASD

=> The application waits for first FETCH as if it were a synchronous I/O, subsequent FETCHes are served from the bufferpool

=> When last matching row from the 32 pages is FETCHed, another 32 pages are transmitted

=> Average time needed for one FETCH: 2 ms

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Prefetch flavors: List Prefetch

=> ORDER BY clause requires physical sort, e.g. ORDER BY is supported by non-clustering index
objective is to avoid the random access to pages

=> Matching RID's are collected (in the RID pool) ...
.. and sorted by page-no and rid-no
.. ANDing/ORing in case of multiple index access is applied
.. resulting RID's are passed to the buffer manager for retrieval

=> Access to the data is skip-sequential

=> Average time needed for one FETCH depends on the number of matching rows per page

less efficient than Sequential Prefetch,
... but much better than random access

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Prefetch flavors: Dynamic Prefetch/ Sequential Detection

=> During run-time all cursors are supervised for sequential access to data, no matter how they are marked in the PLAN_TABLE (at bind time)

=> 5-out-of-last-8-principle:

If 5 (or more) pages of 8 subsequent pages in the page buffer are accessed for rows, Dynamic Prefetch is switched on

If only 4 or less pages from 8 subsequent pages are needed, Dynamic Prefetch is switched off

=> independent from BIND

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Avoid Prefetch in online processing

Sequential Prefetch retrieves 32 pages from DASD
with one call (BP >= 1000 buffers)
Assume row length of 200 bytes => 20 rows/page
At first FETCH 32 * 20 = 640 rows are transmitted
normal online result set: 100 rows, i.e., a 5 pages
result set is processed in more than one dialog step

List Prefetch ... doesn't kill you, it only hurts
sort is performed anyway
small result sets are favorable because of reduced number of I/O's on data
large result sets are more expensive than Sequential Prefetch

Dynamic Prefetch
depends on ORDER BY clause and size of result set

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Avoid Prefetch in online processing

How to get rid of Prefetch?

No selective switch-off possible,
e.g. Sequential Prefetch OFF
List Prefetch ON
All Prefetch flavors are switched off at one time

```
DECLARE cursor-name CURSOR FOR
    SELECT ...
    FROM ...
    WHERE ...
    ORDER BY ...
    OPTIMIZE FOR 1 ROW
```

Prefetch is one of the key performance factors for batch

Examples: Mass processing of data with one cursor
(single table access)

Table-size 10.000.000 rows
rowlength 250 bytes = 16 rows/page = 625.000 pg
result set of cursor amounts to 500.000 rows

Example 1: direct access to data
without Prefetch
with List Prefetch

Example 2: Processing via clustering index

without Prefetch
with Sequential Prefetch

Prefetch is one of the key performance factors for batch

Example 1: direct access to data

without Prefetch

rows are located in 500.000 different pages
500.000 sync.I/O * 20 ms = 10.000 s = 3 hours
(sync. I/O on index leaf pages aren't reflected)

with List Prefetch

same result, if all rows of the result set are located in different pages

shorter I/O-time if pages contain more than one row of result set
e.g. 500.000 rows in 200.000 pages results in I/O time reduced by 60%

Design Guidelines for Large-Scale Batch

Prefetch is one of the key performance factors for batch

Example 2: near-sequential processing via clustering index

without Prefetch

cf. example 1: 3 hours

with Sequential Prefetch

pages are provided asynchronously

dependent on WHERE clause between 31.250 and 625.000 pages have to be read from DASD

31.250 pages * 2 ms = 62,5 s = 1 min

625.000 pages * 2 ms = 1250 s = 21 min

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

What is Parallelism?

Parallel I/O on a table or index in a partitioned tablespace

Flavors of Query-Parallelism

==>	I/O parallelism	since V3.1
==>	CP parallelism	since V4.1
==>	Sysplex parallelism	since V5.1

Prerequisites for Query-Parallelism

==> Bind options

==> Other requirements (e.g., CPU)

I/O parallelism

Technique

utilization of multiple subtasks with asynchronous read engines

every read engine reads strings of 32 pages with Sequential Prefetch into virtual buffer pool

Scenarios

acceleration of I/O bound read-only queries

access to partitioned tablespaces

partitions are positioned on multiple volumes

long elapsed times

Limits

only viable for read-only queries

sufficient size of bufferpool required

parallelism at device level and in the buffer manager, but not in the data manager

