DB2 LUW

# DB2 LUW performance FAQs

- Db2 night show #260
- C Raghavendra
- Senior staff software engineer
- IBM Software labs Data and AI

## Achievements in IBM

1) Hall Of Fame Award - 2019
2) Michelin Client Award for Automation - 2018
3) Outstanding Technical Achievement Award(OTAA) - 2018
4) IBM Rock star Award 2018
5) Successful Completion of ONC Datacenter Move project 2017
6) GCH Approved Ideas for db2 LUW automation Ideas - 2017
7) Eminence and Excellence Cash Prize Award - 2016
8) Eminence and Excellence Cash Prize Award - 2017
9) Top Innovator and Mentor Award of 2016
10) Top Innovator and Mentor Award for 2015
11) Michelin Client Certification of Appreciation - 2014
12) Best Automation for Db2 LUW - 2013
13) Appreciation from Database Service Engineering Team - 2020
14) Db2 LUW Python tool kit Performance script contribution
15) Top Innovator and Mentor award for 2020
16) IBM Gold Champion Learner 2020
17) Awarded IDUG Bronze member award
18) Awarded IDUG silver Member Award

IBM

DB2 LUW

1. Why Runstats is very useful for the Db2 optimizer?

2 Why we should Never feed table scans to Db2 Optimizer?

3. Are we spending lot of time in compiling SQL's?

4. Why IO Intensive Queries are bad for Buffer pool?

5. Why do queries take long time to execute?

6. How do you find SQLS having most reads for the table?

7. How do you drill down which connection is holding the log?

8. Why time spent reading pages into the buffer pool matters?

9. How can we avoid Db2 Lock escalation from row to table level?

10. What is the Influence of DBM configuration parameters on Db2 Optimizer?

11. How is the Bufferpool Accessed by the Application Request?

12. Why it is critical to estimate Buffer pool performance?

13. How does life and time of a prefetch request works?

14. How do you investigate the current Group Buffer pool size is not enough?

DB2 LUW

15. How can you limit the percentage of dirty pages?
16. How does prefetching data using I/O servers will happen?
17. What is this Dirty steal scenario and how we can limit it?
18. How to troubleshoot a Routine behaving slow?
19. How to Prevent Data corruption In Buffer pool by invalid memory access?
20. What is the Correlation Between Backups and CPU Usage?

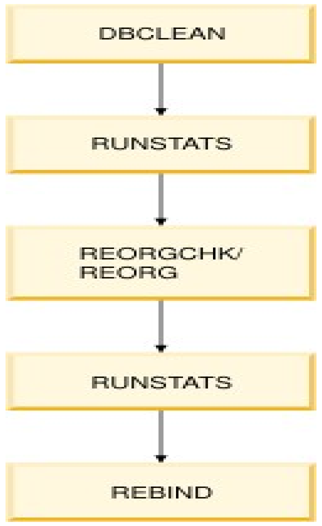
IBM

DB2 LUW

## 1. Why Runstats is very useful for the Db2 optimizer?

Runstats is vital utility for tables and indexes in the database. It is always good practice to have latest statistics so that the Db2 Optimizer gets the latest updates which helps in better performance of the tables and indexes.

So, whenever changes are happening in the table either by Insert/delete/update and other transactions then always use Runstats utility to update the statistics.



Scenario 1:

For example: Reorg utility is run for the table T1 and say Runstats is not run, and the statistics are not updated then even with a rebind, the Db2 Optimizer may continue to use the old access plan rather than a new one. Static SQL requires a rebind to possibly get a new access path, but dynamic SQL can take advantage of the new stats immediately. So, the Reorg may not give full benefit unless Runstats is run.

There are two primary ways in which Reorgs can help performance. Even without doing runstats and rebinding, if the reorg improves clustering by the clustering index, performance can improve; if there were many overflow rows, the reorg will clean them up and this will improve performance.

Result: No opportunity to possibly get a new access path and more improvement in the performance of the queries!

DB2 LUW

Scenario 2:

Reorg utility is run for the table T1 and also Runstats is run after that with Rebind then the statistics will be updated in the catalog tables also and the Db2 optimizer will possibly get an updated access plan.

This will only be the case if a new (and better) access path is chosen. If the old access path was good, the reorg and runstats may not change the access path and therefore the rebind will have no effect on performance.

Result: Queries performance have the best chance to improve, and the execution time will improve fast!

How to check the latest stats time for the table?

db2 "select stats_time from syscat.tables where tabname='TABNAME'"

DB2 LUW

```
-bash-4.4$ db2 "select stats_time ,tabname from syscat.tables where tabschema='XMETASR'"

STATS_TIME                 TABNAME
-------------------------- --------------------------------------------------------------------------------------------------------------------------------
2018-12-04-09.59.08.368385 XMETAMVIEWS
2018-12-04-09.55.13.623286 XMETAREGISTRY
2018-12-04-09.55.13.634172 XMETAMTABLES
2018-12-04-09.59.08.326921 XMETAMFIELDS
2019-01-10-11.52.56.106842 XMETAARTIFACT
2019-01-10-11.52.56.116267 XMETALOCKINFO
2018-12-06-11.08.14.374720 XMETAPROPERTIES
```

DB2 LUW

## 2 Why we should Never feed table scans to Db2 Optimizer?

Tables scans are the prime factor for the poor performance in the larger tables with huge volume of the data. Always a good practice is go with good methods where table scans can be avoided and the DB2 OPTIMIZER will not feel the overhead!.

What factors cause a table scan?
--------------------------------------------

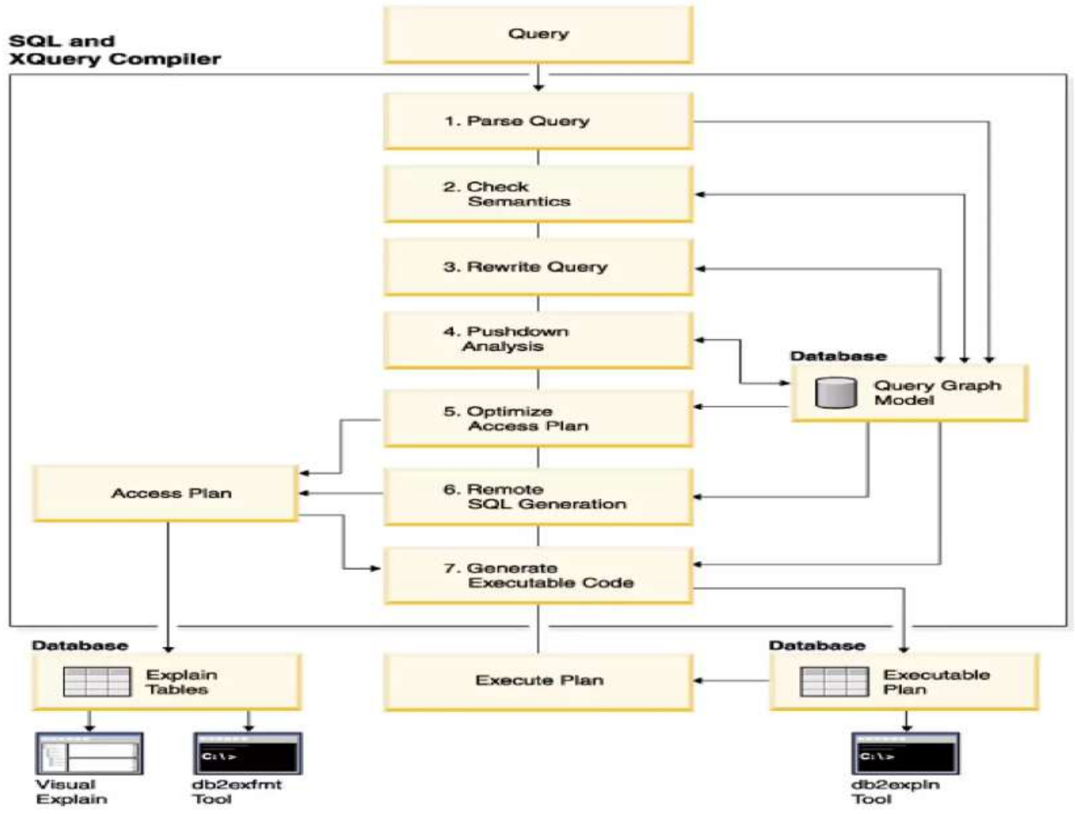
1. The tables are hugely fragmented, and no data maintenance approach is done periodically with Reorg and Runstats and Rebind.

