Db2 13 Utilities

IBM Easy Tier Exploitation
Enhanced Space Level Recovery using Part Level Image Copies
REPAIR WRITELOG for Compression Dictionary
REORG INDEX NOSYSUT1
Inline Stats Page Sampling
Utilities History

IBM Easy Tier Exploitation

FL 100

Description

– IBM Easy Tier: no-charge feature in the DS8xxx and above system storage
  - Provides a performance enhancement by shifting data placement based upon access frequency
    – Frequently accessed to onto SSDs
    – Infrequently accessed to HDDs
  - Easy Tier dynamically moves datasets to help maximize overall "application" performance across the storage system

Db2 Behavior

– REORG Utility automatically exploits Easy Tier via a utility update applied in the maintenance stream starting in Db2 12 for z/OS
  - No changes are needed by the Db2 user
  - REORG utility automatically leverages Easy Tier for placement of Target/Shadow tablespaces

Benefits:

– Improved REORG performance

– Consistent application performance across REORGs as target inherits source placement

Reference: https:/ www.ibm.com/docs/en/db2-for-zos/12?topic=release-integration-ds8870-easy-tier-multi-temperature-management.

Redpapers:

- IBM DS8000 Easy Tier (Updated for DS8000 Release 9.0) https:/ www.redbooks.ibm.com/abstracts/redp4667.html
- DS8870 Easy Tier Application https:/ www.redbooks.ibm.com/redpapers/pdfs/redp5014.pdf

Enhanced Space Level Recovery using Part-Level ICs

FL 100

Db2 TS Recovery Challenge

– When doing a recovery and using the default DSNUM ALL – for TS, IS and IXs – an error message is issued

  DSNU512**I** (DATASET LEVEL RECOVERY IS REQUIRED)

  - Frequently caused when DSNUM ALL without LISTDEF used when only copies available are partition or piece level
  - Objects with these errors are not recovered and the RECOVER ends with RC8

Changes

– RECOVER supports TS-level recovery for all UTS types even if the ICs were taken *at the part or piece level*, including UTS base tables for:
  - XML
  - Auxiliary indexspaces or indexes over XML UTS
  - LOB tablespaces
  - Auxiliary indexspaces or indexes over LOB tablespaces

– RECOVER to CURRENT or point-in-time maximizes use of part-level ICs and uses other IC datasets where part-levels not available (especially when DSNUM ALL used without a LISTDEF)

– Promotes greater use of part-level in-line copies by REORG and LOAD utilities providing greater efficiency

– New message when recovery is processed at the PART or piece level:
  - *DSNU1576I RECOVERY OF object-type object-qualifier.object-name PROCEEDS AT THE PARTITION OR PIECE LEVEL*

REPAIR WRITELOG for Compression Dictionary support

FL 100

Use Case

- Replication products read Db2 log records for source tables, then replicate the SQL INSERT, UPDATE and DELETE operations to the target tables
  - Log data is retrieved using IFCID 306
  - Compressed data requires the compression dictionary for Db2 to decompress the data
    - The current dictionary is always on the source table
    - The prior old compression dictionary may be needed by Db2 under certain circumstances to decompress data
      - » IBM Utilities (REORG/LOAD) write the old dictionary to the Db2 log (e.g. when new dictionary is built)\
      - » ISV utilities have been unable to record the OLD decompression dictionary to the Db2 log.
    - ISV Utilities could cause replication products to fail and/or force an automatic refresh of the target object.

Enhancement

REPAIR WRITELOG has New values for TYPE and SUBTYPE

- ISV products can now write old decompression dictionary log record, up to the maximum log record size supported by Db2
- REPAIR utility returns message DSNU3335I noting location of log record with old decompression dictionary
- ISV utility can then insert a record into SYSIBM.SYSCOPY noting the log record location

Enables replication products to avoid errors for log records that require the old decompression dictionary when using ISV Utilities.