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

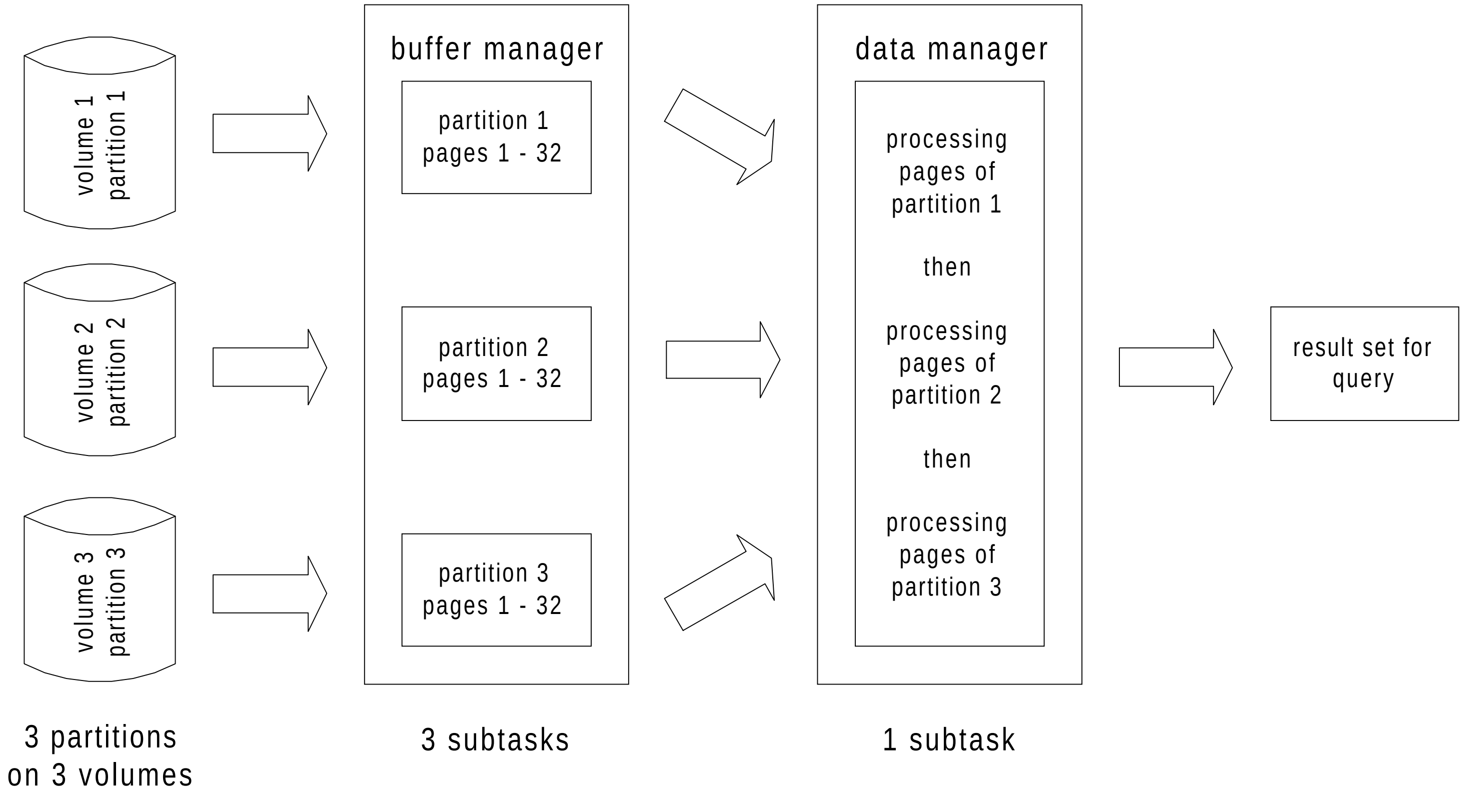
N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

I/O parallelism



Design Guidelines for Large-Scale Batch

INTERNATIONAL DB2 USERS GROUP

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

CPU parallelism

Technique

utilization of multiple subtasks for all DB2 functions,
not only for asynchronous read engines
subtasks can run on all processors of a CEC

Scenarios

acceleration of I/O bound read-only queries
access to partitioned tablespaces
partitions are positioned on multiple volumes
long elapsed times

Limits

only viable for read-only queries
sufficient size of bufferpool required
data must be placed on many devices, else contention of subtasks
the more processors, the better
only with type-2-indexes