2. The catalog statistics say the table is small, and there are no statistcs available for the table.

3. The table could be small and the DB2 decides to go for the tablespace scan instead of using the indexes.

4. Say if a query is using columns in a where clause and proper indexes are not created for those fields!

# Phases of the Optimizer



5)The predicates of the query do not match with the many available indexes on the table

Tables cans act as a huge showstopper for the larger tables with loads of volume of the data.

A lot depends on the query structure how it is written and neatly designed!

The less the table scans for the system the more the efficiency and less overhead on the I/O.

Also, proper indexes gets used when the need is there.

## 3.Are we spending lot of time in compiling SQL's?

The system is behaving slow and also you found out wait time is not an issue, then it is good to move the direction to the processing time in the system!.

One of the key factors to check is are we spending too much time in compiling the SQLs?

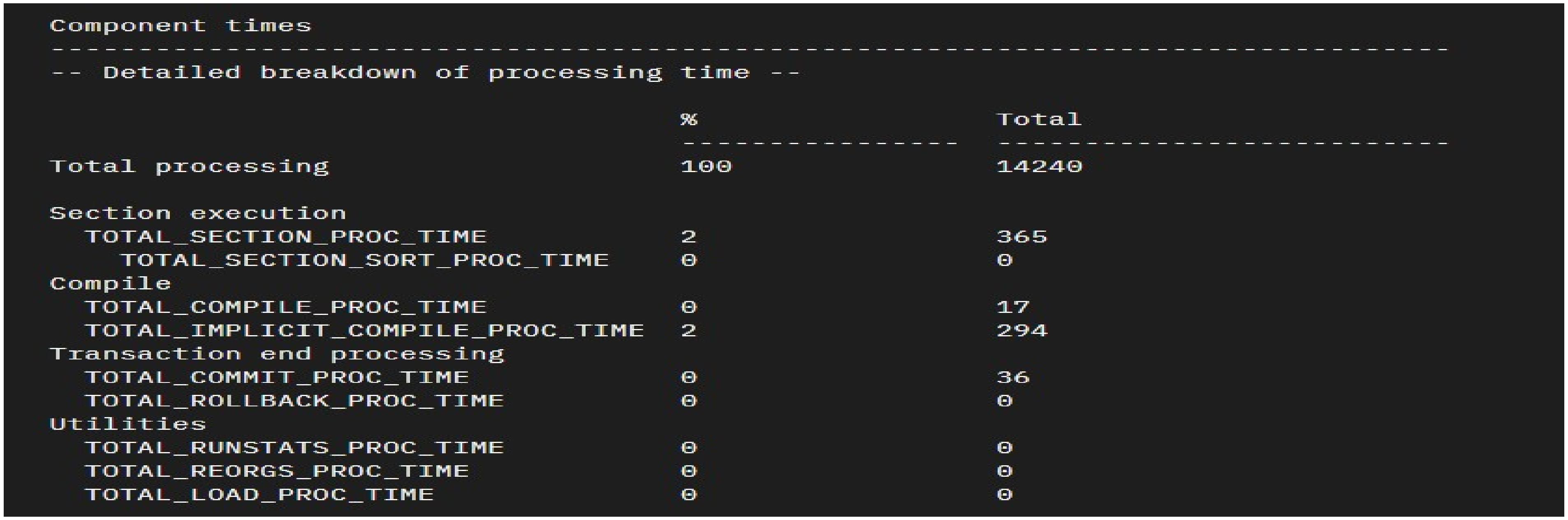
This isn't necessarily a bad thing – sometimes it's unavoidable, for example in warehouse environments with complex SQL and high optimization levels – but it's good to avoid if we can.

IBM

## How do you explore this metrics?

DB2 LUW

monreport. dbsummary () execution gives these parameters and the related parameters for these is TOTAL_SECTION_PROC_TIME TOTAL_COMPILE_PROC_TIME The second one gives us the time spent in compiling SQLs in the percentage details For example, if the value of TOTAL_COMPILE_PROC_TIME = 24% say then 24% of the time is spent in compiling the SQL's TOTA_SECTION_PROC_TIME - 3% means only 3% of the time is spent in executing the SQL's So it is wastage of resources and hinders the database performance!.



```
Component times
--------------------------------------------------------------------------------
-- Detailed breakdown of processing time --

                                    %                 Total
                                    ----------------  --------------------------
Total processing                    100               14240

Section execution
  TOTAL_SECTION_PROC_TIME           2                 365
    TOTAL_SECTION_SORT_PROC_TIME    0                 0
Compile
  TOTAL_COMPILE_PROC_TIME           0                 17
  TOTAL_IMPLICIT_COMPILE_PROC_TIME  2                 294
Transaction end processing
  TOTAL_COMMIT_PROC_TIME            0                 36
  TOTAL_ROLLBACK_PROC_TIME          0                 0
Utilities
  TOTAL_RUNSTATS_PROC_TIME          0                 0
  TOTAL_REORGS_PROC_TIME            0                 0
  TOTAL_LOAD_PROC_TIME              0                 0
```

DB2 LUW

## 4.Why IO Intensive Queries are bad for Buffer pool?

1. Queries performance matters a lot for the database performance processing.

2.If the queries are doing better in terms of all metrics, then the database also does well.

3.If we concentrate here for the queries which are IO intensive, then it is bad sign for Buffer pool performance.

4.Because queries which are IO intensive, they process the results from the disk than the Buffer pool.

5. So, when the processing is happening at the disk level than buffer pool, then the hit ratio of the buffer pool will be low that takes a hit on the database performance too.

6. So, if we want to avoid this circumstance then look at the options of turning the IO intensive queries to Buffer pool Oriented one.

IBM

DB2 LUW

- How do you find these troublesome IO intensive queries?

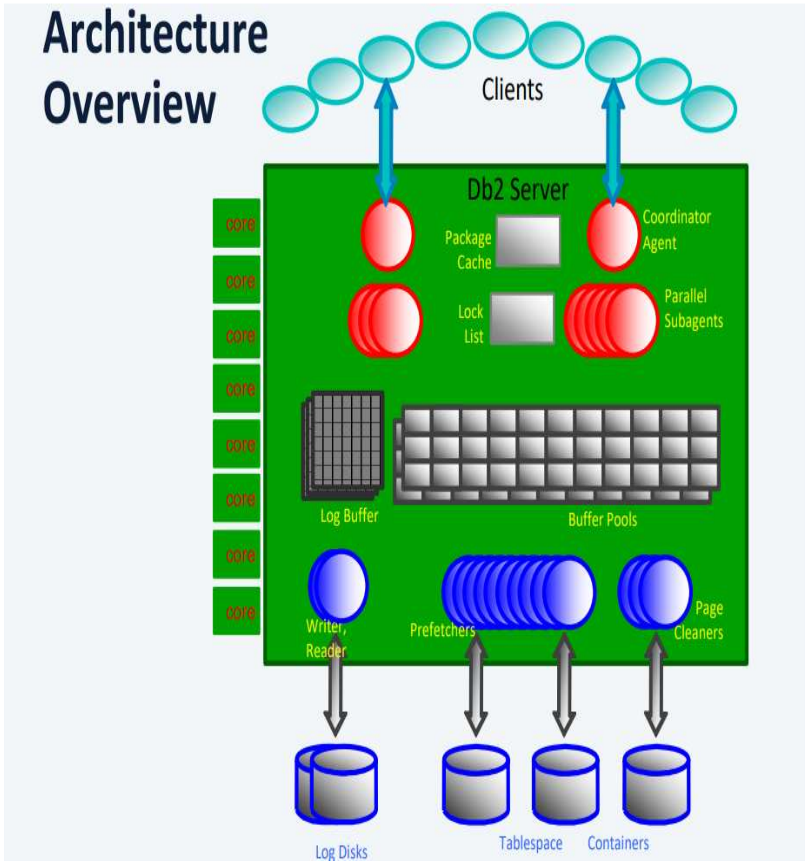
- 5) select stmt_exec_time, num_executions, (pool_read_time + pool_write_time + direct_read_time + direct_write_time) as io_time from table(mon_get_pkg_cache_stmt(null, null,null,-2)) as t order by io_time desc fetch first 5 rows only;

- 6) If pool_read_time+pool_write_time+direct_read_time+direct_write_tim if the above sum is more then it is IO intensive queries

- 7.The common reasons for this to happen is pages are unavailable for the queries to process in Buffer pool.