REORG INDEX NOSYSUT1 – New Default

FL 500

Use Case

- The REORG utility can improve performance by eliminating the work data set by processing the index keys in memory
  - REORG will benefit when the utility leverages subtask parallelism to unload and build the index keys concurrently
  - Introduced in Db2 12 by:
    - New NOSYSUT1 syntax keyword for REORG INDEX
    - Updating the REORG_INDEX_NOSYSUT1 subsystem parameter which is online changeable
    - "PARALLEL num-subtasks" can specify degree of parallelism; however, not required as REORG INDEX determines optimal degree based on runtime resource calculation

Recommended V12 Setting: YES

Changes

- The NOSYSUT1 will always be used in Db2 13 for REORG INDEX types:
  - REORG INDEX SHRLEVEL REFERENCE
  - REORG INDEX SHRLEVEL CHANGE
- At FL500, the only setting for subsystem parameter REORG_INDEX_NOSYSUT1 is YES
- Benefit is automatic once FL500 reached – no changes to REORG utility jobs required

REORG INDEX NOSYSUT1 – New Default

FL 500

REORG INDEX syntax diagram changes

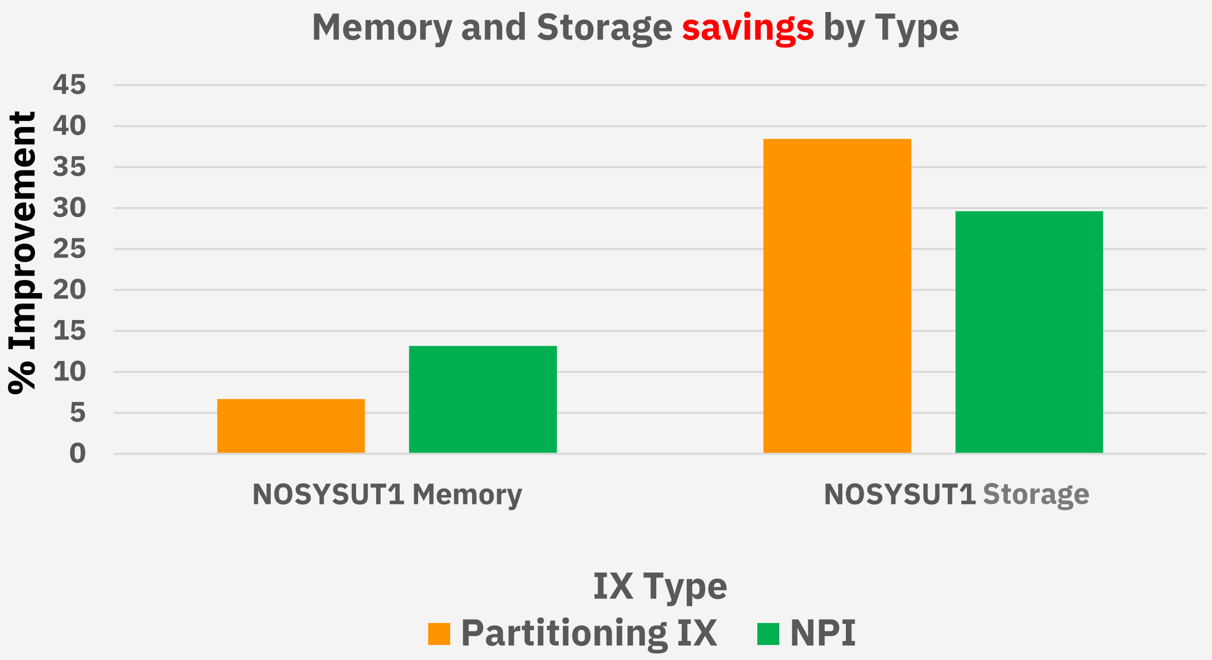
```
>--REORG INDEX----…

   |--NOSYSUT1--|
>--|------------|-----|-------------------------|---
                      |--PARALLEL num-subtasks--|
```

Multi-task and parallel information

- NOSYSUT1 will enable parallel subtasks to unload and to build the index keys in its internal processing.
- The index keys can be processed concurrently by parallel subtasks, as well as operate on different physical partitions of the target partitioned index (PI or DPSI)
- The optimal degree of parallelism based on available system resources at run time, with at least one index unload task and one index build task

- The NOSYSUT1 option is ignored if SHRLEVEL NONE is specified
- For SHRLEVEL REFERENCE or CHANGE execution, NOSYSUT1 option is always enforced.
- Should an error occur during index keys unload or build, the user needs to perform a phase-restart of the utility from the beginning of the UNLOAD phase.
  - This is only for SHRLEVEL REFERENCE since SHRLEVEL CHANGE is not re-startable until the SWITCH phase

