

# Demystifying Recovery

## - Top Questions From Customers Answered

Wei Cao, IBM

Roger Zheng, IBM

Db2 LUW

Tue, Sept 19, 2023 (02:00 PM -02:50 PM Session LUW-10)



# Safe Harbor Statement

- *Copyright © IBM Corporation 2023. All rights reserved.*
- *U.S. Government Users Restricted Rights - Use, duplication, or disclosure restricted by GSA ADP Schedule Contract with IBM Corporation*
- ***THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON CURRENT THINKING REGARDING TRENDS AND DIRECTIONS, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. FUNCTION DESCRIBED HEREIN MAY NEVER BE DELIVERED BY IBM. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.***
- *IBM, the IBM logo, ibm.com and Db2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)*

# Agenda

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



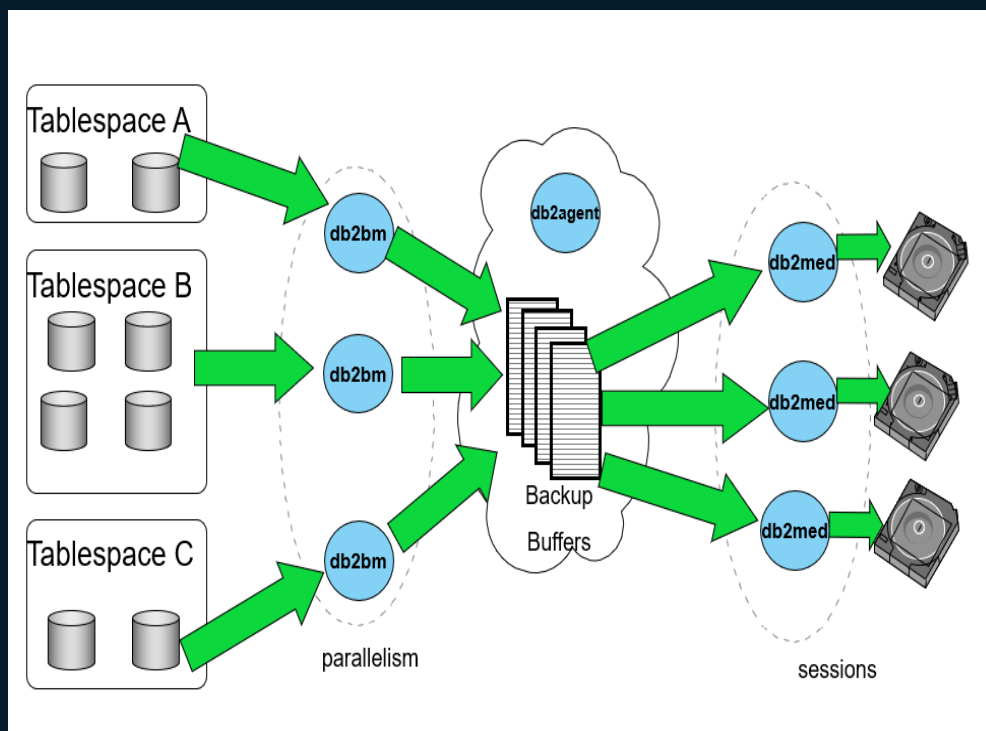


**Question 1:**



**How can I identify  
backup performance  
problems?**

# Backup & Restore – Performance (1|14)



**Backup Process Model**



## **db2agent – agent the drives backup**

Spawn db2bm and  
db2med

Only 1 db2agent per  
backup



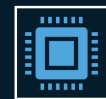
## **db2bm – buffer manipulator**

Reads data from table  
spaces and transfer  
data to backup buffer

Handle compression  
and encryption

1 db2bm per table  
space

Number of db2bm  
equals PARALLELISM



## **db2med - media controllers**

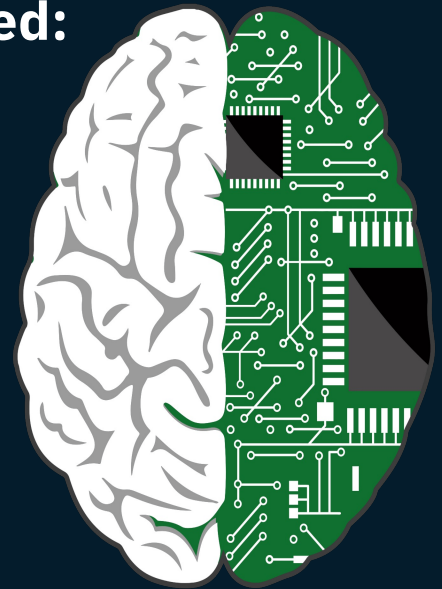
Transfer data from  
backup buffer to target