- 8.) So, page cleaning mechanism has to work correct. Also indexes proper usage wherever it is needed is good practice.

- 9. Proper configuration of buffer pools and setting its related configuration parameters will avoid this issue to maximum extent



## 5. Why do queries take long time to execute?

SQL queries will take long time to execute for multiple reasons.

When the table is newly created and the data is populated with less volume, then the queries will be almost working perfectly fine.

But the real trouble creeps in , when the volume of the data is increasing in the table and also the database maintenance utilities are not running on a periodic basis.

Hence the queries start taking long time to execute.

**MON_CURRENT_SQL - Retrieve key metrics for all activities on all members**
The MON_CURRENT_SQL administrative view returns key metrics for all activities that were submitted on all members of the database and have not yet been completed, including a point-in-time view of currently executing SQL statements (both static and dynamic) in the currently connected database.
You can use the MON_CURRENT_SQL administrative view to identify long running activities and prevent performance problems. This view returns metrics that are aggregated across all members.
The schema is SYSIBMADM.

DB2 LUW

**How to reduce the query execution time?**

1.Check whether the proper indexes are created on the table and on the fields where the query clause is using(using syscat.indexes or describe indexes for table command)

2.Check whether the regular Reorg and Runstats are run on the tables.(stats time in syscat.tables and Reorgchk command)

3.Check whether the cost of the queries is more or less(Db2expln or Db2exfmt)

4.Check the statement execution time from the package cache.(monreport.pkgcache())
If the query is executing long then there are lot of contributing factors for it. we have to dig down for it.

DB2 LUW

## 6.How do you find SQLS having most reads for the table?

Rows Read and Rows written are the critical metrics for the performance of the queries. There will be lot of queries which does huge reads based on the transactions.

How can we find those culprit SQL's making most reads?

```
SELECTMEMBER,EXECUTABLE_ID,NUM_REFERENCES,NUM_REF_WITH_METRICS,ROWS_READ,ROWS_INSERTED,ROWS_UPDATED,
ROWS_DELETEDFROM TABLE(MON_GET_TABLE_USAGE_LIST(' ', ' ', -1)) ORDER BY ROWS_READ DESCFETCH FIRST 10 ROWS ONLY;
```

DB2 LUW

**How to get the text of that SQL?**

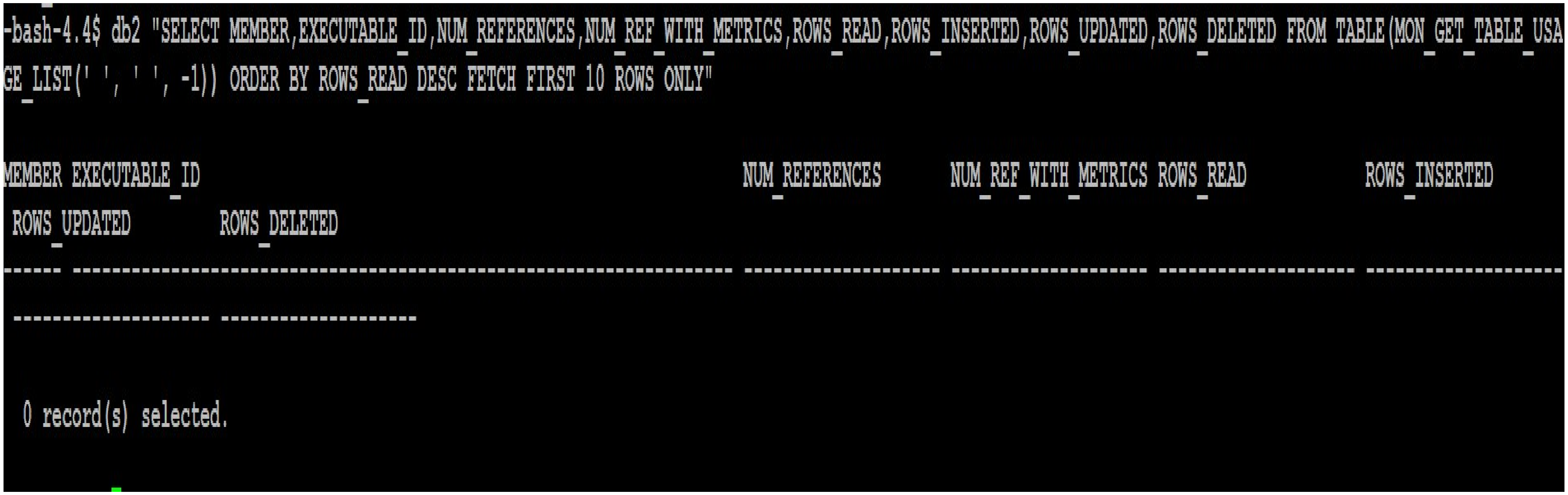
Get the Executable id from the above SQL and then pass into the MON_GET_PKG_CACHE_STMT table function.

SQL Query One Liner to get this task done :

```
SELECT STMT_TEXTFROM
TABLE(MON_GET_PKG_CACHE_STMT(NULL,x'01000000000000007C0000000000000000000000200200
81126171720728997', NULL, -1))
```

For a batch job processing a large % of the table, it will need to have read a high number of rows. Therefore, high rows read is not always a problem.

DB2 LUW



```
-bash-4.4$ db2 "SELECT MEMBER,EXECUTABLE_ID,NUM_REFERENCES,NUM_REF_WITH_METRICS,ROWS_READ,ROWS_INSERTED,ROWS_UPDATED,ROWS_DELETED FROM TABLE(MON_GET_TABLE_USA
GE_LIST(' ', ' ', -1)) ORDER BY ROWS_READ DESC FETCH FIRST 10 ROWS ONLY"

MEMBER EXECUTABLE_ID                                                                  NUM_REFERENCES       NUM_REF_WITH_METRICS ROWS_READ            ROWS_INSERTED
 ROWS_UPDATED         ROWS_DELETED
------ ------------------------------------------------------------------- -------------------- -------------------- -------------------- --------------------
 -------------------- --------------------

  0 record(s) selected.
```

## 7.How do you drill down which connection is holding the log?

Log space filling up is one of the critical information for the database performance issue. The more the log space is held by the transactions it degrades the performance.

