DB2 V11 Performance enhancement with autonomic free space management

Frances Villafuerte IBM francesv@us.ibm.com
Maryela Weihrauch IBM Weihrau@us.ibm.com

Session Code: A7
Oct 15,2013 15:15-16:16 | Platform: <DB2 for z/OS>

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988

Agenda

- Performance concern
- Function and usage of Free Space
- Free Space management prior to V11
- V11 enhancement in Free space management
- Case study
- Early performance evaluation result
- Things to think about

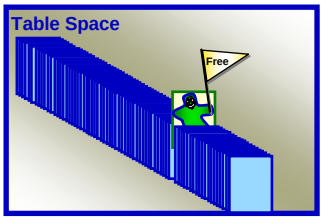
Performance concerns

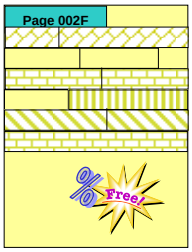
- Most common performance concerns
  - High CPU usage due to space searching
  - High get page during the insert/update/query transaction
  - High I/O accessing – data is disorganized
  - Cluster key ratio
    - Poor clustering ratio can result in less effectiveness of data prefetch for sequential fetch
    - More frequent REORG to restore clustering key ratio
- Considering tuning techniques for optimizing performance
  - Balancing between space reuse and performance
  - Better management for cluster key range
  - Better management for index keys
  - Better management on free space

Free space management is one of the many techniques on performance turning. This is the main topic for this session.

Table Space attributes for free space

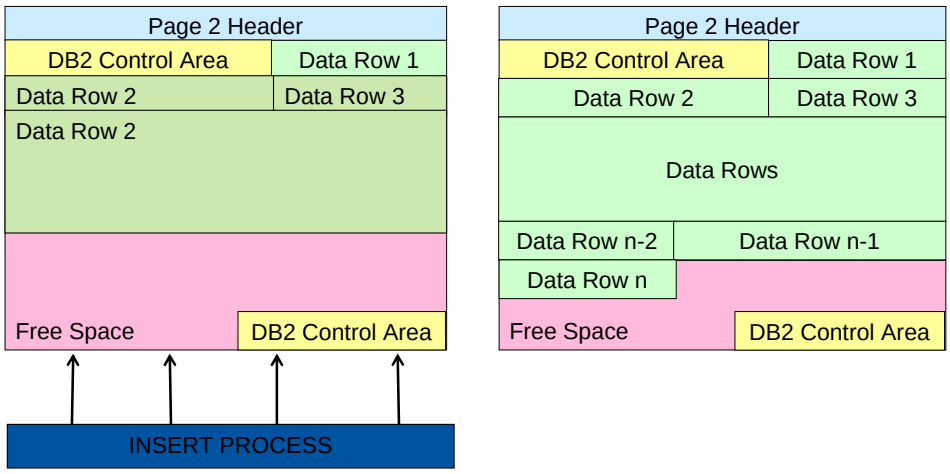
- Table space attributes
  - Create table space, alter table space
- FREEPAGE n
  - DB2 leaves a page free space every n pages during LOAD or REORG
- PCTFREE n
  - DB2 leaves percentage of free space on each page during LOAD or REORG
  - N can be 0 – 99
  - The default is PCTFREE 5 % for data page (PCTFREE)





