IDUG

IDUG

2024 NA Db2 Tech Conference

Demystifying Recovery - Top Questions From Customers Answered

Michael Roecken, Roger Zheng

IBM

Session Code:RAS1 | Platform: Db2 LUW

Safe Harbor Statement

IBM's statements regarding its plans, directions, and intent, including the statements made in and during this presentation, are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding future products or features is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding future products or features is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about future products or features may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance and compression data is based on measurements and projections using IBM benchmarks in a controlled environment. The actual throughput, performance or compression that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Agenda

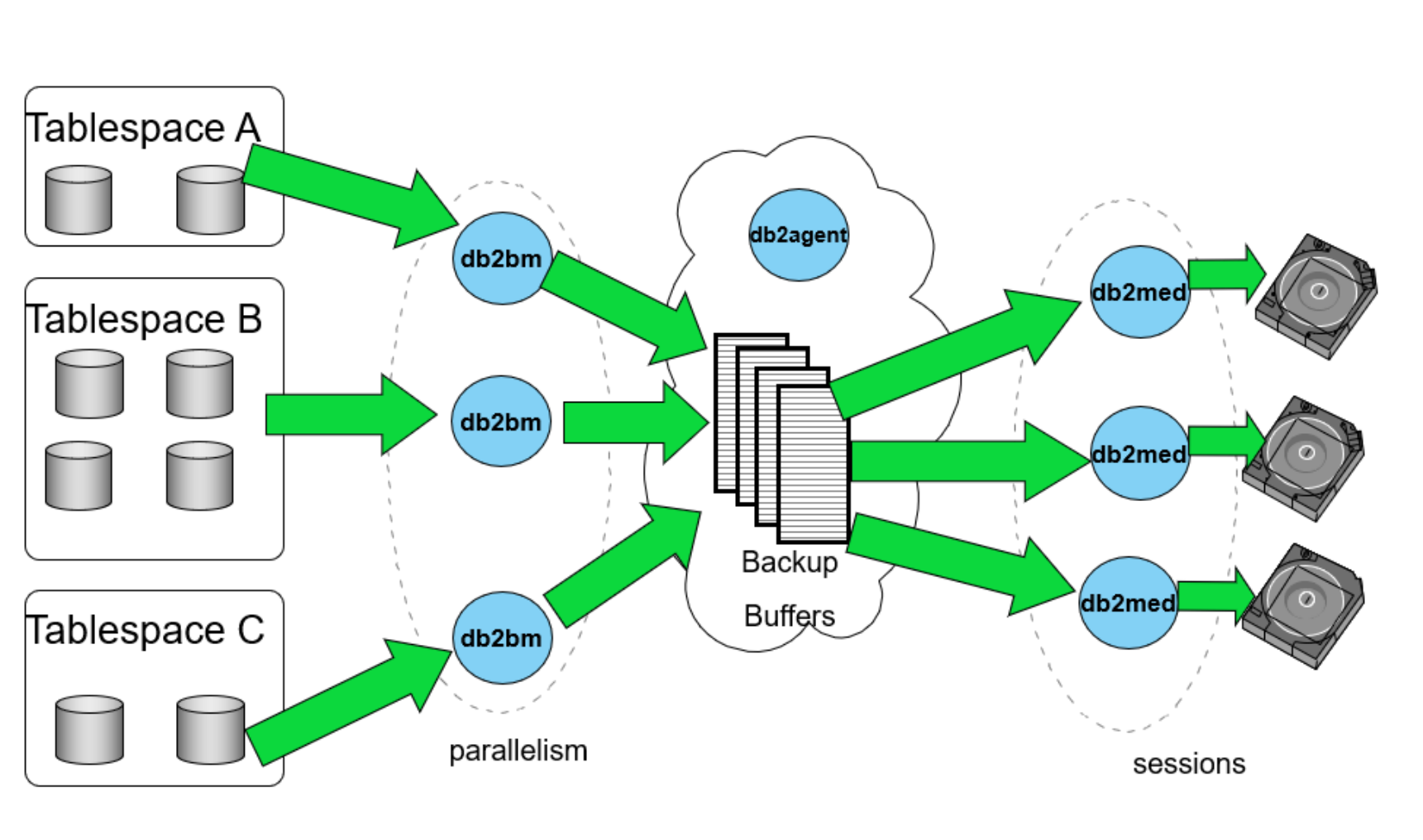
- **Backup & Restore**
  - **Performance**
  - **Object (Remote) Storage**
- Logging
  - Transaction Log Full
  - Disk Full in Archives
- Recovery
  - Point-in-time Recovery
  - Inaccessible Table Spaces
  - Monitor Crash Recovery

Question 1:

How can I identify backup performance problems?

Backup & Restore – Performance (1|10)

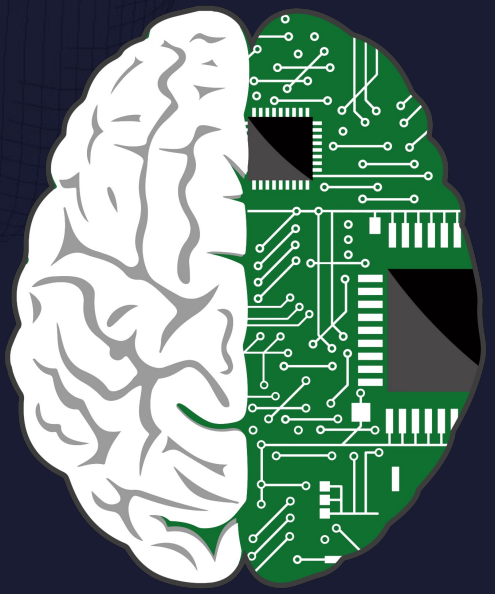
Backup Process Model



- **db2agent –**
  1 agent the drives backup
- **db2bm –**
  1 or more buffer manipulators
  - Reads data from database
  - Parallelism
- **db2med –**
  1 or more media controllers
  - Writes data to target
  - Number of targets/sessions

Backup & Restore – Performance (2|10)

- **Backup is self-tuning / autonomic**
  - If not specified, the following settings will be computed:
    - Parallelism (db2bm)
    - Buffer size
    - Number of buffers
  - All setting are dependent largely on the following settings:
    - `UTIL_HEAP_SZ`
    - Number of CPUs
    - Number of table spaces
    - Extent size
    - Page size
  - Autonomic algorithm does handle all possible options:
    - Compression
    - Data deduplication



Backup & Restore – Performance (3|10)