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

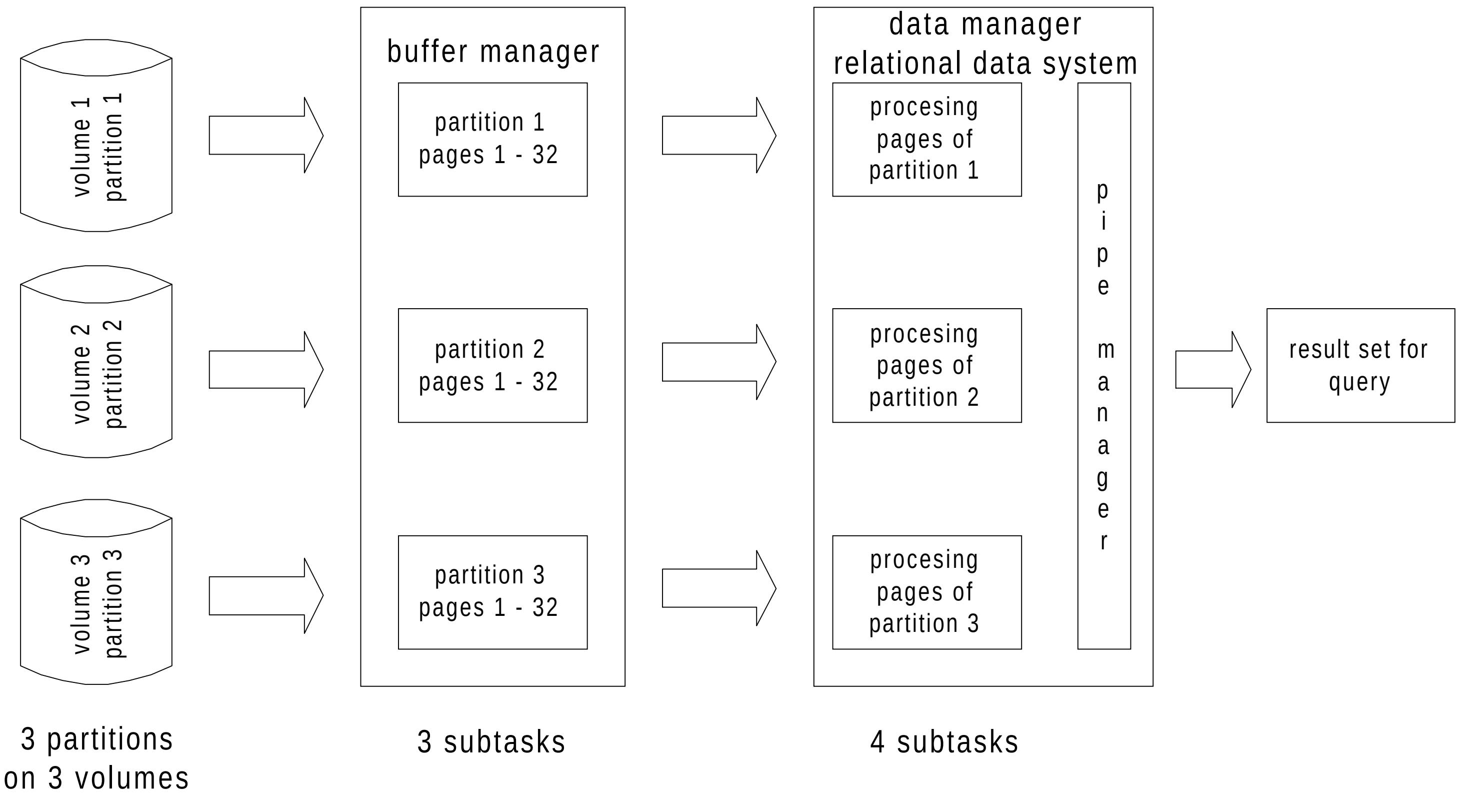
N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

CPU parallelism



Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Sysplex parallelism

Technique

utilization of multiple sub-tasks for all DB2 functions, not only for asynchronous read engines

subtasks can run on all processors of a CEC

1 parallel sysplex = up to 32 MVS/ESA CEC's coupled loosely, each with one MVS/ESA and DB2 image

Scenarios

acceleration of I/O-bound read-only queries

access to partitioned tablespaces, very large TS

partitions are positioned on multiple volumes, at max. 254

very long elapsed times

Limits

should a query be run on various/all processors of a parallel sysplex?

aren't there other tasks waiting for resources?