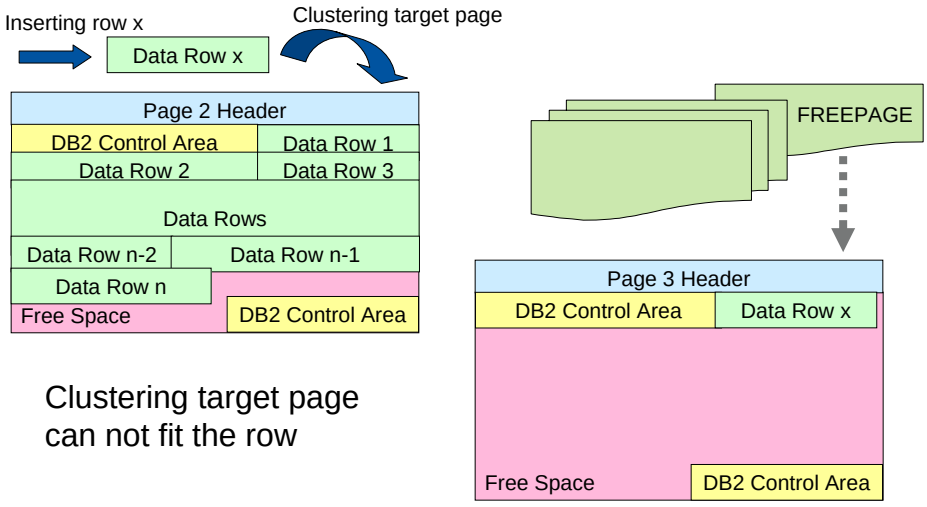
Free Space (PCTFREE) usage for INSERT

- One of the tuning technique for enhancing insert performance
- Use free space to keep data in the clustering range, reduce frequency of REORG



5

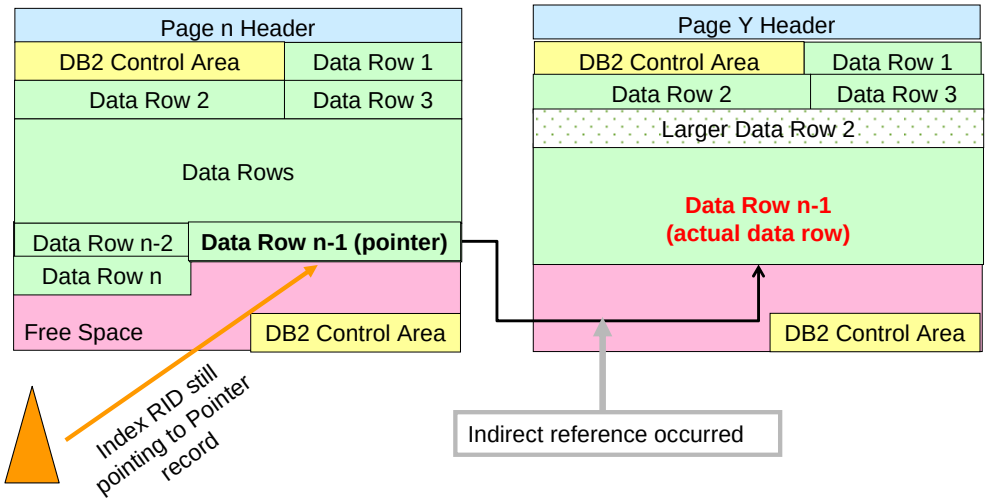
Free Space (FREEPAGE) usage for INSERT



Clustering target page can not fit the row

Free Space Usage (PCTFREE) for Update – Indirect Reference

- UPDATE to a record that increases the row length
  - no room to fit back to the same page creates indirect reference



7

As there is insufficient space then an overflow record is created

NOTE That there have been changes in this process with DB2 V11. However, large amount of overflow record could cause performance degradation.

The Common Cause Of Indirect Reference

- Rows with VARCHAR columns
  - Like rows are inserted with nullable columns and later updated with actual values
- Compressed rows
  - Even when all columns are fixed length these really are variable
- How to recognize indirect reference
  - Using Real Time Statistic (RTS) value
    - REORGNEARINDREF & REORGFARINDREF
    - TOTALROWS
    - Ratio between (REORGNEARINDREF + REORGFARINDREF) and TOTALROWS can provide hints on the affect of indirect reference

Side effects of indirect reference

- Side effect of update activities creating indirect reference
  - More overhead of accessing data
    - Two page accessing
    - More serialization
  - Performance degradation
    - Extra I/O affects query performance
  - Require a complex free space management to reduce indirect reference
    - No external keyword to reserve more free space just to accommodate update activities
  - Require frequent REORG to remove indirect reference