- **Let the database do the tuning**
  - The autonomic algorithm does a good job so start with that
  - There can always be some scenario where it may not be the most optimal
  - If you run into one of these situations, have historical data available
    - Track past performance and what parameters were used and tune until you find a sweet spot
      - ➔ BAR statistics from db2diag.log can help

Backup & Restore – Performance (4|10)

- **Some "simple" things you can do to help:**
  - **Distribute your data evenly across table spaces ****
    - Not as important in v12.1 which has Intra Tablespace Parallelism
  - **More memory ➔ increase UTIL_HEAP_SZ (suggest starting at 50,000)**
    - Affects both the number of buffers and buffer size
  - **At minimum: Number of buffers >= Number of db2bm + Number of db2med + 1**
    - Point being have enough to cover parallelism + media targets/sessions
    - If any one side becomes a bottleneck can prevent a waiting for buffer situation
      - Keep adding more until it makes no further difference
  - **Try more media targets/sessions**
    - Depending on media, sharing the load to write can help
  - **Avoid running less than ideal operations during online backup, for example:**
    - LOAD with the ALLOW NO ACCESS option
    - REORG

Backup & Restore – Performance (5|10)

- **Some "more advanced" things you can do to help:**
  - **Compression and encryption can be CPU intensive operations**
    - Less data volume to write out the better but at what cost?
    - Explore various options: table, backup, vendor, hardware
    - Try to offload these operations to hardware (NX842 / ZLIB)
  - **Big buffer pools can cause higher flush times for online backup**
    - Avoid flushes by setting registry variable `DB2_REDUCE_FLUSHING_DURING_BACKUP`
  - **To reduce time retrieving log files into backup image:**
    - Minimize long running transactions
    - Flush regularly (review settings of `PAGE_AGE_TRGT_MCR` / `PAGE_AGE_TRGT_GCR`)
    - Point `OVERFLOWLOGPATH` to archive log directory if local
  - **Still not good enough ➔ explore snapshot/flashcopy technology**

Backup & Restore – Performance (6|10)

- **Monitor in progress BACKUP (and RESTORE) using `db2pd -barstats` command**
  - Displays:
    - List of EDUs / threads involved
    - Table space details including status of queue
    - Real time performance statistics
    - Backup image size estimates
    - Progress monitor information
    - Buffer pool flush times
    - Database history file pruning times and status
    - Include logs processing details and times

Backup & Restore – Performance (7|10)

BAR Stats Example:

- Backup to vendor
- Parallelism of 10 (BMs)
- 1 session (MC)
  - Media target
- util_heap_sz 30000

```
FUNCTION: DB2 UDB, database utilities, sqluxLogDataStats, probe:2051
MESSAGE : Performance statistics
DATA #1 : String, 1414 bytes

Parallelism       = 10
Number of buffers = 10
Buffer size       = 10489856 (2561 4kB pages)

BM#    Total       I/O       MsgQ       WaitQ      Buffers       MBytes
---  --------  --------  --------  --------  --------  --------
000   1420.79     27.63   1269.31    122.62       789      7860
001   1420.74     53.90   1363.71      0.45      1683     16830
002   1420.74     51.27   1331.79     35.27      1508     15074
003   1420.74     39.19   1301.87     77.89      1118     11171
004   1420.74     32.95   1280.14    106.14       951      9507
005   1420.74     39.45   1223.28    156.55       932      9311
006   1420.74     31.31   1214.64    173.34       912      9114
007   1420.74     44.01   1260.33    114.55      1162     11615
008   1420.74     37.83   1246.14    135.36       897      8964
009   1420.74     27.64   1242.88    149.02       768      7676
---  --------  --------  --------  --------  --------  -------
TOT  14207.48    385.25  12734.14   1071.23     10720    107125

MC#    Total       I/O       MsgQ       WaitQ      Buffers       MBytes
---  --------  --------  --------  --------  --------  --------
000   1421.51   1420.33      0.41      0.00     10721     107231
---  --------  --------  --------  --------  --------  --------
TOT   1421.51   1420.33      0.41      0.00     10721     107231
```

Backup & Restore – Performance (8|10)

BAR Stats Explanation:

- **BM# - T**he number we assigned to an individual Buffer Manipulator. BM's READ data from the database's table spaces during a backup and place them into buffers.
- **MC# - T**he number assigned to an individual Media Controller. MC's WRITE buffers out to the target location.
- **Total -** The total amount of time spent by the process in seconds.
- **I/O -** The amount of time spent either reading or writing data. For the BM's this represents time reading data from table spaces and filling the buffer. For MC it is the time spent reading from a buffer and sending it to the target destination.
- **MsgQ -** This is the amount of time we spend waiting to get a buffer. For BM's it is how long spent waiting to get an empty buffer for filling. For MC's it is time spent waiting to get a full buffer in order to write out.
- **WaitQ -** Amount of time spent waiting on directives from the agent overseeing the whole backup.
- **Buffers -** The number of buffers processed by a particular BM or MC. A BM filled X number of buffers. An MC wrote out X number of buffers.
- **MBytes -** The amount of data handled by a particular BM or MC in megabytes (also kBytes, Gbytes, etc.).

Backup & Restore – Performance (9|10)

Util_heap_sz = 30000									
Parallelism = 10									
Number of buffers = 10									
Buffer size 10489856 (2561 4k pages)									
BM#	Total	I/O	MsgQ	WaitQ	Buffers	Mbytes	% Time on I/O	% time waiting for buffers (MsgQ)	% time waiting control msgs (WaitQ)
0	1420.79	27.63	1269.31	122.62	789	7860	1.94%	89.34%	8.63%
1	1420.74	53.90	1363.71	0.45	1683	16830	3.79%	95.99%	0.03%
2	1420.74	51.27	1331.79	35.27	1508	15074	3.61%	93.74%	2.48%
3	1420.74	39.19	1301.87	77.89	1118	11171	2.76%	91.63%	5.48%
4	1420.74	32.95	1280.14	106.14	951	9507	2.32%	90.10%	7.47%
5	1420.74	39.45	1223.28	156.55	932	9311	2.78%	86.10%	11.02%
6	1420.74	31.31	1214.64	173.34	912	9114	2.20%	85.49%	12.20%
7	1420.74	44.01	1260.33	114.55	1162	11615	3.10%	88.71%	8.06%
8	1420.74	37.83	1246.14	135.36	897	8964	2.66%	87.71%	9.53%
9	1420.74	27.64	1242.88	149.02	768	7676	1.95%	87.48%	10.49%
TOT	14207.45	385.18	12734.09	1071.19	10720	107122	2.71%	89.63%	7.54%