Normally there will be excessive long running transactions in the database system which causes the performance issues. The consequences of these long running transactions are :

1.Contention due to locks held too long
2.Active log space filling up

So, if the problem is active log space filling up then the only area where you find the information related to the transaction which is holding the log space was Db2diag.log before Db2 V10.5

DB2 LUW

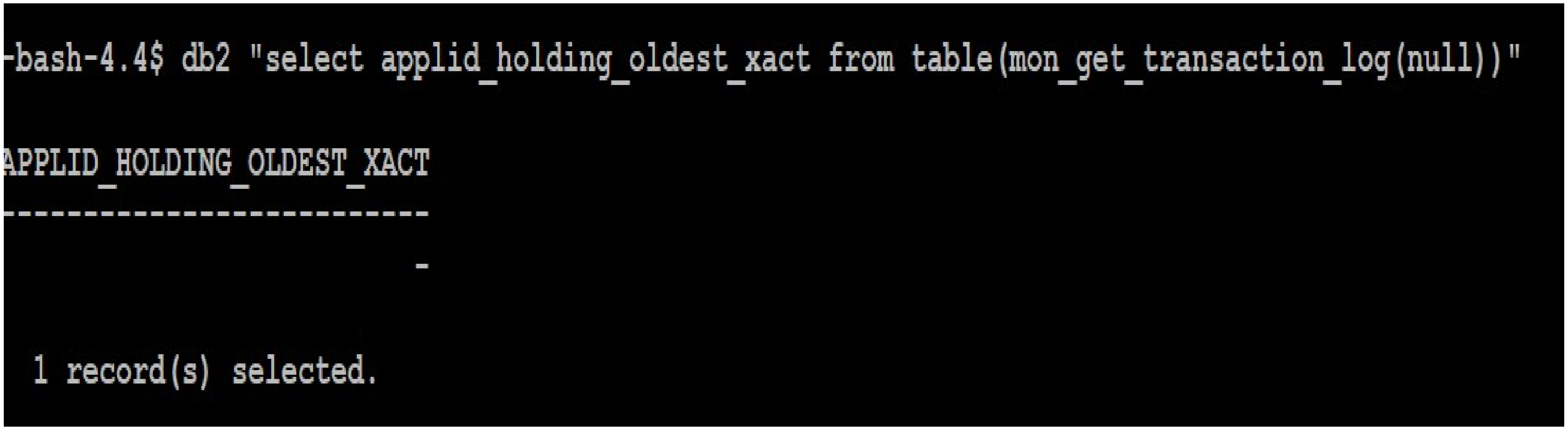
## What’s the new way of finding after Db2V10.5?

DB2 provides a very critical monitoring element in the table function mon_get_transaction_log() called as applid_holding_oldest_xact .

So, frame a query to select that

```
select applid_holding_oldest_xact from table(mon_get_transaction_log(null))
```

Say you get the value of 1568 then this is the culprit id which is holding the active log space.
Once it is identified then the option can be given whether to force off the application id or not. Based on the choice the task can be executed .



```
-bash-4.4$ db2 "select applid_holding_oldest_xact from table(mon_get_transaction_log(null))"

APPLID_HOLDING_OLDEST_XACT
--------------------------
                         -


  1 record(s) selected.
```

IBM

## 8. Why time spent reading pages into the buffer pool matters?

DB2 LUW

One of the most important factors for the performance of the database is time spent reading pages into the buffer pool. This is called as pool read time. This pool read time can be very high at times due to the too many reads requests or individual reads taking too long time to complete. To get into the cause of this at the table space level, check the average read time from the MON_GET_TABLESPACE table function.

1)More than 4ms on sdd is getting bad and more than 2ms on ssd is bad - then the cause can be due to the storage problem or too many I/O requests.

2)Here the tip is calculate average read time for busiest tablespaces , in case some storage paths have longer read times than others.

Also, one more area to look upon here is , is there a high rate of physical reads or low buffer pool hit ratio?.

For this
1)Look at the individual statements which have the highest I/O and also look for methods to reduce that by indexing etc.
2)Once IO is minimized with indexing method, If pool read time is still higher then look for increasing the Bufferpool size or the storage.

- **How do you monitor Pool Read time metrics?**

------------------------------------------------------------

- We can get the pool_read_time metrics by the execution of the monreport.dbsummary() stored procedure!.

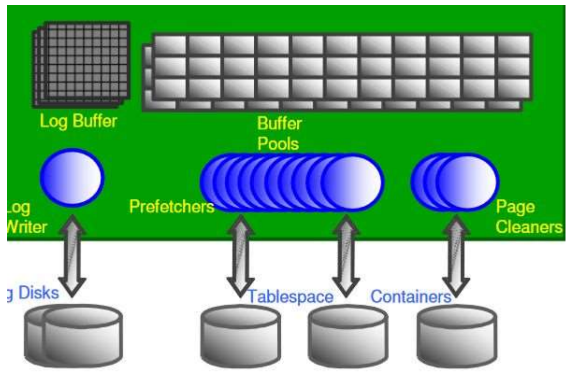
- Once you execute this procedure say db2 call "monreport.dbsummary(30)">dbsummary.out

- Then in the output file look into the section of

- Detailed Breakdown of TOTAL_WAIT_TIME

- Here it will have two parts I/O Wait time and Pool_read_time

DB2 LUW



## 9. How can we avoid Db2 Lock escalation from row to table level?

When a RR isolation level is used then we will get lot of row level locks and when it crosses a certain threshold depending on the MAXLOCKS and LOCKLIST parameter it escalates to table level lock.

This process is called as Lock escalation. Even if the transaction does not reach the threshold for lock escalation, it will also result in more locks being held for a longer period of time.

With RR, every row that is looked at by Db2 will get a read lock. In most cases you should be using CS isolation level. As the program goes through the cursor, Db2 releases the lock on the previous row.

DB2 LUW

## Can we avoid this Scenario?

Yes, we can avoid this scenario, Db2 provides a registry variable

DB2_AVOID_LOCK_ESCALATION

When the DB2_AVOID_LOCK_ESCALATION registry variable is ON, lock escalation will not be performed.

Instead, SQL0912N is returned to the application that requested the lock that would normally result in lock escalation. The application is able to either COMMIT or ROLLBACK which will free the locks held by this application. This variable can be updated online without restarting the instance.

```
db2set DB2_AVOID_LOCK_ESCALATION = ON (Default is OFF)
```

Changes to this variable do not require the database instance to be restarted. This variable is applicable to all the operating systems!

DB2 LUW

## 10. What is the Influence of DBM configuration parameters on Db2 Optimizer?

Database manager configuration parameters are those which influence the behavior of the instance and the databases that are running under it.

There are some critical database manager configuration parameters that influence the Db2 Optimizer!

1)Parallelism(INTRA_PARALLEL)
2)CPU Speed(CPUSPEED)
3)Communications speed(COMM_BANDWIDTH)

Let we discuss each in a brief how they influence the DB2 Optimizer.

## 1. Parallelism (INTRA_PARALLEL):

Indicates whether intra-partition parallelism is enabled. When YES, some parts of query execution (e.g., index sort) can run in parallel within a single database partition.

## 2.CPU Speed (CPUSPEED):

The optimizer uses the CPU speed (milliseconds per instruction) to estimate the cost of performing certain operations.
**Recommendation:** do not adjust this parameter unless you are modelling a production environment on a test system or
assessing the impact of a hardware change

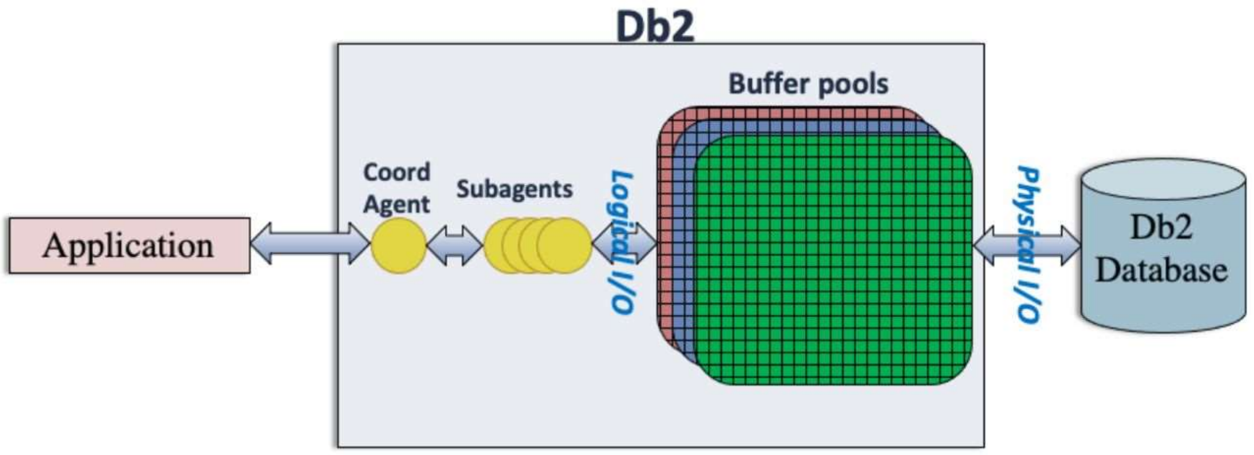
## 3.Communications speed (COMM_BANDWIDTH):

The optimizer uses the value (megabytes per second) in this parameter to estimate the cost overperforming certain operations between the database partition servers in a partitioned database environment.