Benefit with adequate and distribute free space

- Benefit with adequate and distribute free space – PCTFREE and/or FREEPAGE
  - Better clustering ratio for performance
  - For efficient sequential read of data via clustering index
  - For efficient sequential read of index
  - To minimize index split
  - Improve prefetch processing
    - Reduce # of I/O's accessing

Free Space Management prior to V11 - 1

- Free space controlled by
  - Defined on partition or table space level
    - CREATE TABLESPACE or ALTER TABLESPACE
- Reserved during LOAD or REORG processing
  - Page is marked full when reaches the designated percentage
  - Require frequent REORG of table space to remove indirect reference
- Used by insert but not update
  - Insert Can consume all free space whenever it can
  - Leaves nothing for UPDATE processing
- No easy way to managed free space specific for update only
  - Cause more indirect reference
- Common technique used for Insert performance
  - Search for available space near the optimal page to store data rows in clustering index sequence
  - Reduce CPU caused by exhaustive space search during the insert

11

DB2 Sequoia solution of Free Space Management

- DB2 11 improves free space management for UPDATE transaction
  - Capable of reserving free space for UPDATES within a page
  - Capable of managing free space to reduce indirect reference
  - Autonomic option for estimating free space
    - Capture history of UPDATE and INSERT behaviour in RTS table
- Benefits
  - Reduced REORG requirements
  - Improve query and Update performance
    - Reduced indirect reference
      - Reduced # of I/Os
      - Reduced locking
      - Reduced CPU

12

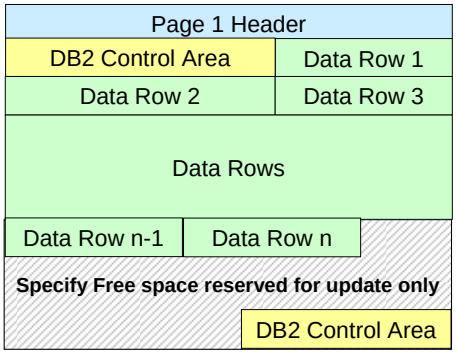
Autonomic option utilizes RTS statistics can better estimate optimal amount of space to reserved for future update.

DB2 V12 Managing Free Space for Update

- Available in V11 NFM
- Supports all types of table space **except**
  - Hash table
  - LOB
  - XML
  - Workfile
- Free space for update is effective if ….
  - Table has variable length column or compressed
  - DB2 automatically ignores the specified free space if no variable length column
  - DB2 ignores specified free space if table has Valid/Edit proc

DB2 V11 Free Space Control For Update

- New keyword **PCTFREE FOR UPDATE**
  - DB2 will stop inserting rows if the free space reaches this value
  - Page will be marked full on the space map page, preventing further inserts
  - Space will only be used by updating the record on the page
    - Overflow record will be created if not enough space
    - Overflow record will NOT occupy space reserved for UPDATES



14

Setting PCTFREE FOR UPDATE Value - DDL

- CREATE TABLESPACE statement
  - Specify at the table space level or partition level
  - The value at table space level plus value at partition level cannot exceed the allowable space of the page
  - DB2 automatically adjusts the value to allow at least one row to be inserted
- ALTER TABLESPACE
  - Setting PCTFREE FOR UPDATE on existing table space
  - Used to readjust the value
- ZPARM – system sub parameter
  - PERCENT FREE FOR UPDATE (PCTFREE_UPD)
  - Possible value :
    - **Auto** ➔ indicates autonomic solution by DB2
    - Range from 0 – 99 percent value of the page size

15

Setting PCTFREE FOR UPDATE Value - DDL

- Syntax
  - CREATE/ALTER TABLESPACE .. PCTFREE <X integer>
  - CREATE/ALTER TABLESPACE .. PCTFREE x **FOR UPDATE** <Y integer>
  - CREATE/ALTER TABLESPACE .. PCTFREE **FOR UPDATE** <Y integer>
- PCTFREE <X integer>
  - Traditional PCTFREE parameter honored by REORG/LOAD
  - Can still be used for the managing clustering purpose
  - INSERT transaction still does not honored this value
- **FOR UPDATE** <Y integer>
  - Percentage of space reserve on each page for update transaction