MC#	Total	I/O	MsgQ	WaitQ	Buffers	Mbytes	% time on I/O	% time waiting for buffers (MsgQ)	% time waiting for control msgs (WaitQ)
0	1421.51	1420.33	0.41	0.00	10721	107231	99.92%	0.03%	0.00%
TOT	1421.51	1420.33	0.41	0.00	10721	107231	99.92%	0.03%	0.00%

- BMs:
  - On average spent 2.71% of time waiting to read data from the database
  - Spent 89.63% of the time waiting for buffers
- MCs:
  - On average spent 99.92% of the time waiting for I/O (in this case vendor)
- Conclusion:
  - MC cannot free up a buffer until the vendor confirms it has been written
  - BMs must wait for the MCs to free the buffers
  - Bottleneck is in writing to the target device

Backup & Restore – Performance (10|10)

Solution Considerations:

1. Increase the number of media targets / sessions
    - Share the load of writing
    - Doing so may alter other resource parameters (like more BMs, number of buffers, etc.)
2. Allocate more buffers and alter buffer size through increasing util_heap_sz
    - BMs do not have to wait as long

- **As always – "it depends" – comes into play depending on other resource factors**
    - Start by changing one thing at a time and gauge
- **A blend of both will yield best result**
    - The more BMs or MCs in use the more buffers and size of buffers can matter, so having a well defined util_heap_sz can help with the balancing act

12.1 Improvements to BARSTATS

- Separate reporting of time spent due to throttling
- Separate reporting of time spent for including log files in backup image, and total bytes for log files
- Reporting time in HH:MM:SS format

Question 2:

How do I use object (remote) storage for backup, restore and log archiving?

Backup & Restore & Log Archive – Object (Remote) Storage (1|9)

- **Supported Cloud Object Storage providers using S3 protocol:**
  - IBM Cloud Object Storage
  - Amazon Simple Storage Service (S3)
  - Others: ceph, minio
- **Supported platforms:**
  - All Linux platforms **– since 11.5.9.0**
  - SuSe / RHEL Linux only before 11.5.9.0
- **Supported recovery operations:**
  - **Backup & Restore – since 11.1.0.0**
  - **Log Archive & Retrieve – since 11.5.7.0**

Backup & Restore & Log Archive – Object (Remote) Storage (2|9)

- **Local staging path is required depending on operation**
- **The default staging path is in `<instance_directory>/sqllib/tmp/RemoteStorage.xxxx`, where xxxx refers to the member number**
  - Can override with registry variable DB2_OBJECT_STORAGE_LOCAL_STAGING_PATH
  - Must be large enough to hold files from many operations such as: BACKUP, RESTORE, LOAD, INGEST, log archive/retrieve, etc.
  - Suggest move to separate storage location outside of sqllib directory

Backup & Restore & Log Archive – Object (Remote) Storage (3|9)

- **DB2_ENABLE_COS_SDK = OFF**
  - Communication uses old legacy libcurl method
  - Requires local staging path to hold temporary files and has limitations on file sizes and lack of support for streaming multipart uploads
  - Should not set as registry variable is deprecated

  - **BACKUP**: each session mas maximum image size of 5GB up to a total of 5TB

Backup & Restore & Log Archive – Object (Remote) Storage (4|9)

- **DB2_ENABLE_COS_SDK = ON [DEFAULT]**
  - Communication uses Cloud Object Storage (COS) SDK packaged with Db2 server, which removes libcurl limitations
  - File size limitations – see `MULTIPARTSIZEMB` database configuration parameter
  - Allows streaming multipart uploads, which can remove need for local staging path for some operations

  - **BACKUP**: local staging path not required; uses streaming multipart upload; each session mas maximum image size of multiplying the value of `MULTIPARTSIZEMB` by the maximum number of parts that are allowed by the Cloud Object Storage provider
    - 100MB (default MULTIPARTSIZEMB) x 10,000 parts (max allowed by S3 protocol - works for AWS and COS) = ~1TB "single upload" size
  - **RESTORE**: local staging path is required to temporarily store downloaded backup images
  - **Log archive and retrieve**: local staging path is required to temporarily store the log files being uploaded or downloaded

Backup & Restore & Log Archive – Object (Remote) Storage (5|9)

- **To interact with the object storage provider a recovery operation requires a storage access alias to be catalogued containing location and credential information**
  - CATALOG STORAGE ACCESS command
- **Set default CONTAINER clause to S3 bucket**
- **Set default OBJECT clause to act like a "sub-directory" (e.g. "backups" or "archives")**
  - By setting any one of these avoids having to specify each time when recovery operation is executed
- **Requires an encrypted keystore to hold credentials**
  - Same as what is configured for Db2 instance and native encryption

Backup & Restore & Log Archive – Object (Remote) Storage (6|9)

Example:

- **Create a local keystore:**

```
gsk8capicmd_64 -keydb -create
   -db "/home/roecken/remote/keystores/ne-keystore.p12"
   -pw "g00d.pWd" -type pkcs12 -stash
```

- **Configure the Db2 instance to use the keystore:**

```
UPDATE DBM CFG USING
   KEYSTORE_LOCATION /home/roecken/remote/keystores/ne-keystore.p12
   KEYSTORE_TYPE pkcs12
```

- **Create an alias:**

```
CATALOG STORAGE ACCESS ALIAS myAlias
   VENDOR s3
   SERVER <server>
   USER <user> PASSWORD <password>
   CONTAINER myContainer
```

Backup & Restore & Log Archive – Object (Remote) Storage (7|9)

- **The syntax for specifying a remote storage location is:**

```
DB2REMOTE://<alias>/<container>/<object>
```

- **Backup:**

```
BACKUP DB sample TO DB2REMOTE://myAlias//backups
BACKUP DB sample TO DB2REMOTE://myAlias/myContainer/backups
```

- **Restore:**

```
RESTORE DB sample FROM DB2REMOTE://myAlias//backups
RESTORE DB sample FROM DB2REMOTE://myAlias/myContainer/backups
```

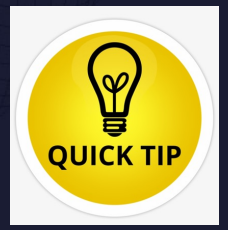
- **Log archive:**

```
UPDATE DB CFG FOR sample USING
        LOGARCHMETH1 DB2REMOTE://myAlias//archives
UPDATE DB CFG FOR sample USING
        LOGARCHMETH1 DB2REMOTE://myAlias/myContainer/archives
```