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

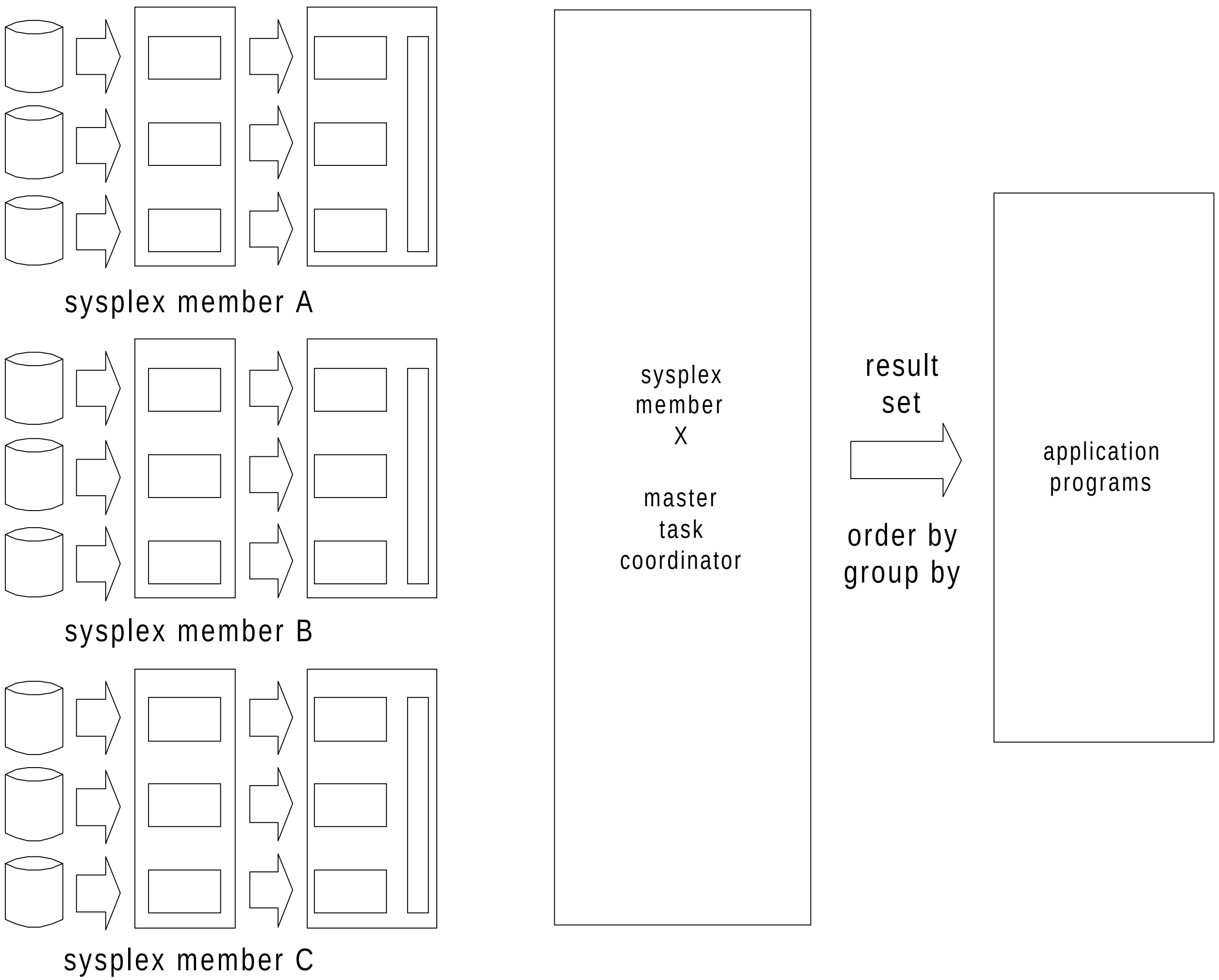
N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Sysplex-parallelism



Prerequisites for Query-Parallelism

- **Bind options**
  - **static SQL**
    - DEGREE(ANY) at BIND or REBIND
    - only effective for static SQL
    - packages contain static **AND** dynamic SQL
  - **dynamic SQL**
    - SET CURRENT DEGREE = "ANY"
    - this special register is only effective for dynamic SQL

Prerequisites for Query-Parallelism

Other requirements

VPPSEQT
(virtual bufferpool parallel sequential threshhold) must be sufficiently sized

For CPU-parallelism at least 2 **ACTIVE** tightly coupled processors are required

Remark: If only one processor is active at the start of a query, I/O-parallelism will be activated

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

PLAN_TABLE is information source

variant 1: EXPLAIN particular statement

step 1: population of PLAN_TABLE

```
EXPLAIN PLAN SET QUERYNO = nn FOR
        SELECT ...
        FROM ...
        WHERE ...
```

host-variables must be eliminated, instead use value or ?