DB2 V11 Managing Free Space for Update

CREATE TABLESPACE … PCTFREE <X integer> FOR UPDATE <Y integer>

or

ALTER TABLESPACE … PCTFREE <X integer> FOR UPDATE <Y integer>

- **FOR UPDATE** <integer>
  - Percentage of free space to be left by INSERT processing or DB2 utility
  - Reserved space can only be used by UPDATE processing
  - Range -1 to 99
    - 0 means no space is to be reserved for UPDATE
    - Special value -1 means autonomic option, space determined by DB2
- Specifying values for both **X** and **Y** value
  - Can be specified at partition or table space level
    - Table space acts as default for partitions
  - **X** + **Y** cannot exceed 99%
    - Any combinations of the two values – defaults, partition, table space
    - DB2 will scale back Y value to meet requirements

17

DB2 11 Managing Free Space Examples

1. Create Tablespace TS1
   PCTFREE 60 FOR UPDATE 50…
   PART 1 PCTFREE 10

   ← Sum of x+ y > 99 SQL -644

2. Create Tablespace TS2
   PCTFREE 20 FOR UPDATE 30
   PART 1 PCTFREE FOR UPDATE 80

   ← Sum of x+ y > 99 SQL -644 (table space level + partition level)

3. CREATE Tablespace TS3
   PCTFREE 50

   ←

4. CREATE Tablespace TS4
   PCTFREE 10…….
   PART 1 PCTFREE 50
   PART 2 PCTFREE 30

   ←

ZPARM value will be used for the default.
PCTFEE FOR UPDATE value is scale back if the sum of ZPARM value and PCTFREE value is more than 99%

18

Setting PCTFREE FOR UPDATE Value - Autonomic option

- DB2 uses RTS values to estimate PCTFREE FOR UPDATE value
- Special value -1 is specified

CREATE/ALTER TABLESPACE .. PCTFREE n **FOR UPDATE** -1

OR

ZPARM panel with AUTO option

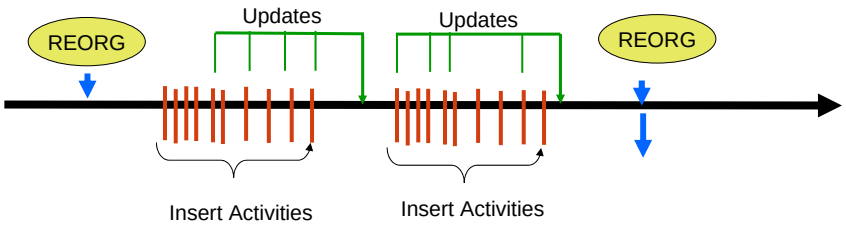
```
DSNTIP71 INSTALL DB2 - SQL OBJECT DEFAULTS PANEL 2
===>
Enter general options for table spaces and indexes below (continued):
1 DEFAULT PARTITION SEGSIZE===> 32  For explicitly-created table spaces
                                    (0, 4, 8, 12, ..., 56, 60, or 64)

2 PERCENT FREE FOR UPDATE ===> AUTO Percent of space per page reserved
                                    for update (AUTO or 0 - 99)
```

```
PRESS: ENTER to continue RETURN to exit HELP for more information
```

19

How Free Space Autonomic option works

- Percentage of free space for UPDATE is calculated between two REORG interval
  - Base on INSERT / UPDATE activities ratio between two REORG interval, DB2 estimates the free space that should be reserved for UPDATE transaction
  - Calculation is done during the INSERT
  - Free space is not reserved until sufficient activities for the calculation
    - Space is reserved on the new created data page