Backup & Restore & Log Archive – Object (Remote) Storage (8|9)

- **Under the covers, Db2 treats object (remote) storage targets like local storage (same type of file I/O expectations)**

- **Take care on choosing your location and size of your local storage path**
  - Concurrency of operations and size of files will dictate requirement



- **Performance impact of MULTIPARTSIZEMB is essentially none**
  - Just a way to deal with large files that are larger than what a single request can manage
    - Like: “Upload this large file in small chunks of size X”
    - Provides no parallelism benefit over and above what BACKUP offers

Backup & Restore & Log Archive – Object (Remote) Storage (9|9)

- **Performance in general comes down to data volume and speed of the network**
  - Many customers have jumped on aboard and are surprised by performance
  - Target is likely slower since it is likely off-site and public-network connected
  - Consider what “zone” from your provider is being used for the remote SERVER
  - Try enabling compressed backups / log archives

Agenda

- Backup & Restore
  - Performance
  - Object (Remote) Storage
- **Logging**
  - **Transaction Log Full**
  - **Disk Full in Archives**
- Recovery
  - Point-in-time Recovery
  - Inaccessible Table Spaces
  - Monitor Crash Recovery

Question 3:

How can I avoid transaction log full (SQL0964C)?

Logging – Transaction Log Full (1|14)

- **Maximum active log space**
  - (LOGPRIMARY + LOGSECOND) * LOGFILSIZ
- **Fixed pre-allocated active log space**
  - LOGPRIMARY * LOGFILSIZ
  - 2 log files created synchronously on start-up
  - Remainder created asynchronously after start-up
  - Consider setting registry variable `DB2_USE_FAST_LOG_PREALLOCATION`
- **LOGSECOND allocated on demand**
  - Slight performance impact

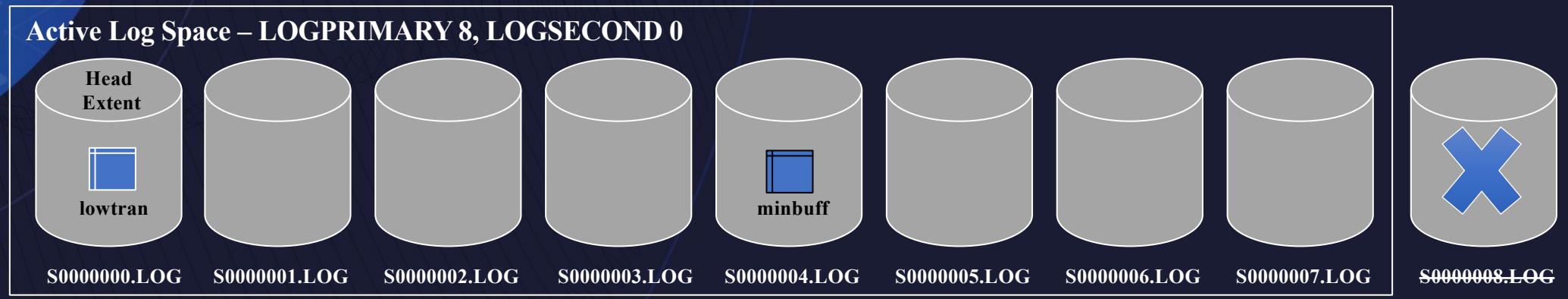
Logging – Transaction Log Full (2|14)

- **Recommendation:**
  - Prepare enough primary log files to store normal workload
- **LOGPRIMARY + LOGSECOND**
  - **Circular**: maximum 256 total files
  - **Recoverable**: maximum 8192 total files

- **lowtran**
  - First (lowest) log record belonging to oldest open transaction
- **minbuff**
  - Log record of the oldest (minimum) dirty page in buffer pool

Logging – Transaction Log Full (3|14)

- Db2 saves log files from min(lowtran, minbuff) called “First Active Log” or head extent for rollback/crash recovery
- Transaction log full is when Db2 needs to create a new log file above LOGPRIMARY+LOGSECOND but cannot because lowtran and/or minbuff do not move up



Logging – Transaction Log Full (4|14)

- **Determine reason for transaction log full from db2diag.log:**
  - **Lowtran**: Long running open transaction

```
ADM1823E  The active log is full and is held by application handle
          "0-12". Terminate this application by COMMIT, ROLLBACK or
          FORCE APPLICATION.
```

  - **Minbuff**: Bufferpool flushing not often enough

```
ADM1822W  The active transaction log is being held by dirty pages.
      Database performance may be impacted.
```

Logging – Transaction Log Full (5|14)

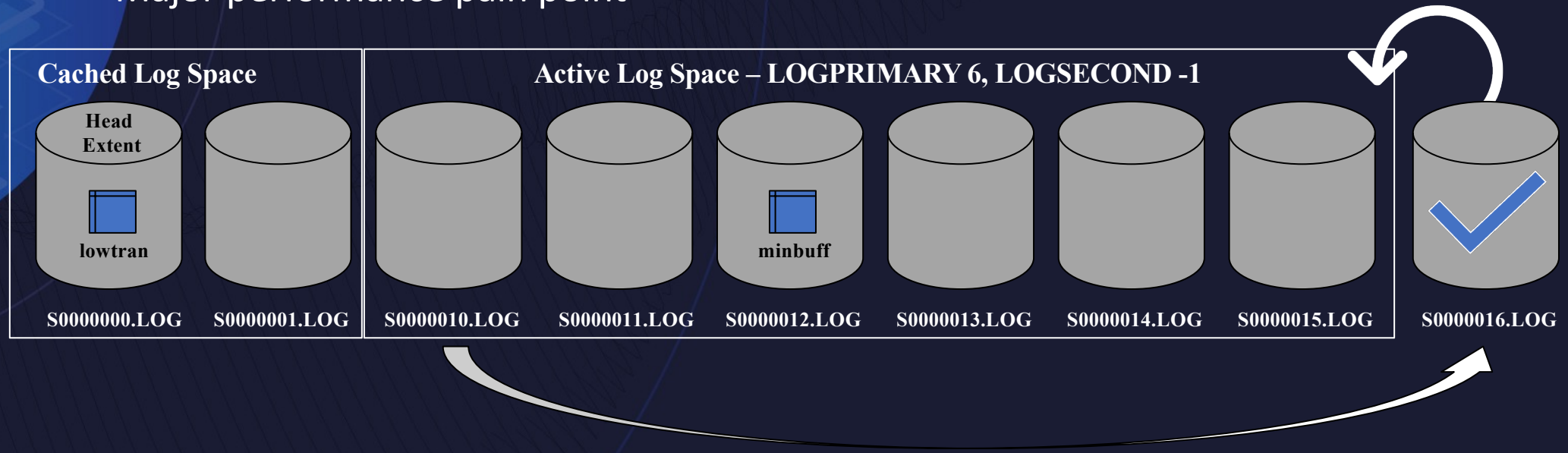
- **If lowtran:**
  - **Short Term**
    - Find offending transaction from:
      - db2diag.log
      - `APPLID_HOLDING_OLDEST_XACT` field from `MON_GET_TRANSACTION_LOG`
    - Commit or rollback or force application
    - Increase LOGSECOND dynamically
      - Maybe even using infinite logging (LOGSECOND = -1)
  - **Long term**
    - Assess whether offending application is committing often enough
    - `NUM_LOG_SPAN` for long running transactions
      - When any uncommitted transaction reaches specified number of log files, it is rolled back
    - `MAX_LOG` for large log volume
      - When any one transaction produces log data which exceeds, it is rolled back
    - **Advanced Log Space Management**