REORG INDEX NOSYSUT1 – New Default



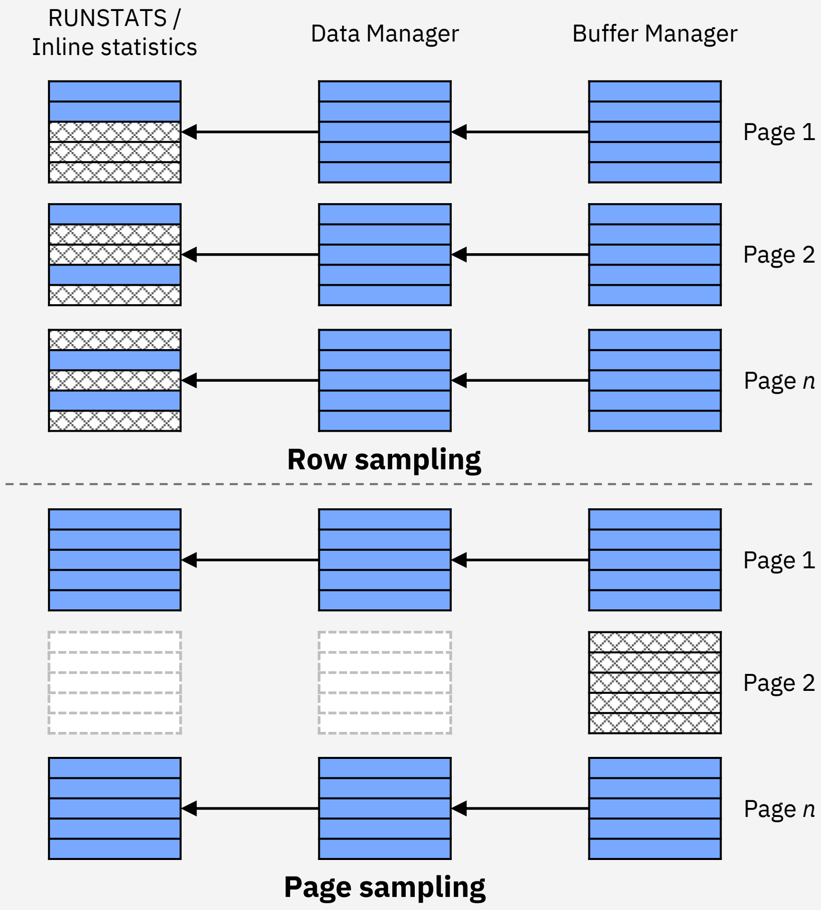
With this new enhancement performance measurements show significant improvements:

- Reductions in Memory Requirements
- Significant reductions in storage requirements
- Partitioned indexes show up to an 86% improvement in elapsed time, and CPU reduction can be even higher
- Tasks are now zIIP eligible providing reductions in non-zIIP CPU of between 50-95%

Page Sampling support for Inline Statistics

FL 500

- Db2 10 introduced page sampling for RUNSTATS to reduce CPU consumption and elapsed time
- Since Db2 12, page sampling is the default for RUNSTATS
- Db2 13 now supports page sampling for inline statistics collected by LOAD REPLACE and REORG TABLESPACE
- New keyword **TABLESAMPLE SYSTEM** added to LOAD REPLACE and REORG TABLESPACE syntax
  - AUTO: let Db2 determine the sampling rate
  - *Numeric value*: user-defined sampling rate (0 < *n* ≤ 100)
  - NONE: do not use page sampling, instead use the value specified by the SAMPLE keyword
- ZPARM STATPGSAMP introduced in V12R1M505, that sets the system-wide default for page sampling, is extended to inline statistics for V13



Utilities History Table – overview

FL 501

Motivation

- Running, monitoring and optimizing utility executions is an important part in the daily management of Db2
- Db2 does not provide an easy way to obtain utility execution history and statistics in real time
- It is required to retrieve and scan the utility job outputs to get the relevant information
- Users should be able to easily retrieve utility execution history and statistics without any additional tooling or processing

Solution

- New catalog table **SYSIBM.SYSUTILITIES**
- New ZPARM **UTILITY_HISTORY NONE | UTILITY** to control history collection for each data sharing group member
- New column **EVENTID** in SYSIBM.SYSCOPY
- Uses IFCID 25 trace record statistics (elapsed, CPU, zIIP, sort CPU, sort zIIP times)
- Multi-step implementation
  - Today: collection of utility-level information
  - Future: collection of object-level and utility phase-level information