Number of db2med  
equals number of  
targets or sessions

# Backup & Restore – Performance (2|14)

## 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|14)

## 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
  - Use the `db2pd` command with the `-barstats` to monitor backup progress



# Backup & Restore – Performance (4|14)

## Some “simple” things you can do to help:

- Increase PARALLELISM on BACKUP DB command
  - Determines # of db2bm, that is one per table space
- Distribute your data evenly across table spaces \*\*
  - Unbalanced table spaces may cause longer backup time
  - **vNext will improve it with intra-tablespace parallelism**
- Allocating more memory by increasing the UTIL\_HEAP\_SZ
  - Affects both the number of buffers and buffer size
- Increase the number of buffers
  - Number of buffers  $\geq$  Number of db2bm + Number of db2med + 1
  - Enough buffers will ensure db2bm and db2med do not have to wait for buffer



# Backup & Restore – Performance (5|14)

## Some “simple” things you can do to help (cont’d):

- Increase the backup buffer size
  - Ideal size: common multiple of the extent sizes plus one page
- Specify table space backup
  - Facilitates the management of table data/indexes/LF/LOB data in separate table spaces.
- Use 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 TABLE

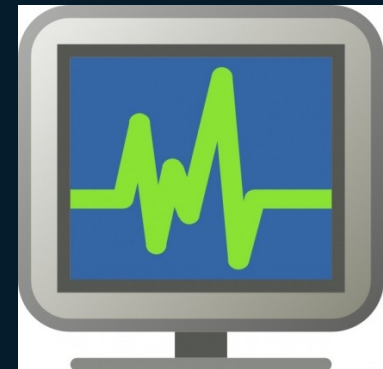
# Backup & Restore – Performance (6|14)

## Some “more advanced” things you can do to help:

- Offload compression and encryption to hardware
  - NX842-accelerated compression of POWER7+ on AIX
  - ZLIB compression on AIX and Linux on IBM z systems
  - Review `reg var DB2_BCKP_COMPRESSION` and `db2 cfg parm encr lib`
- Avoid flushing buffer pools by setting registry variable `DB2_REDUCE_FLUSHING_DURING_BACKUP`
  - Big buffer pools can cause higher flush times for online backup
  - May need more time at rollforward time
- Reduce time retrieving log files into backup image
  - Minimize log 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 (7|14)

- 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 (8|14)

- **Final stats** is displayed in **Db2 diagnostic log**

- **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 (9|14)



## BM: Buffer Manipulator

- READ data from the table spaces and place them into buffers.
- **BM#** - the number we assigned to an individual Buffer Manipulator.

## MC: Media Controller

- WRITE buffers out to the target location.
- **MC#** - the number assigned to an individual Media Controller.

```
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 (10|14)



**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 BM: 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.

```
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 (11|14)



**MsgQ** – time spent waiting to get a buffer

- For BM: time spent waiting to get an empty buffer for filling.
- For MC: time spent waiting to get a full buffer to write out.

**WaitQ** - time spent waiting on directives from the db2agent overseeing the whole backup.

```
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 (12|14)



**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.

```
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 (13|14)

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%

- On average spent 2.71% of time waiting to read data from the database
- Spent 89.63% of the time waiting for buffers

BMs:

- On average spent 99.92% of the time waiting for I/O (in this case vendor)

MCs:

- 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

Conclusion:

# Backup & Restore – Performance (14 | 14)

## Solution Considerations:

- **Increase the number of sessions**
  - Share the load of writing to target
  - Doing so may alter other resource parameters (like more BMs, number of buffers, etc.)
- **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



**Question 2:**



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

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

- **Recovery operations enabled for remote storage:**
  - Backup & Restore – since 11.1.0.0
  - Log Archive & Retrieve – since 11.5.7.0
- **Supported Cloud Object Storage providers using S3 protocol:**
  - IBM Cloud Object Storage
  - Amazon Simple Storage Service (S3)
- **Supported platforms – Db2 server running on x86\_64 platform only:**
  - SuSe
  - RHEL Linux
  - Ubuntu Linux

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

## Local staging path

- Hold temporary files for remote storage communication with cloud object storage
- Only required in certain situations depending on operations
- 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**