Logging – Transaction Log Full (6|14)

- **If minbuff:**
  - **Short Term**
    - Attempt to issue `FLUSH BUFFERPOOL ALL` command
    - Increase LOGSECOND dynamically
      - Maybe even using infinite logging (LOGSECOND = -1)
  - **Long term**
    - Review database configuration parameters:
      - **`PAGE_AGE_TRGT_MCR`**: time-based flushing of local bufferpool
      - **`PAGE_AGE_TRGT_GCR`** (pureScale): time-based flushing of group bufferpool
      - **`SOFTMAX`** (deprecated): log data volume flushing
    - Tune to normal workload requirements
      - Larger values keep more changed pages in memory
        - Helps runtime performance
        - But increases recovery time

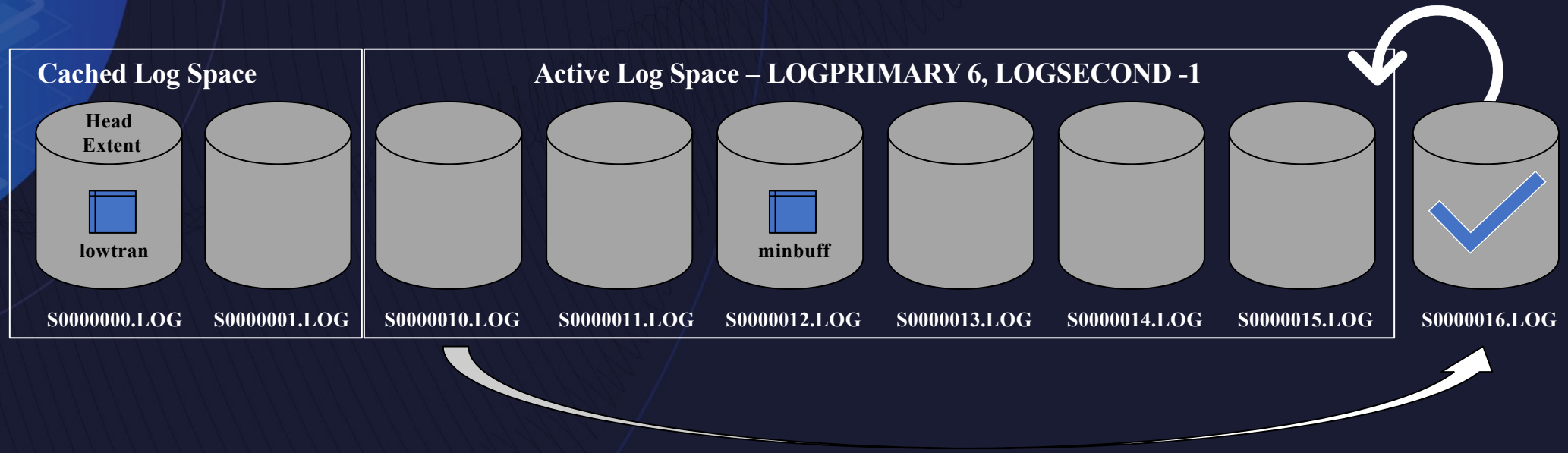
Logging – Transaction Log Full (7|14)

- **One way to avoid transaction log full is use infinite logging (LOGSECOND = -1)**
  - Files from head extent and onwards not guaranteed to be in active log path
  - Avoid rogue transactions by using configuration parameters
    - NUM_LOG_SPAN and/or MAX_LOG
  - Rollback and crash recovery may have to retrieve log files from archives
    - Major performance pain point



Logging – Transaction Log Full (8|14)

- Online backup must include many more log files
  - Increased image size
  - Longer running backups
- Db2 caches some files (up to 8) above active log space to mitigate need to retrieve log files from archives

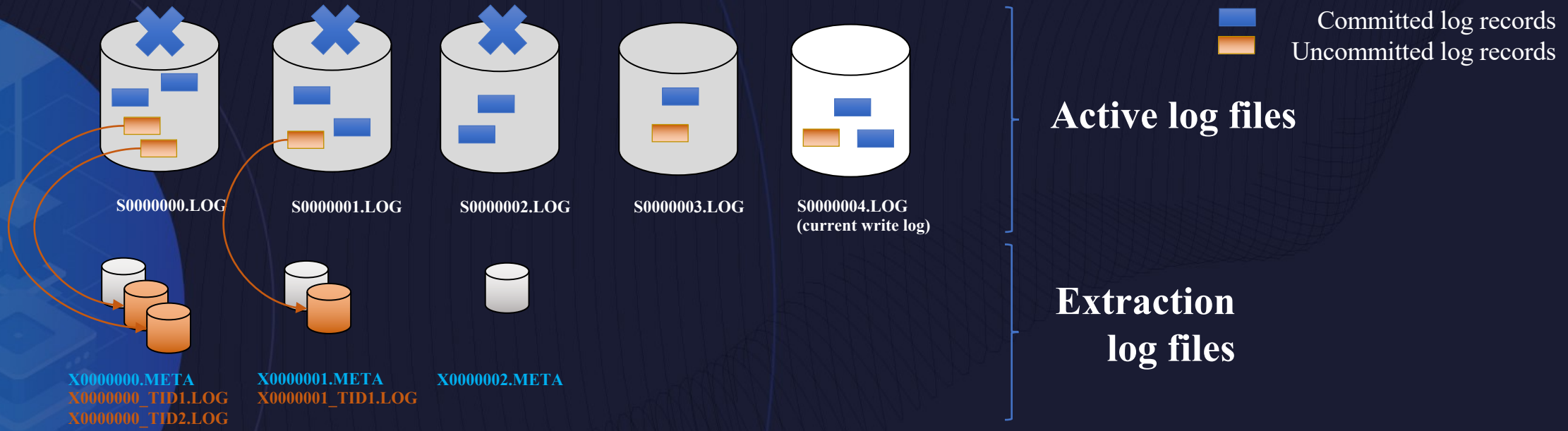


Logging – Transaction Log Full (9|14)