To alter the reserve space on the existing page, run REORG or LOAD utility

Autonomic option – RTS Columns

- RTS Columns
  - UPDATESIZE – a new interval counter between REORGs
    - Accumulated bytes that have been added or removed
    - Reset to zero by REORG or LOAD REPLACE
    - Value is tracked and externalized to RTS in NFM
  - REORGUPDATES (existing column)
    - Shows number of rows updated since last REORG or LOAD REPLACE
  - Columns used for estimating the percentage (at partition level)
    - UPDATESIZE – delta updated bytes
    - REORGUPDATES - # of updated rows since last REORG
    - REORGINSERTS - # of inserted rows since last REORG
    - DATASIZE – Total data size for all rows exist in the table space
    - TOTALROWS – number of rows in the table space

Managing Free Space with REORG & LOAD

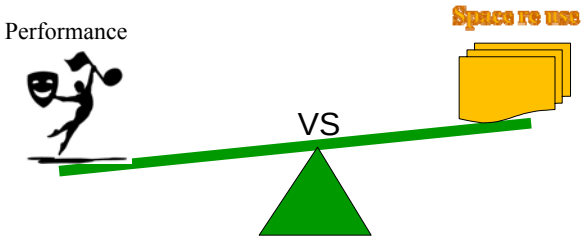
- Always use RTS value to calculate optimal percentage for update
  - Base on the total number of rows, size of rows, update activities of entire table space to estimate the percentage for update
  - Updates PCTFREE_UPD_CALC column in SYSTABLEPART
    - A new column in SYSTABLEPART table
    - Initial value = -1 after migrated to V11
    - Only REORG/LOAD sets this value
    - Value will only be used by INSERT if autonomic option is used, otherwise specified PCTFREE FOR UPDATE is used.
  - Message if RTS values do not exist for the object
  - Total space reserved on the page after REORG will be the sum of both free space definitions (PCTFREE)
    - PCTFREE for insert and PCTFREE_UPD_CALC or PCTFREE FOR UPDATE

22

Managing Free Space After REORG/LOAD

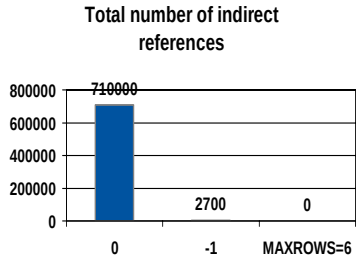
- Subsequence insert after utility
  - Respects to the value specified by PCTFREE FOR UPDATE
  - If autonomic option
    - Insert reserve space base on PCTFREE_UPD_CALC from REORG
    - Insert transaction continue to calculate PCTFREE value periodically
    - Readjust if the new calculated value > PCTFREE_UPD_CALC value
- ZPARM REORG_IGNORE_FREESPACE = YES
  - An existing ZPARM
  - REORG will not reserve either PCTFREE attributes
  - REORG will still calculate PCTFREE_UPD_CALC

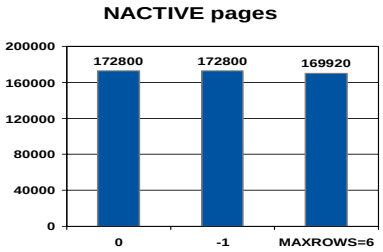
Performance Evaluation
Balance between space reuse and performance



Early performance evaluation

- Early performance evaluation
  - 100 insert/commit/100 Update/commit, 1 millions rows
  - Increasing row size from ~450 bytes to ~ 600 bytes
  - This test case example indicates autonomic option reduces the number of the indirect reference significantly without increasing additional space

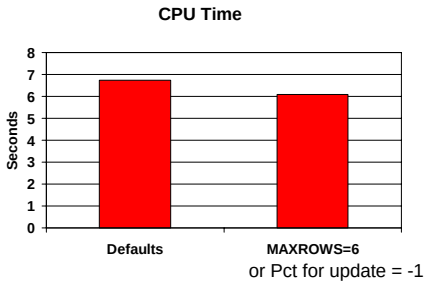




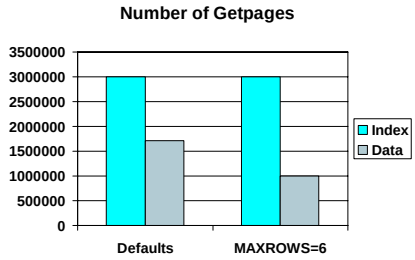
Note: The optimal space use for the average size of row in this example is approximately 6 rows.