## 11. How is the Bufferpool Accessed by the Application Request?

- For the better performance of the applications, it is always a good sign to use the bufferpools instead of disk for the processing so that it will be quicker as the pages are used in the bufferpool than the disk.

Now when the applications is accessing the bufferpools then what exactly happens internally at the Db2 luw architecture level?



Please find the flow of steps for it

1.The application opens a connection to process a statement that is initially handled by the Coordinator Agent.

2.The coordinator Agent will subdivide the tasks across the multiple sub agents and each subagent will do their tasks.There will only be subagents if intra_parallel is on and max_query_degree is > 1 and dft_degree is > 1

3.Depending on the application request, we may or may not execute logical I/O to check the buffer pools for pages.

4.If the pages are found in the Buffer pool for processing of the request by the application, then Logical I/O is executed which will happen in the buffer pool itself.

5.If the pages are not found in the Buffer pool for processing then the application will process Physical I/O which will fetch the pages from the disk level to process.

6.It then resides in the buffer pool until it is later victimized to make room for another page.

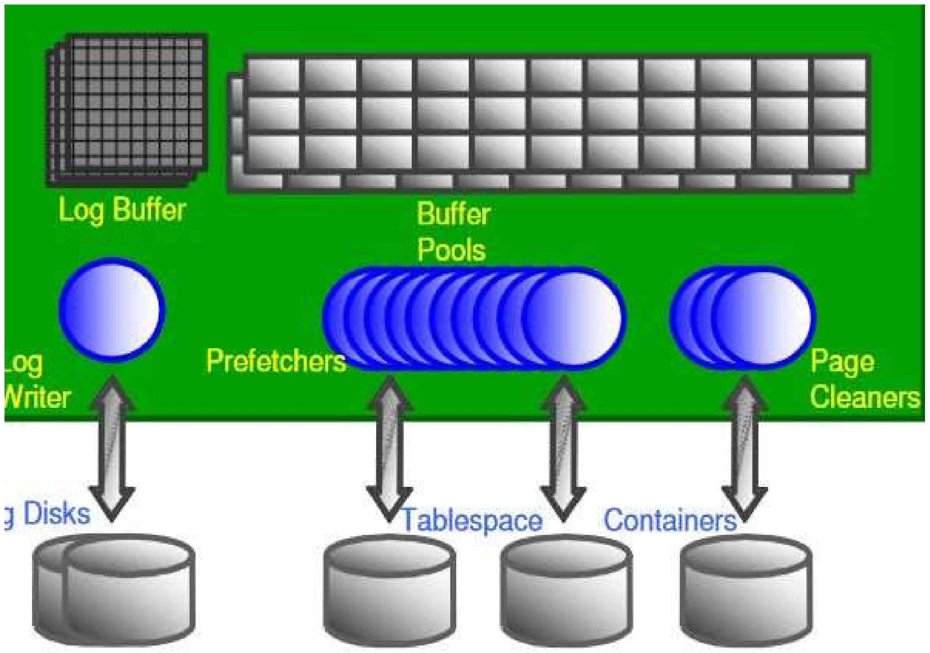
DB2 LUW

## 12.Why it is critical to estimate Buffer pool performance?

Bufferpool is part of database shared memory where the pages gets placed from the disk to the Bufferpool before the application processes these pages!

Buffer pool hit ratios are one of the most fundamental metrics, and give an important overall measure of how effectively the system is exploiting memory to avoid disk I/O.

Hit ratios of 80-85% or better for data and 90-95% or better for indexes are generally considered good for an OLTP environment, and of course these ratios can be calculated for individual buffer pools using data from the buffer pool snapshot.



**Data pages**

Data pages**:** ((pool_data_lbp_pages_found - pool_async_data_lbp_pages_found) / (pool_data_l_reads + pool_temp_data_l_reads)) × 100

**Index pages**

Index pages**:** ((pool_index_lbp_pages_found - pool_async_index_lbp_pages_found ) / (pool_index_l_reads + pool_temp_index_l_reads)) × 100

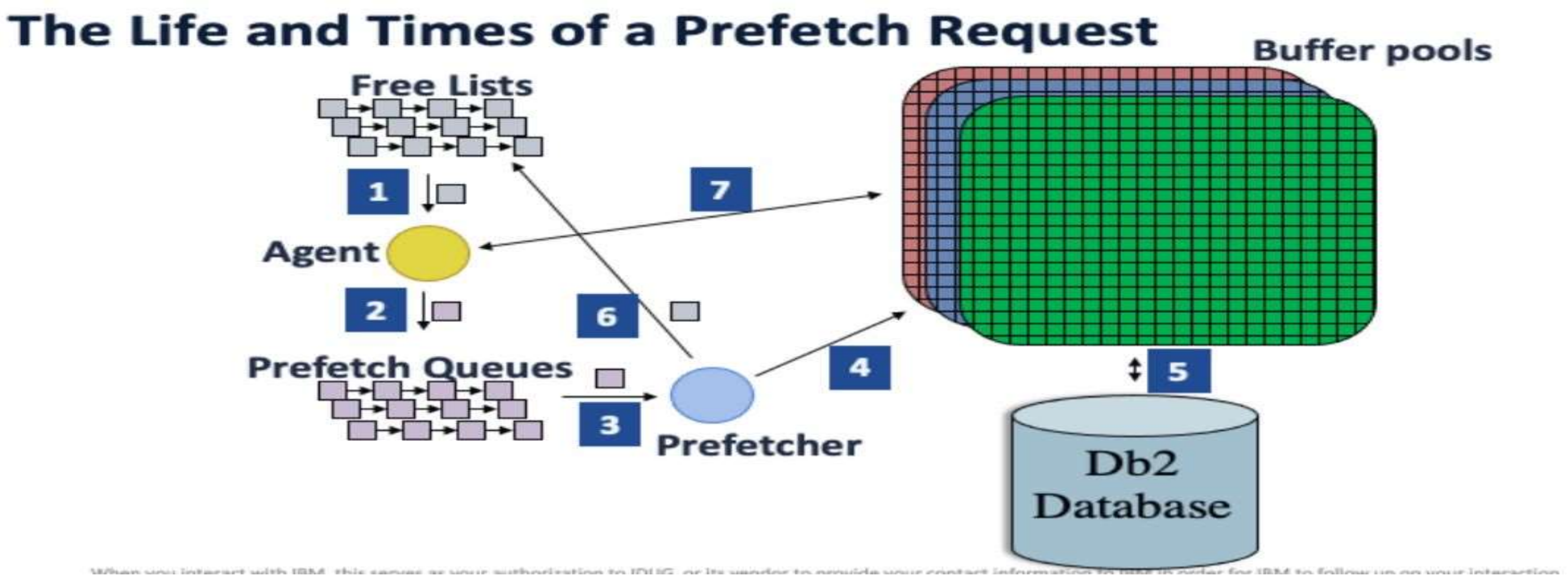
**XML Storage object**

XML storage object (XDA) pages**:** ((pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found ) / (pool_xda_l_reads + pool_temp_xda_l_reads)) × 100

If the Bufferpool is healthy and getting utilized well then there is less need of I/O, means hitting the disk to get the data is reduced which is very good sign for the performance of the applications.