- Another way to avoid transaction log full is with Advanced Log Space Management

- Extract (copy) log records for long running active transactions to separate extraction log files under active log path
- Allows active log files to be closed, archived, and reused, thus avoiding transaction log full

Logging – Transaction Log Full (10|14)



- **New files in active log path:**
  - **X<logFileNum>_TID<tranId>_<tranLogStreamId>.LOG** – extraction transaction ID (TID) file. Extracted log records for a specific transaction used by rollback, currently committed and recovery. 1 file per log file where log data is extracted for a transaction ID.
  - **X<logFileNum>.TMP** - meta data about extracted logs created during an in progress extraction for an active log file.
  - **X<logFileNum>.META** - meta data about extracted logs created after extraction completes for an active log file.

Logging – Transaction Log Full (11|14)

Advanced Log Space Management

Timeline:

- **11.5.0.0 – Technical Preview (not for production use)**
- **11.5.4.0 – Production support**
  - No mirror log support
  - No HADR support
  - No online backup support
  - No pureScale support
- **11.5.5.0 – Basic mirror log support**
- **11.5.6.0 – HADR and online backup support**

Logging – Transaction Log Full (12|14)

- **Enabled with registry variable DB2_ADVANCED_LOG_SPACE_MGMT=ON**
- 

Databases must be configured with archive logging
- **Ideal for workloads with long running, low logging volume transactions**
- **Extraction takes place by new EDU/thread – db2loggx running internal read log**
  - No to minimal impact to active workloads
  - Throttled
    - Only runs when needed and producing a benefit

Logging – Transaction Log Full (13|14)

- **Extraction will be throttled based on policies such as:**
  - Log archiving not healthy
    - Log data from the active log files that is not archived yet is not extracted
    - Ensure log archiving is healthy and/or a `FAILARCHPATH` is configured
  - Buffer pool flushing is slow
    - Active log data that is at or above what has been flushed from the buffer pools is not extracted
    - Ensure `PAGE_AGE_TRGT_MCR` and `PAGE_AGE_TRGT_GCR` (or `SOFTMAX` on older database configurations) are set to appropriate values based on your workload throughput
  - Log space consumption
    - Consumed active log space is below the threshold (80%)
    - Avoids extracting too aggressively and wasting resources
  - Disk available
    - Not enough disk space to hold active and extraction log files
  - High extraction ratio (a.k.a monster rule)
    - Database has a relatively high uncommitted workload to committed workload ratio (30%)
    - Extracted data may be equal or greater in size than the active log files
    - High extraction I/O rate could impact workload throughput

Logging – Transaction Log Full (14|14)

- **Throttled extraction results in idling**
  - No infinite logging
    - Transaction log full still possible in extreme cases
  - Infinite logging
    - Transaction log full will not happen
    - Improves on the negatives of infinite logging by avoiding un-necessary retrieves

Question 4:

How can I avoid disk full (`SQLO_DISK`) in my log archive target?

Logging – Disk Full in Archives (1|5)

- **Make sure on a separate file system from anything else esp. active log path**

- **Unhealthy archive log path means active log path will grow as log files cannot be archived**
  - This can lead to disk full in active log path and can cause workload failure or for infinite logging can cause database to come down
    - By setting `BLK_LOG_DSK_FUL` can avoid disk errors and make application wait

- **Avoid by setting `FAILARCHPATH` so that when archive log path becomes unhealthy Db2 can still move files out of active log path**
  - Temporary until archive log path healthy again
  - Also make sure on a separate file system

Logging – Disk Full in Archives (2|5)

- **Detect in advance**
  - Set up a monitoring script that reports physical disk space left on the file system where the log archive path(s) exist
- **React**
  - Cleanup whatever is not needed
  - May need to do manually at first ➔ BE CAREFUL not to delete too much
  - **Avoid the need for manual cleanup by letting Db2 manage**
    ➔ **Use Db2 Automatic Pruning**

Logging – Disk Full in Archives (3|5)

- **Db2 Automatic Pruning**
  - Choose how long to retain recovery objects like backup, load copy and archive logs and when to automatically prune them from history file **AND physically from media (local/vendor)**
    - To enable, set `AUTO_DEL_REC_OBJ` database configuration parameter
    - Set retention policy with database configuration parameters:
      - `NUM_DB_BACKUPS`: Number of full database backups to retain
      - `REC_HIS_RETENTN`: Number of days to retain historical data
    - When does this automatic deletion occur?
      - After a successful backup
      - On an explicit PRUNE HISTORY AND DELETE command
    - If using this does not work, then please raise with IBM Support

Logging – Disk Full in Archives (4|5)

- **If still does not help:**
  - Consider whether turning on log archive compression can help to decrease storage requirement
    - `LOGARCHCOMPR1/2` database configuration parameter ➔ [ON; NX842; ZLIB]
  - Consider using a vendor product as your log archive method type to defer storage management
  - Determine your recovery and/or replication retention requirements and allocate enough disk space to cover

Logging – Disk Full in Archives (5|5)

- **Additional helpful tips:**
  - If archiving to DISK, set `OVERFLOWLOGPATH` database configuration parameter to the archive location
    - If `LOGARCHMETH1/2` set to `DISK:/archivePath/`
    - Set `OVERFLOWLOGPATH` to `/archivePath/<instance name>/<database name>/`
    - Avoids the retrieval of archived logs
      - Saves the cost of un-necessary copying
      - Negated if using log archive compression
  - If on UNIX and archiving using TSM/vendor, set `LOGARCHOPT1/2` database configuration parameter option:

    `"--vendor_archive_timeout=<number of seconds>"`
    - If TSM/vendor unresponsive for the specified timeout then Db2 will interrupt and follow retry/failure protocol
      - Avoids possible database hangs

Agenda

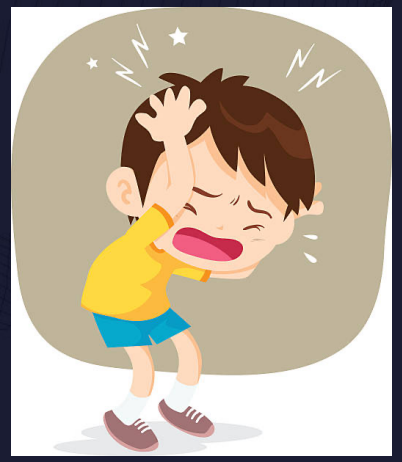
- Backup & Restore
  - Performance
  - Object (Remote) Storage
- Logging
  - Transaction Log Full
  - Disk Full in Archives
- **Recovery**
  - **Point-in-time Recovery**
  - **Inaccessible Table Spaces**
  - **Monitor Crash Recovery**

Question 5:

How do I rollforward to a point-in-time?