Early performance evaluation - Query

- 1 million random selects with setup table space of :
- 10 million rows, 3 level index
- 71.7% of rows have indirect references without reserving space for update
- MAXROWS=6 eliminates all of them base on the row size in this example. The outcome is comparable when using autonomic option of PCTFREE FOR UPDATE = -1
- Result: Indirect references cause the class 2 CPU time to increase by 10.6%



or Pct for update = -1



or Pct for update = -1

MAXROWS

- Controls number of row stored into a page
- Similar way of managing data flow like free space management
- Frequently used for INSERT performance
- Rarely used for update activities because hard to estimate
  - Needs deep understanding of the data
  - Successful rate dependents on row size
  - If the distribution of data from day to day activities remains consistently, then MAXROWS is an effective way to manage free space
  - If the distribution of data is not consistently from day to day activities, MAXROWS becomes more difficult to use

MAXROWS vs PCTFREE FOR UPDATE

MAXROWS

- Good for consistent row size
- Base on number of rows can fit in a page
- Must manually and carefully estimate 'average' row size and number of 'average' size rows that can fit comfortably in a single data page
  - Estimate value base on RTS
- Hard to adopt when table schema is changed
- Frequent monitoring and tuning the value for optimal performance

PCFREE FOR UPDATE

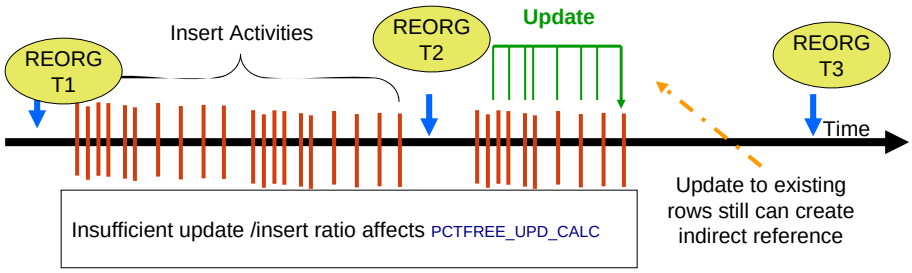
- Good for either consistent or inconsistent row size
- Base on the percentage of space usage
- Provides more flexible free space distribution
- Estimated value base on RTS
  - Data size, number of rows and avg row size
- Easily adopt when table schema is changed
- Autonomic options in V11 NFM
- **Workload dependent**

Table schema changes is referring to activities such as adding a new column or changing the size of the column. In this type of

Activities can change the average size of the row. In this case, if maxrows is used to reserve the space, a new maxrows value needs

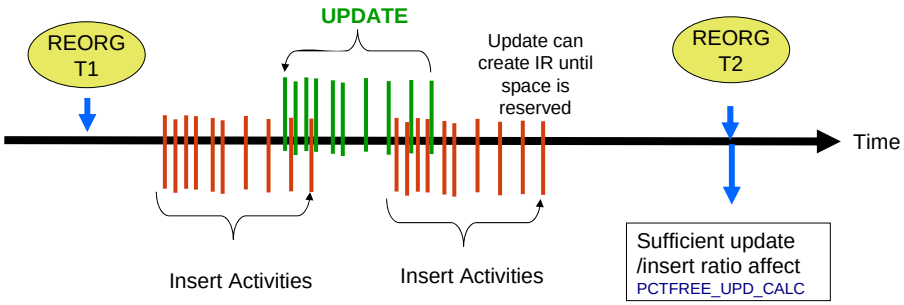
To be adjusted accordingly.