On the other face of the coin , if the Bufferpool is not healthy and the hit ratios are poor then applications will process from the disks which deters the performance as it uses disks instead of memory which is not a good sign for the database performance!

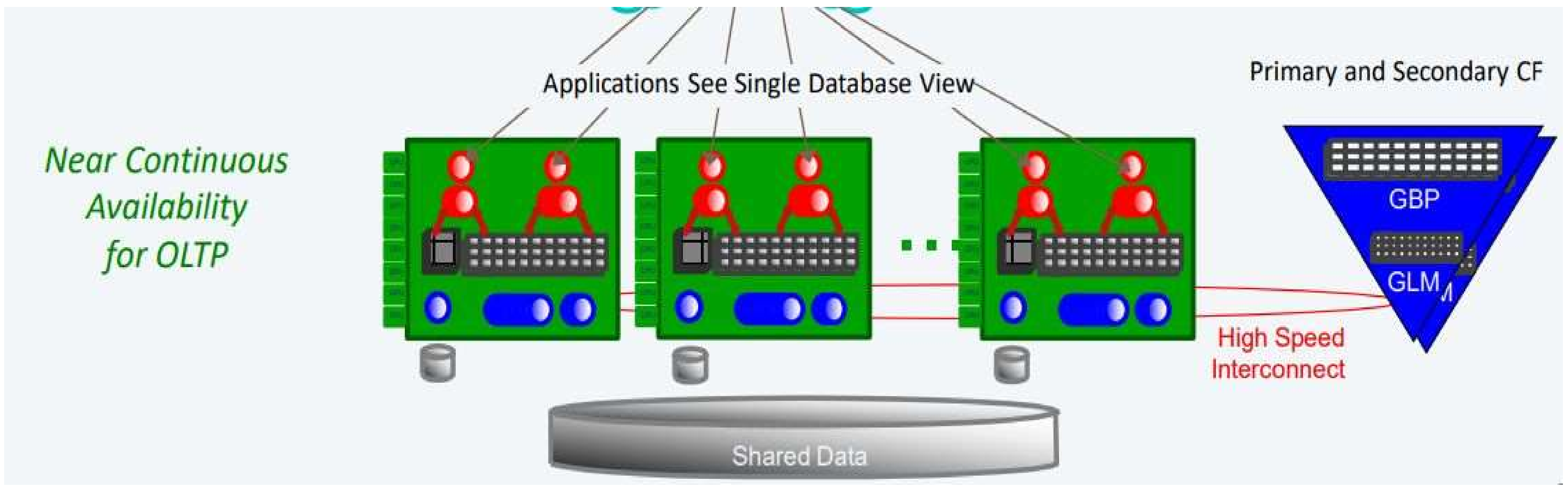
**13. How does life and time of a prefetch request works?**



**Please find the flow how a db2 prefetch request works :**

1. Agent locates an available prefetch request from a free list .
2. Agent populates the prefetch request with info about what pages to prefetch and places it on a prefetch queue.
3. Prefetcher retrieves the populated prefetch request from a prefetch queue.
4. Prefetcher drives logical I/O for requested pages.
5. Physical I/O takes place for any pages that are not already in the buffer pool
6. Prefetcher resets the prefetch request and places it back on a free list
7. Agent executes logical I/O for the prefetched pages

## 14. How do you investigate the current Group Buffer pool size is not enough?


Applications See Single Database View
Primary and Secondary CF
Near Continuous Availability for OLTP
GBP
GLM
High Speed Interconnect
Shared Data

DB2 LUW

Db2 LUW pure scale works on the concept of the Group Buffer pools as it operates at the different db2 member levels.

At times the size of the Group Buffer pool size used reaches full and it is not sufficient for the operation of the transactions.

How do you investigate this?

If the group buffer pool (GBP) does not have sufficient space when attempting to register a page or write a page to the GBP, a GBP_FULL error occurs.

```
SELECT SUM(T.NUM_GBP_FULL) AS NUM_GBP_FULL FROM TABLE(MON_GET_GROUP_BUFFERPOOL(-2)) AS T;

NUM_GBP_FULL
------------
24

1 record(s) selected.
```

So here 24 times the Group Buffer pool is full and has given error.

If the value of NUM_GBP_FULL increases by more than one per minute, then the current size of the GBP likely does not meet your needs.

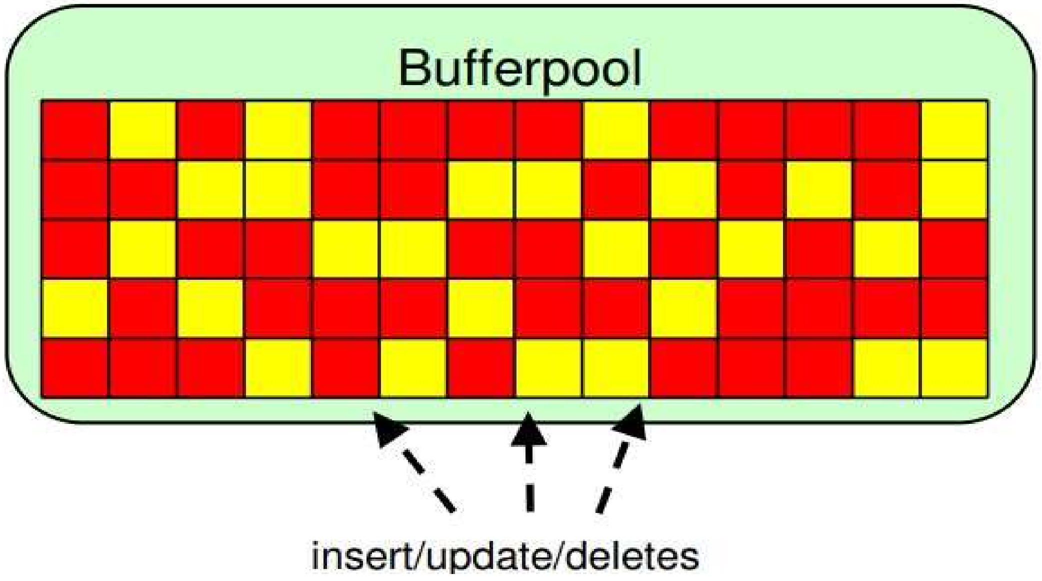
In this case, increase the size of the GBP with the command:

```
UPDATE DB CFG USING CF_GBP_SIZE
```

IBM

**15. How can you limit the percentage of dirty pages?**

## Limit the percentage of dirty pages



Now 61% dirty pages!

There is a db cfg parameter called chnpgs_thresh which does this job.

chngpgs_thresh is the point where cleaners start cleaning if they have not started already. If using alternate cleaners this parameter has no effect.

Once the cleaners are started softmax(old way) or page_age_trgt_mcr(new way) are the way you control the page cleaners to make sure yu have enough clean pages to not dirty steal.