Recovery – Point-in-time Recovery (1|7)

- **Complications:**
  - Determining what point in time to use, esp. in a multi-partition environment (DPF)
    - Want to be precise to limit any data loss
    - Minimum recovery time (MRT)
  - Making all log files available
    - Archives or overflow?
    - Handling location in a multi-partition environment
  - If a multi-partition environment, coordinating command across partitions



Recovery – Point-in-time Recovery (2|7)

- **Before you begin:**
  - Recovery starts with backup
  - Recommended way to backup a database in a multi-partitioned environment (DPF) is to use single system view (SSV) backup
    - One command:

```
db2 backup database sample on all dbpartitionnums
      to /db2home/db2inst1/backup/ without prompting
```

      ➔ easier to backup
    - One timestamp for all images across all partitions
      ➔ easier to restore
    - Online backup will include logs by default across all partitions
      ➔ easier recovery to end of backup

Recovery – Point-in-time Recovery (3|7)

- **Before you begin (con't):**
  - Cannot replay logs from a different Db2 Version (VV.RR)
  - Cancel any previously no longer needed roll forwards
    - `db2 rollforward db sample CANCEL`
  - Only one ROLLFORWARD operation at a time can be issued
  - In a multi-partition environment (DPF):
    - Must be run on catalog partition
    - Point-in-time always involves all nodes
  - END OF BACKUP is just a special type of point-in-time
    - Db2 determines point in time across all partitions

Recovery – Point-in-time Recovery (4|7)

- **Before you begin (con't):**
  - Suggest you run command in two steps:
    - `db2 rollforward db sample to 2023-07-12-14.21.56`
    - `db2 rollforward db sample STOP`
  - Determine where log files will come from:
    - From archives?
    - From an overflow log path?
    - Both?
  - Restore logs from online backup image using `LOGTARGET` and supply to `ROLLFORWARD` command through `OVERFLOW LOG PATH`
    - Just supply base path, Db2 knows where to look:
      - Base path
      - Subdirectory: NODEnnnn/LOGSTREAMmmmm/
  - Use `NORETRIEVE` option if logs are completely locally supplied

Recovery – Point-in-time Recovery (5|7)

- **Before you begin (con't):**
  - How to choose a correct point-in-time?
    - Point-in-time can be local or in UTC/GMT
      - Timestamps in log files are in UTC/GMT
    - Need to achieve minimum recovery time (MRT)
      - All table spaces need to be in sync with catalogs
      - All partitions need to be in sync with catalog partition
    - Better to choose less if unsure
      - Db2 will return error and suggest newer time if need be
      - Each iteration of `ROLLFORWARD` can go forwards in time, but never backwards
    - **Use `db2fmtlog` to format logs and find meta data details about log records including table space ID, object ID and timestamps used for point-in-time:**

```
| LREC | 18654 0005633A 000000000199 Commit SE 2021-05-12-22.24.34 GMT
```

      - Not all log records have time stamps, mainly transaction ending (e.g. commit) log records

11.5.6+

Consider `RECOVER` command

Recovery – Point-in-time Recovery (6|7)

- **Once started:**
  - When a `ROLLFORWARD` command completes it displays status information for each database partition/member
    - `QUERY STATUS` can obtain as well
      - For database rollforward: the time stamp (in UTC) of the last committed transaction since rollforward processing began
      - Log files no longer needed and next log file to be processed

```
                                Rollforward Status

Input database alias                   = sample
Number of members have returned status = 3

Member ID    Rollforward                Next log           Log files processed        Last committed transaction
             status                      to be read
-----------  -------------------------  -----------------  -------------------------  --------------------------
           0  DB  working                S0001423.LOG        S0001422.LOG-S0001422.LOG  2021-10-27-07.32.56.000000 UTC
           1  DB  working                S0004727.LOG                     -             2021-10-25-03.05.53.000000 UTC
           2  DB  working                S0004584.LOG                     -             2021-10-25-03.04.32.000000 UTC

DB20000I  The ROLLFORWARD command completed successfully.
```

Recovery – Point-in-time Recovery (7|7)

- **Once started (con't):**
  - To monitor, use `LIST UTILITIES` or `db2pd -recovery`
    - Displays forward and backward phase estimates and total completed
  - To cancel an in progress rollforward that is running issue `CTRL-C` from terminal or `FORCE APPLICATION`
  - If giving up on a rollforward operation cancel it, using `CANCEL` option


```
ID                                   = 7
Type                                 = ROLLFORWARD RECOVERY
Database Name                        = SAMPLE
Member Number                        = 0
Description                          = Database Rollforward
Recovery
Start Time                           = 01/11/2023 12:56:53.770404
State                                = Executing
Invocation Type                      = User
Progress Monitoring:
   Estimated Percentage Complete     = 50
   Phase Number                      = 1
      Description                    = Forward
      Total Work                     = 528236 bytes
      Completed Work                 = 528236 bytes
      Start Time                     = 01/11/2023 12:56:53.770492

   Phase Number [Current]            = 2
      Description                    = Backward
      Total Work                     = 528236 bytes
      Completed Work                 = 0 bytes
      Start Time                     = 01/11/2023 12:56:56.886036
```

Question 6:

How can I avoid a table space becoming unavailable after a graceful takeover on my HADR standby?