Case Study – Unbalance of INSERT and UPDATE workload



- Candidate for using explicitly define PCTFREE FOR UPDATE
  - Space is reserved regardless
- With autonomic option
  - Space will not be reserved adequately due to unbalance of insert verse update ratio between two REORG interval
- Number of overflow records depends on specification
- Continue monitoring and tuning base on RTS history for optimal result

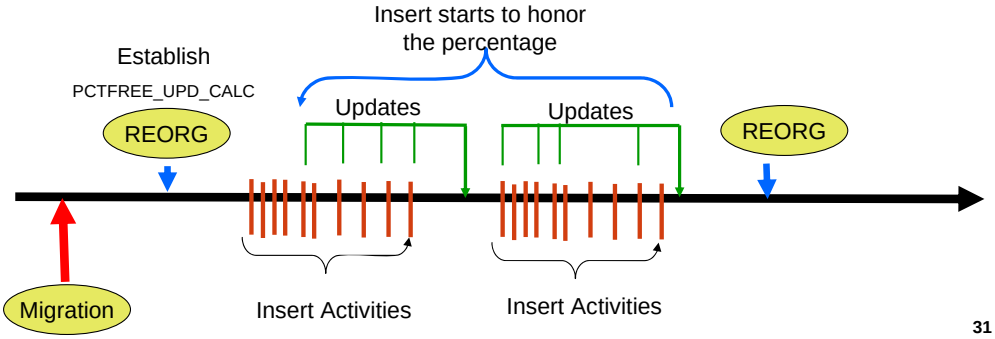
Case Study – Proper ratio of INSERT and UPDATE



- Good candidate for autonomic option
  - Spaces will be reserved as history of update activities are accumulated
  - Overflow record will continue to occur until space is reserved
- REORG provides better estimate value
  - Better distribution of free space
  - Insert if flexible to adjust estimated value upward if workload is changed

Observation After Migration to V11 NFM

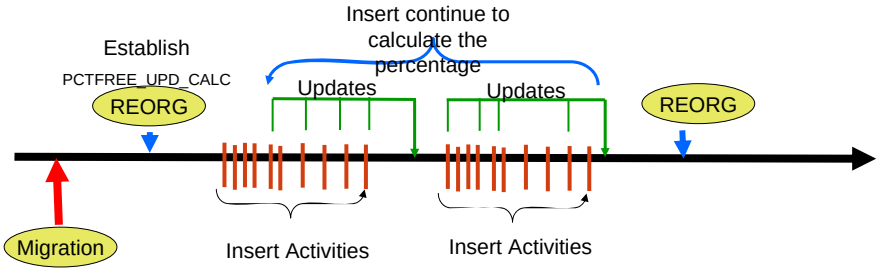
- INSERT transaction starts honoring space for update
  - ALTER or CREATE
- The value of RTS columns
  - The new column UPDATESIZE is created and initialized to null during migration
  - Rest of RTS columns has values base on last REORG
  - UPDATESIZE value is not at the same consistent point with rest of RTS columns



31

Observation After Migration to V11 NFM continue

- For existing object
  - UPDATESIZE value is initialized to null
  - The column UPDATESIZE did not have history prior to migration
  - No space will be reserved until INSERT/REORG have adequate value to calculate adequate reserve size
- For newly created object, the value is initialized to zero
  - Value will be accumulated after CREATE DDL
- May take couple REORGs to normalize the estimate value
- Insert continue to calculate the percentage if autonomic option



Things to think about

- Evaluate the trade off between performance and space reuse
  - Could possible use more space with free space management
- If reserved space is defined too high, space is wasted
- Insert performance is degraded if the PCTFREE for UPDATE is not adequate
- The pattern of insert, update and commit behavior can affect the space reuse for update if autonomic option is used.
- Possible less clustering ratio
  - The reserved space is used by update can not be reused for insert to ensure clustering order
  - Insert degradation can be a trade off from the improvements of query performance
  - Candidate page does not work, more time spend on space search
- Require automation of RTS monitoring procedure to adjust space usage

Frances Villafuerte
IBM
Francesv@us.ibm.com
Maryela Weihrauch
IBM
Weihrau@us.ibm.com

Session : A7
Title : DB2 V11 Performance enhancement with autonomic free space management