If there are no clean pages an agent will steal a page and clean it syncronously, it will not wait for a cleaner to clean a page
-db cfg parameter: chngpgs_thresh ("changed pages threshold")

-Page cleaners are triggered if percentage of dirty pages exceeds this value

- Default value: 60

- Lower value: more aggressive page cleaning (min: 5)

- Higher value: more delayed page cleaning (max: 99)

## Dirty page steals a show stopper(Db2 LUW internals)

DB2 LUW

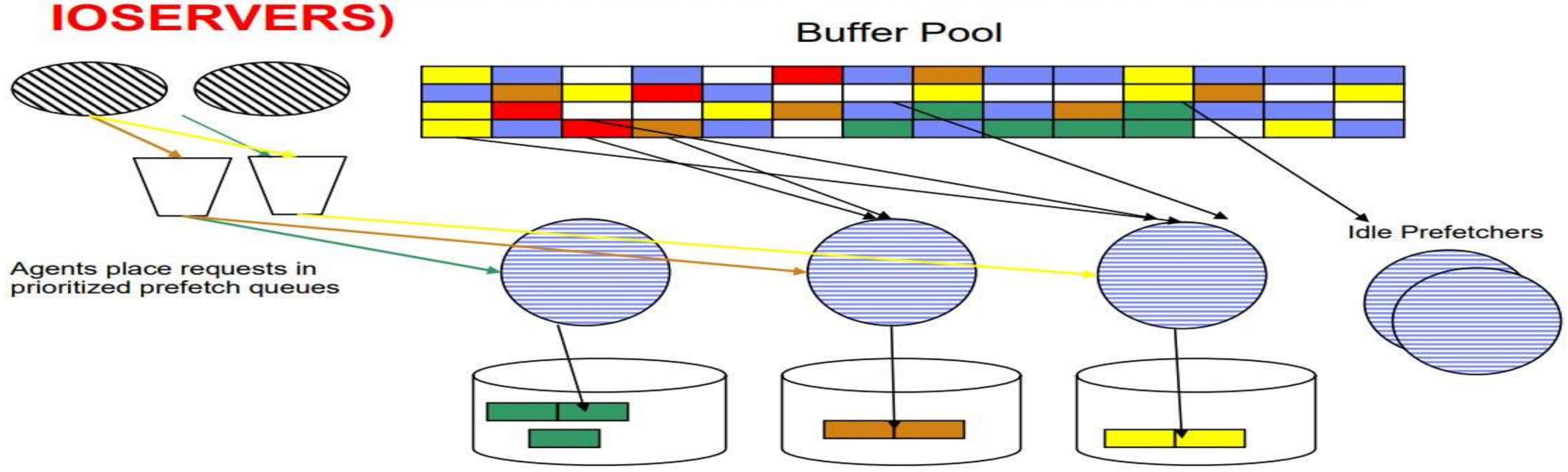
1) Dirty page steals can be a real thorn in your side.
2) If the page cleaners aren't keeping up (or if their disk writes are too slow, etc
 then
3) DB2 is faced with an insufficient supply of clean pages in the bufferpool.
4) As a result, agents that want to bring in pages from disk have nowhere clean to put them
5) so they have to synchronously write the dirty page to 'make way'.
6) We want to keep these low – so if you're getting more than 1 or 2 per thousand transactions
7) then you should look into whether cleaning can be made more aggressive (lowering softmax, lowering chngpgs_thresh)
8) pool_drty_pg_steal_clns - Buffer pool victim page cleaners triggered monitor element
9) The number of times a page cleaner was invoked because a synchronous write was needed during the victim buffer replacement for the databas
10) MON_GET_BUFFERPOOL table function has the above monitor element pool_drty_pg_steal_clns
11) When the DB2_USE_ALTERNATE_PAGE_CLEANING registry variable is OFF:
 -The pool_drty_pg_steal_clns monitor element is inserted into the monitor stream.
 -The pool_drty_pg_steal_clns monitor element counts the number of times a page cleaner was invoked because a synchronous write was needed
12) When the DB2_USE_ALTERNATE_PAGE_CLEANING registry variable is ON:
 -The pool_drty_pg_steal_clns monitor element inserts 0 into the monitor stream.
 -There is no explicit triggering of the page cleaners when a synchronous write is needed during victim buffer replacement. To determine whethe
 please refer to the pool_no_victim_buffer monitor element.

Note: Although dirty pages are written out to disk, the pages are not removed from
 the buffer pool right away, unless the space is needed to read in new pages

**16. How does prefetching data using I/O servers will happen?**

## Prefetching

## I/O Prefetching done by DB2 by Prefetchers (aka. IOSERVERS)



Pages are read directly into buffer pool memory on most platforms. Due to limitations on some platforms I/O is done in big block reads to a private buffer within the prefetcher then copied into buffer pool memory

Prefetching pages means that one or more pages are retrieved from disk in the expectation that they will be required by an application.

Prefetching index and data pages into the buffer pool can help to improve performance by reducing I/O wait times.

In addition, parallel I/O enhances prefetching efficiency. I/O servers are used to prefetch data into a buffer pool.

1.The user application passes the request to the database agent that has been assigned to the user.

2.The database agent determines that prefetching should be used to obtain the data that is required to satisfy the request and writes a prefetch request to the I/O server queue application by the database manager.

3.The first available I/O server reads the prefetch request from the queue and then reads the data from the table space into the buffer pool.

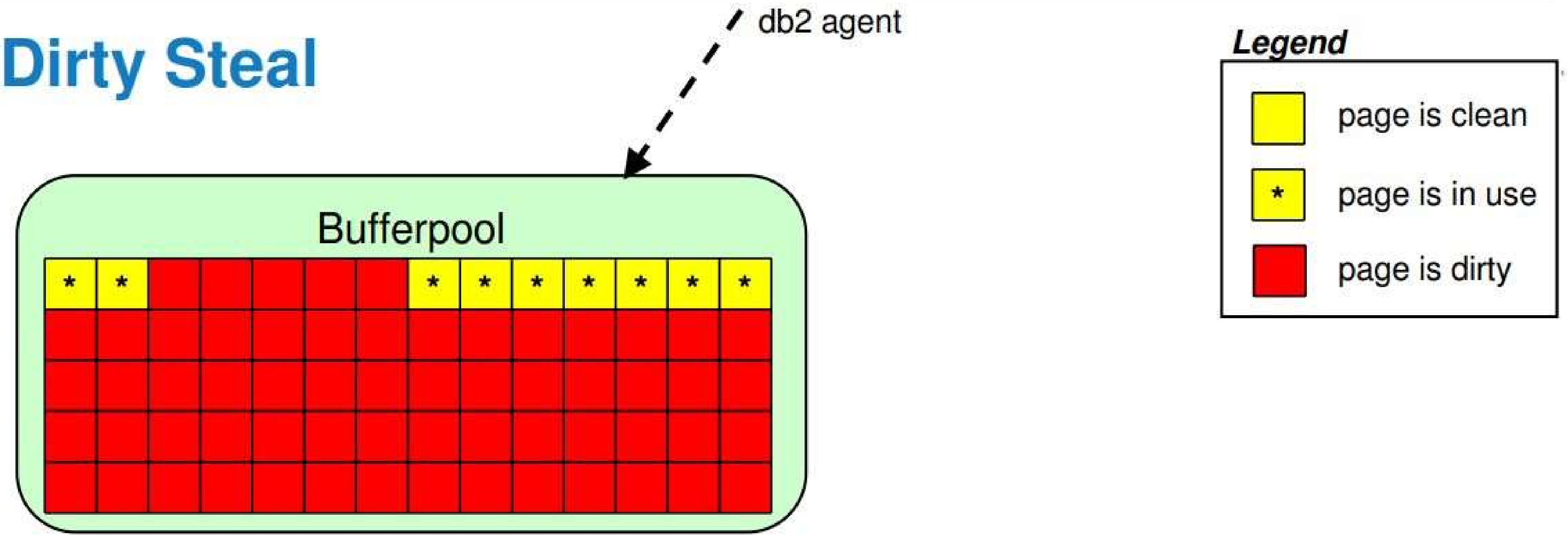
4.The number of I/O servers that can simultaneously fetch data
from a table space depends on the number of prefetch requests in the queue and the number of I/O servers specified by the num_ioservers database configuration parameter.

5. The database agent performs the necessary operations on the data pages in the buffer pool and returns the result to the user application.

17.What is this Dirty steal scenario and how we can limit it?

DB2 LUW

## Dirty Steal



The individual units of the database where the data resides are in pages! So the page cleaning and tuning become very vital for the database performance to be good and consistent.

**The pages can be broadly classified into three states:**
•Page is clean
•Page is in use
•Page is dirty (Dirty pages contain data that has been changed, but not written to the disk)
Page cleaning is an asynchronous process that is a big part of the huge power of Db2 buffer pools. When Db2 changes a page, the log file record is externalized to disk. This provides the "Durability" in ACID. However, the dirty page is still in the buffer pool.