step 2: show results from PLAN_TABLE

```
SELECT *
FROM PLAN_TABLE
WHERE QUERYNO = nn
ORDER BY QBLOCKNO, PLANNO, MIXOPSEQ
```

Design Guidelines for Large-Scale Batch

PLAN_TABLE is information source

variant 2: BIND with option EXPLAIN(YES)

relevant information is stored in package_owner.PLAN_TABLE

CURRENT SQLID is qualifier for PLAN_TABLE in case of dynamic SQL

General remarks:

EXPLAIN(YES) should always be activated

overhead can be neglected in comparison to the benefits

added cost equals additional INSERTs into PLAN_TABLE

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

PLAN_TABLE Column	Explanation
ACCESS_DEGREE	number of parallel tasks of a query settled during BIND with usage of host-variables the value may be 0 number of parallel tasks can differ at execution time
ACCESS_PGROUP_ID	parallel group ID for access to new table (cf. SORTN_) a parallel group is a set of commands with equal number of tasks executed in parallel
JOIN_DEGREE	number of parallel tasks for a join of composite table (SORTC_) with new table is settled at bind time, but can differ at execution time with usage of host-variables the value may be 0
JOIN_PGROUP_ID	ID of parallel group that joins the composite table with the new table
SORTC_PGROUP_ID	ID of parallel group for parallel sort of composite table
SORTN_PGROUP_ID	ID of parallel group for parallel sort of new table
PARALLELISM_MODE	type of parallelism I = query I/O parallelism C = query CPU parallelism X = sysplex query parallelism

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Query uses	Parallelism possible?			Remarks
	I/O	CPU	Syspl	
Isolation RR or RS	Y	Y	N	Transporting the locks to the coordinator too expensive Instead use BIND-option CS or UR and execute LOCK TABLE ... IN SHARE MODE before query execution
access with RID list (list prefetch or multiple index access)	Y	Y	N	PLAN_TABLE.PREFETCH = 'L' PLAN_TABLE.ACCESSTYPE = 'M', 'MX', 'MI', 'MQ'
access with type-1-index	N	N		
correlated subquery	N	N	N	DB2 attempts parallel processing for outer table in case of non-correlated queries both tables are parallelized
IN-list	N	N	N	PLAN_TABLE.ACCESSTYPE = 'N'
updateable or ambiguous cursor with CURRENTDATA(YES)	N	N	N	

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Query uses	Parallelism possible?			Remarks
	I/O	**CPU**	**Syspl**	
OUTER JOIN	N	N	N	PLAN_TABLE.JOINTYPE = 'F', 'L'
Merge scan join with more than one column	N	N	N	
materialized views, materialized nested table expressions	N	N	N	
EXIST in WHERE-predicate	N	N	N	
UNION for more than one query block	N	N	N	DB2 cannot process independent sub-selects in parallel
access to temporary table	N	N	N	If a temporary table is populated with INSERT INTO ... SELECT, the subselect can be performed in parallel

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Read-only Batch running alone	**Read-only Batch running together with others**
FOR FETCH ONLY	FOR FETCH ONLY
PREFETCH	PREFETCH
LOCK table	
Table unload	
Query parallelism on part. TS	Query parallelism on part. TS

Batch with Updates running alone	**Batch with Updates running together with others**
PREFETCH	PREFETCH
Table unload	
Query parallelism on part. TS	Query parallelism on part. TS
Short COMMIT intervals	Short COMMIT intervals

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Further recommendations, esp. for parallel processing

query parallelism	jobs running in parallel	combination of both

Avoid hot-spot partitions, partitions should be of similar size

artificial keys with no (organizational) meaning

otherwise manual balancing necessary
(changes imply DROP + CREATE, wait for version 6)

V6: ALTER INDEX to adjust partition limits,
REORG PENDING

Design Guidelines for Large-Scale Batch

Jobs running in parallel

input data should match the partitioning key

=> no interference on partitioning index
=> no timeouts, no deadlocks

Avoid timeout/deadlock

short COMMIT intervals

parallelize job streams according to partitioning index

Minimize locking

short COMMIT intervals
eliminate ambiguous cursors => **FOR READ ONLY**

Design Guidelines for Large-Scale Batch

Examples for Large-Scale Batch

Reasons to reduce elapsed time

N times Online Processing or dedicated batch design?

Parallelism: Options, Limits, typical usage

Some SQL performs in parallel, some doesn't

Summary

Secondary indexes

Type-1-indexes frequently cause timeout and deadlock problems

=> move to type-2-indexes
=> V6: no further support of type-1-indexes

Update activities of jobs running in parallel

separate class