Recovery – Inaccessible Table Spaces (1|6)

- **Follow recommendations as much as possible to keep both primary and standby systems the same:**
  - Same container paths
  - Same (or greater) physical disk space
  - Shared paths/access for load copy images

- **Not doing so can cause table space replay operations to fail on that table space and for the table space to become inaccessible**

Recovery – Inaccessible Table Spaces (2|6)

- **Certain operations can place a table space and/or a table into an inaccessible state on the standby:**
  - Not logged operations
  - Non-recoverable load / load `COPY NO`
  - Load `COPY YES` image not available for replay

Recovery – Inaccessible Table Spaces (3|6)

- **Avoid those operations by:**
  - Setting `BLOCKNONLOGGED` database configuration parameter to **YES**
    - `CREATE TABLE` and `ALTER TABLE` statements will fail if one of the following conditions is true:
      - The `NOT LOGGED INITIALLY` parameter is specified
      - The `NOT LOGGED` parameter is specified for a LOB column
      - A `CLOB`, `DBCLOB`, or `BLOB` column is defined as not logged
    - `LOAD` command fails if the following situations exist:
      - You specify the `NONRECOVERABLE` option
      - You specify the `COPY NO` option
  - Setting `DB2_LOAD_COPY_NO_OVERRIDE` registry variable to `"COPY YES to <path_name>"` to convert load `COPY NO`
    - Nothing available for non-recoverable loads
  - Use a shared file system for load `COPY YES`

Recovery – Inaccessible Table Spaces (4|6)

- **Monitor for inaccessible table spaces:**
  - db2pd -hadr or `MON_GET_HADR`
    - HADR_FLAGS: `STANDBY_TABLESPACE_ERROR`
  - If condition occurs:
    - Check db2diag.log on standby
    - db2pd -tablespaces
      - Common error states:
        - OFFLINE (0x4000) / RESTORE_PENDING (x100) / ROLLFORWARD_PENDING (x80)

- **Monitor for unavailable tables:**
  - If reads on standby:
    - `db2 "select TABSCHEMA, TABNAME, TABTYPE, AVAILABLE from TABLE(ADMIN_GET_TAB_INFO(null, null)) where AVAILABLE='N'`
  - Offline:
    - `db2dart /TS /TSI /QCK 15`

Recovery – Inaccessible Table Spaces (5|6)

- **Detect on first error:**
  - **`DB2_FAIL_RECOVERY_ON_TABLESPACE_ERROR`** registry variable
    - This variable specifies whether a recovery operation (including database/table space rollforward, database/table space restore and HADR standby replay) should fail after encountering an error condition on a table space.
    - **NO**: (Default) Current behavior of not failing operation on table space error and table space state goes recovery pending
    - **YES**: Database/table space rollforward, database/table space restore and HADR standby replay fails if any table space error is seen and table space state untouched
      - In 11.5.7.0: YES, now includes database/table space restore when there is an issue with a table space container the restore will fail

Recovery – Inaccessible Table Spaces (6|6)

- **If detected:**
  - HADR standby monitoring should tell that standby has gone down
  - Look at db2diag.log for details on failure
  - Resolve problem and re-activate standby:
    - Create or mount file system
    - Create more physical disk space
  - Last resort table space restore on standby:
    - **Standby**: DEACTIVATE
    - **Primary**: FLUSH BUFFERPOOL ALL / BACKUP table space
    - **Standby**: RESTORE table space / ACTIVATE
- **Avoid table space becoming inaccessible before takeover command is ever issued**

Question 7:

How can I monitor crash recovery or make data available sooner?

Recovery – Monitor Crash Recovery (1|5)

- **Crash recovery brings the database to a consistent state following a “crash” (abnormal termination) of the database**

- **Crash recovery is performed:**
  - Explicitly by the user using `RESTART DATABASE` command
  - Implicitly by the database during first connect if `AUTORESTART= ON`

- **Consists of two phases:**
  - Forward (redo) phase
    - Transaction logs are used to redo work that may not be persisted to disk yet
    - Multiple agents attempt to replay transactions in parallel
  - Backward (undo) phase
    - Uncommitted (in-flight) work is rolled back
    - Performed using a serial process

Recovery – Monitor Crash Recovery (2|5)

- **Crash recovery is considered an offline operation**
  - Connections are blocked until recovery completes
- **In multi-partition environments (DPF), crash recovery done per partition**
  - Catalog partition always done first
- **In pureScale environments, two types of crash recovery exist:**
  - Member crash recovery: Recover just one member
  - Group crash recovery: Recover all members
- **To monitor, use `LIST UTILITIES` or `db2pd -recovery`**
  - Displays forward and backward phase estimates and total completed

Recovery – Monitor Crash Recovery (3|5)

- **Things that can affect crash recovery time:**
  - **Recovery in general starts at the minimum of:**
    - **lowtran -** First (lowest) log record belonging to oldest open transaction
    - **minbuff -** Log record of the oldest (minimum) dirty page in buffer pool
  - **lowtran less than minbuff**
    - Will read log records, but only to rebuild transaction table for backward phase
    - Very little to replay below minbuff
    - Long running transactions
    - Shorten length, commit frequently
    - `MAX_LOG` / `NUM_LOG_SPAN`

Recovery – Monitor Crash Recovery (4|5)

- **Things that can affect crash recovery time (con't):**
  - **minbuff less than lowtran**
    - Buffer pool flushing
      - Pages containing the committed transactions are not written back to disk very frequently (Note that the use of asynchronous page cleaners (`NUM_IOCLEANERS`) can help avoid this situation)
      - Larger bufferpool sizes take longer to flush
      - **`PAGE_AGE_TRGT_MCR`**: time-based flushing of local bufferpool
      - **`PAGE_AGE_TRGT_GCR`** (pureScale): time-based flushing of group bufferpool
      - **`SOFTMAX`** (deprecated): log data volume flushing
  - **Large log files can mean less flushes of in-memory lowtran/minbuff to disk**
    - Choose a size that does not affect runtime workloads and meets log archiving requirements

Recovery – Monitor Crash Recovery (5|5)

- **Things that can affect crash recovery time (con't):**
  - **Backward phase is a serial process**
    - Typically, only a subset of the data belongs to uncommitted workload that needs compensating
    - If this is not "hot" data, making the database available sooner has potential merit
      - Consider configuring database to allow for accessibility during the backward phase
        ➔ Set registry variable `DB2_ONLINERECOVERY=YES`
        - Also applies to HADR takeover
        - No instance restart is necessary
        - Uncommitted transactions for workload like: DDL, catalogs, column organized tables still need to be done synchronously
        - For transactions being rolled back asynchronously they will hold locks until complete

Resources

- **IBM Documentation**
  - https://www.ibm.com/docs/en/db2/11.5
- **HADR Wiki (incl. Best Practices)**
  - https://ibm.github.io/db2-hadr-wiki/

Questions ???

IDUG

IDUG

2024 NA Db2 Tech Conference

Thank You

Speaker: Michael Roecken, Roger Zheng

IBM

@roecken

rzheng@ca.ibm.com

Session Code: RAS1

Please fill out your session evaluation!

@IDUGDb2

#IDUG_NA24