If the percentage of the dirty pages is more in the bufferpool then that is hindrance to the bufferpool performance, in this scenario page cleaners gets triggered, and the pages will be cleaned so that the pages can be used by the transactions.
When this dirty steal scenario gets formed?
•DB2 agent executes a transaction, needs to access a page
•Page is not in the BP (i.e. buffer pool miss), need to read the page
•Agent cannot find a vacant slot found in the BP to read-in a page!

**Consequence:** Transaction needs to wait for dirty pages to cleaned out synchronously , which takes a significant hit on the performance!

## 18.How to troubleshoot a Routine behaving slow?

At times in a database, a routine which was performing well starts behaving very slow in execution. So how to deal this performance issue?

1)Check the stored procedures details using MON_GET_ROUTINE table function

2)Analyze what are the critical queries that are involved inside the routine that are executing

3)Say for example you find a critical select statement inside the code of a routine which is performing slow, then go with the steps of troubleshooting how to deal with the query improvement

But the golden question is if we have huge dmls operating inside the routine then it will be time consuming for the access plans to generate for each individual statements so how to deal with this?

1) For example, if you are using db2expln to get the cost of the statement then

2)It provided an option to get the access plan at the routine level

3) db2expln -m option can be used

4) So, get the package list for all the routines in the database and store in a file with the package name and routine name

5) Then write a small script and in the loop condition use the db2expln -m option to get the access plan for the routine level

6)We can store all the access plans generated for each routine in a particular path

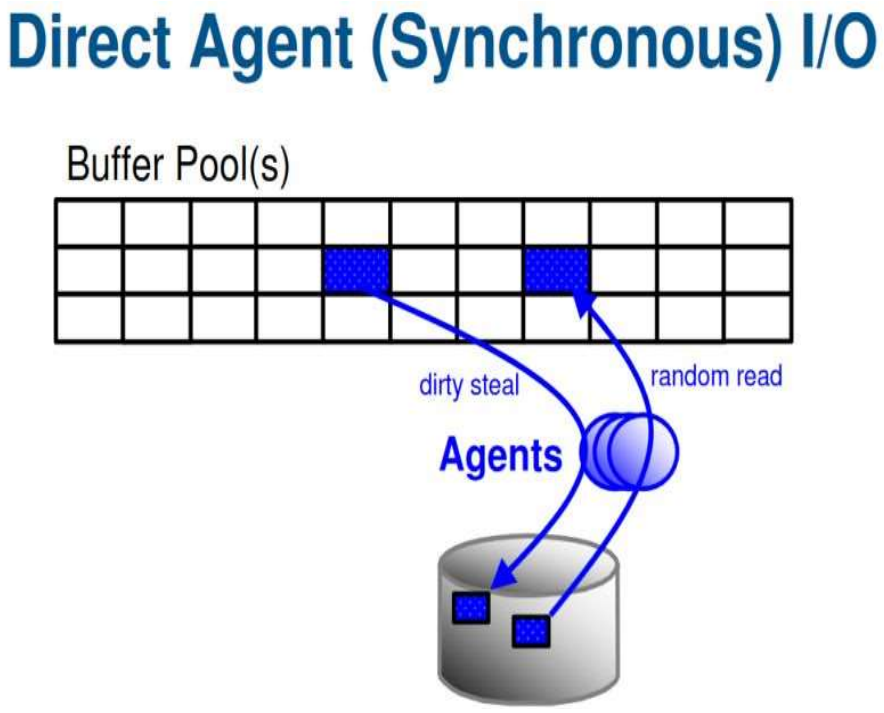
The advantage of the above procedure is a lot of time gets saved when we get a access plan at the routine level than individual SQL level!

## 19.How to Prevent Data corruption In Buffer pool by invalid memory access?

Buffer pool is one of the important area in the DB2 LUW architecture which has to be protected for the performance improvement.

As per the thumb rule it is always good if the hit ratio of the Buffer pool is above 90% so that it is getting utilized to the fullest extent and also there is very less overhead at the disk level, which in turn contributes to the overall database performance!.

Sometimes data corruption can happen at the buffer pool level, because of the invalid memory access!



**How to prevent this scenario?**

——————————————————-

Db2 provides a registry variable for this which can be enabled !.

The registry variable is DB2_MEMORY_PROTECT This registry variable enables a memory protection feature that uses storage keys to prevent data corruption in the buffer pool caused by invalid memory access.

Memory protection works by identifying at which times the Db2 engine threads should have access to the buffer pool memory and at which times they should not have access.

When DB2_MEMORY_PROTECT is set to YES, any time a Db2 engine thread tries to illegally access buffer pool memory, that engine thread traps.

Note: You will not be able to use the memory protection if DB2_LGPAGE_BP is set to YES. Even if DB2_MEMORY_PROTECT is set to YES, Db2 will fail to protect the buffer pool memory and disable the feature.

DB2 LUW

## 20. What is the Correlation Between Backups and CPU Usage?

1.Backups are very important for the data to be up to date so that any time we need it back can be restored!

2.But full backup is very costly process, especially when the database size is huge like in db2 warehouse environments.

3.CPU usage will be at the high level when the full backup is happening, because lot of metrics are getting backed up like logs, tablespaces etc.

4.So it is always a good practice to take full backup weekly once and go with the incremental backups in the remaining days.

5.Each successful backup and restore operation generates a single record in the db2diag.log file, which provides information on the performance of that operation

IBM

DB2 LUW

6.The log record is informational and is written at diaglevel 3 (the default) as well as diaglevel 4. With the level of detail provided, one can get a very clear picture of where each thread spent its time while the operation was running.

7.Also when the Backup is used in compression mode, then also CPU gets utilised a lot . An advantage is space gets saved.

8.Over all to be precise use the backup method based on the circumstanced so that the CPU hogging is avoided.

9. With the minimal of the CPU hogging, the database performance remains intact.

10. Db2 backups are needed and vital, but also it is decision ti make how clever we use this utility so that it cannot hog the CPU resource.

Figure 1 is a graphical representation of the backup process model.

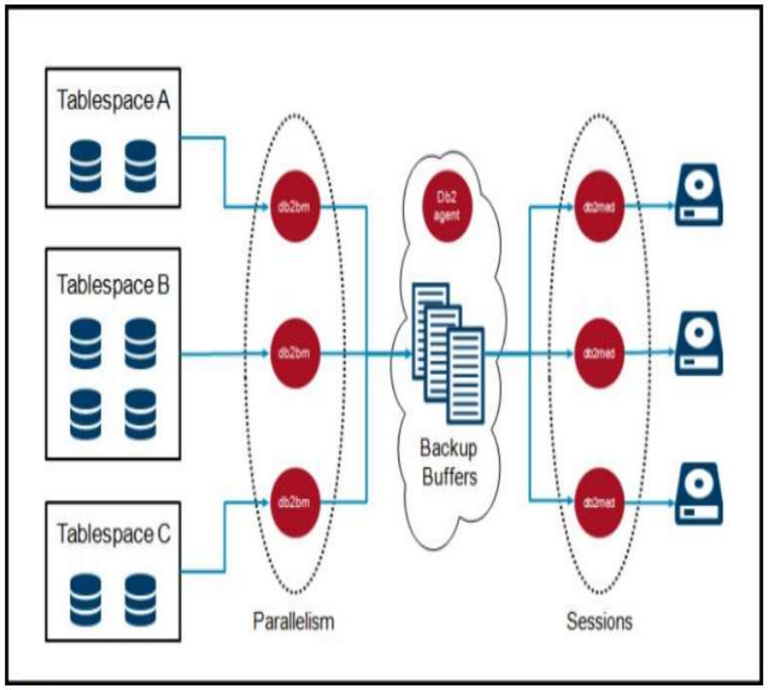


Figure 1. Backup process model

IBM

DB2 LUW

# Thank you, Any Questions