Utilities History Table – normal flow

FL 501

```
/ DB2COPY JOB DB2ADM …
/ STEP1 EXEC DSNUPROC,UID='COPYTS' …
/ SYSIN DD *
LISTDEF COPYLIST
         INCLUDE TABLESPACE DSN8D13A.DSN8S13E
         INCLUDE TABLESPACE DSN8D13A.DSN8S13D
COPY LIST COPYLIST …
```

1 When the utility driver begins execution, a row is INSERTed

EVENTID	NAME	JOBNAME	UTILID	USERID	STARTTS	STARTLOGPOINT	CONDITION
1001	COPY	DB2COPY	COPYTS	DB2ADM	2022-04-05 13:26	…001F8C16A04…	*blank*

2 After utility-in-progress states are set, the row is UPDATEd

EVENTID	NUMOBJECTS	LISTNAME
1001	2	COPYLIST

3 When the utility terminates, the row is finally UPDATEd

EVENTID	ENDTS	ELAPSED TIME	CPU TIME	ZIIP TIME	SORT CPUTIME	SORT ZIIPTIME	RETURNCODE	CONDITION
1001	2022-04-05 13:28	418295	22910	0	0	0	0	E

Remark: The table columns and data is simplified for display purpose.

Utilities History Table – special cases

FL 501

When a utility ABENDs, the row is **not** updated. The utility is in a stopped state.

EVENTID	ENDTS	RETURNCODE	CONDITION
1002	NULL	NULL	*blank*

Issue –DIS UTIL command to determine if active or stopped

When a utility is RESTARTed, the corresponding row is UPDATEd like this:

EVENTID	RESTART	JOBNAME	USERID	GROUP_MEMBER
1002	Y	blank		DSNB

When a utility completes after RESTART, the row is finally UPDATEd like this:

EVENTID	ENDTS	ELAPSEDTIME	CPUTIME	ZIIPTIME	RETURNCODE	CONDITION
1002	2022-04-05 13:28	418295	22910	0	0 | 4 | 8	E

When a –TERM UTIL or STA DB(…) SP(…) ACCESS(FORCE) command terminates a *stopped utility*, the corresponding row is updated like this:

EVENTID	ENDTS	ELAPSEDTIME	RETURNCODE	CONDITION
1002	2022-04-05 13:28	418295	NULL	T | F

ELAPSEDTIME includes the time the utility was in stopped state

When a –TERM UTIL command is issued on an *active utility*, the corresponding row is updated like this:

EVENTID	ENDTS	ELAPSEDTIME	RETURNCODE	CONDITION
1002	2022-04-05 13:28	418295	8	T

Utilities History Table – operational aspects

FL 501

- New messages are added to the utility job output
  - **DSNU3031I** UTILITY HISTORY COLLECTION IS ACTIVE. LEVEL: UTILITY, EVENTID: *event-id-number*
  - **DSNU3032I** ERROR DURING UTILITY HISTORY COLLECTION, RETURN CODE X'*return-code'* REASON CODE X'*reason-code'*
- SQL INSERT, UPDATE and DELETE are allowed on SYSIBM.SYSUTILITIES table, e.g. for cleanup processing (example in Db2 13 and More Redbook) or tools integration
- It is recommended to use ISO(UR) when querying SYSIBM.SYSUTILITIES to avoid contention
- Users can define indexes on the table as needed to optimize query performance
- Utility history information is not collected for utilities executed on SYSIBM.SYSUTILITIES table, its index and tablespace, for RECOVER or REBUILD INDEX on catalog and directory objects, for objects in a restrictive state and when executing in preview mode

Utilities History Table – Sample queries

FL 501

- *“Show all utilities that started/stopped between midnight and 6am”*
- *“Show all utilities that ended with one or more errors (RC >=8) in the last 24 hours”*
- *“Show the top 10 CPU-consuming utility executions in the last 7 days”*
- *“Show restarted utilities in active or stopped state”*
- *“Show the most recent successful execution of REORG TABLESPACE for a specific table space or REORG INDEX for a specific index space”* (joining data in SYSUTILITIES and SYSCOPY using the EVENTID column)
- SQL and more sample queries are available in the Db2 13 for z/OS and More Redbook (SG24-8527-00)