- **Should not set as registry variable is deprecated**
- Communication uses old legacy `libcurl` method
  - Requires local staging path to hold temporary files
  - Has limitations on file sizes; max file size 5GB
  - Lack of support for streaming multipart uploads
- **BACKUP:** each session has 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 remote storage multipart upload part size (`MULTIPARTSIZEMB`) database configuration parameter
- Allows streaming multipart uploads, which can remove need for local staging path for some operations
- **BACKUP:** local staging path is not required
  - Each session maximum image size: `MULTIPARTSIZEMB * maximum number of parts`
  - 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)

## Remote storage alias

- Db2 commands enabled for remote storage needs a remote storage alias to interact with the object storage provider
- **CATALOG STORAGE ACCESS** command creates a remote storage alias with account credentials stored in an encrypted keystore
- Managing remote storage alias
  - List storage access aliases
    - `list storage access`
  - Remove a storage access alias
    - `uncatalog storage access alias <alias>`
  - Rotate the master key in the keystore
    - `rotate master key for storage access`



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

## Example:

- **Create a local keystore:**

```
gsk8capicmd_64 -keydb -create  
-db "/home/db2inst1/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/db2inst1/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)

- **Syntax for specifying a remote storage location:**

```
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
```

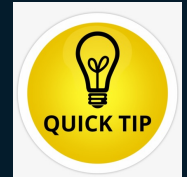
```
RESTORE DB sample FROM DB2REMOTE://myAlias/myContainer/backups
```

- **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**
  - 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



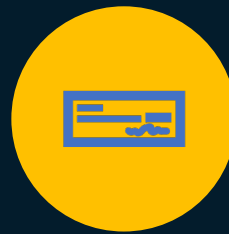
# Agenda

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





**Question 3:**



**How can I avoid  
transaction log full  
(SQL0964C)?**

# Logging – Transaction Log Full (1 | 13)


- **Maximum active log space**
  - $(\text{LOGPRIMARY} + \text{LOGSECOND}) * \text{LOGFILSIZ}$
- **Fixed pre-allocated active log space**
  - $\text{LOGPRIMARY} * \text{LOGFILSIZ}$
  - 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|13)

- **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**

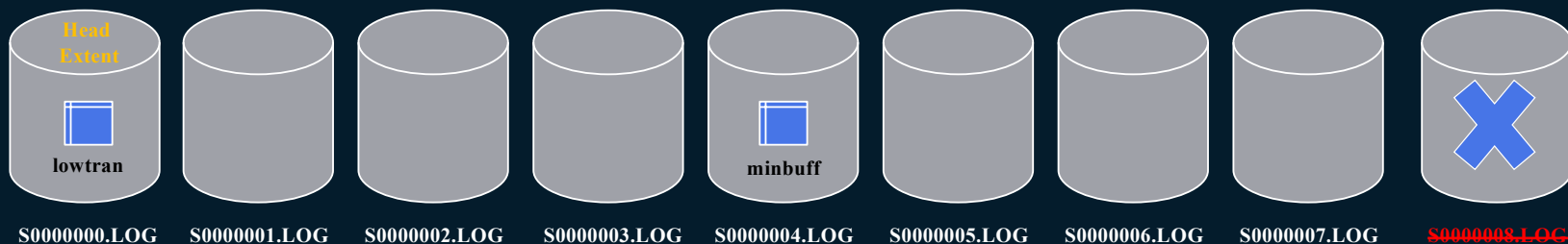
- Log record of the oldest (minimum) dirty page in buffer pool



# Logging – Transaction Log Full (3 | 13)

- 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

Active Log Space – LOGPRIMARY 8, LOGSECOND 0



# Logging – Transaction Log Full (4|13)

- **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 | 13)

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

11.5.4+

**Advanced Log Space Management**

# Logging – Transaction Log Full (6 | 13)

## If **minbuff**:

- **Short Term**

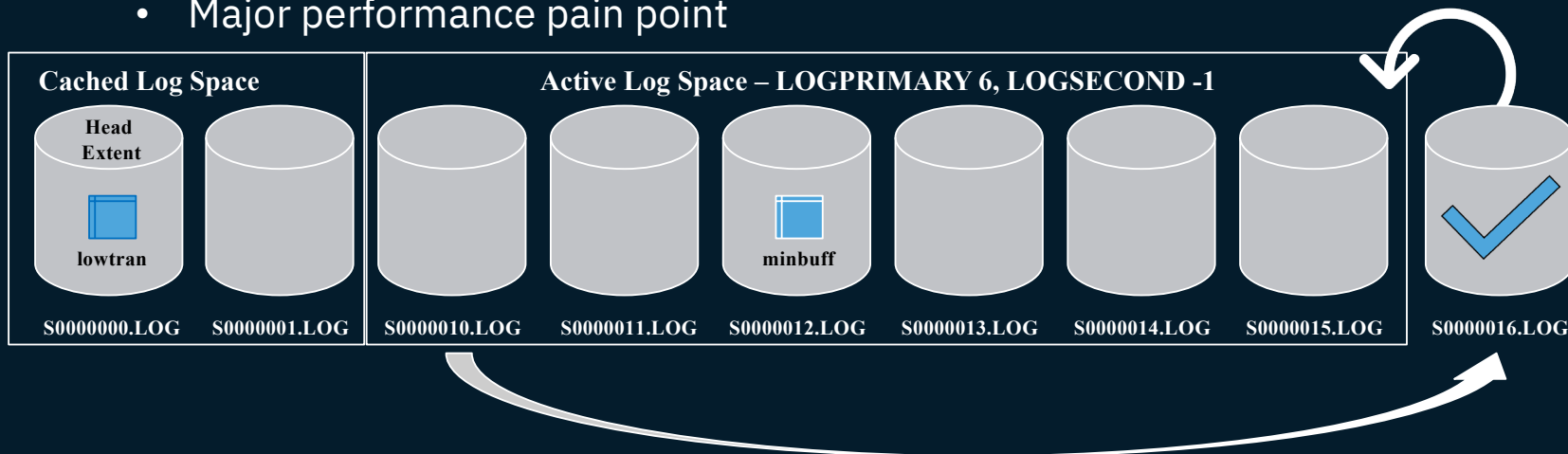
- Attempt to issue `FLUSH BUFFERPOOL ALL` command
- Increase `LOGSECOND` dynamically
  - Maybe even using infinite logging (`LOGSECOND = -1`)

- **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

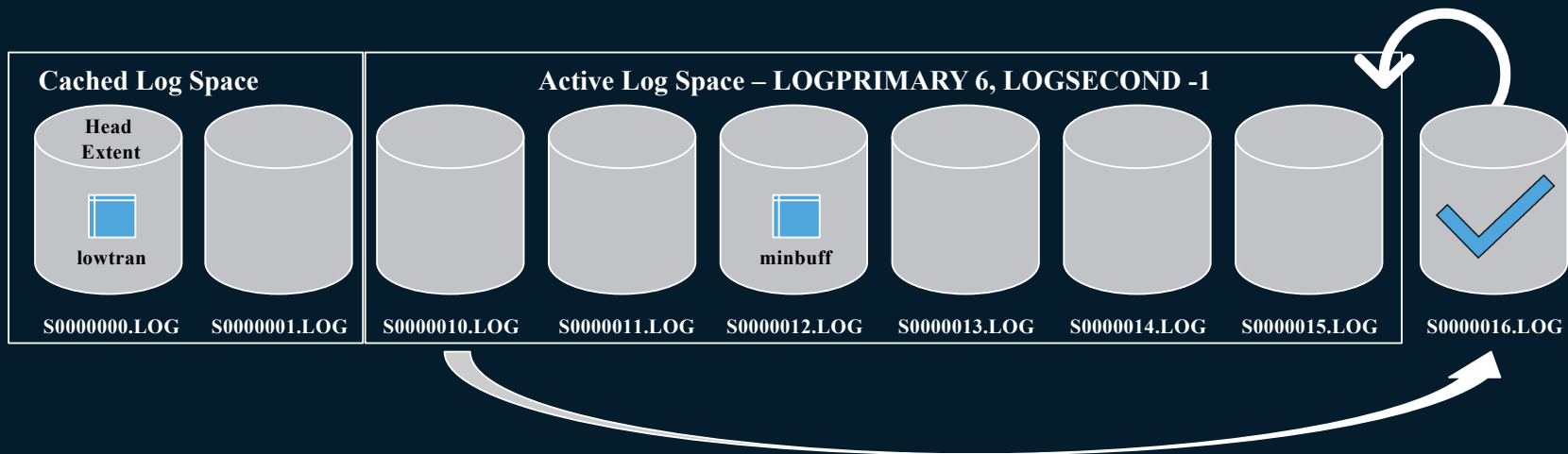
# Logging – Transaction Log Full (7 | 13)

- 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



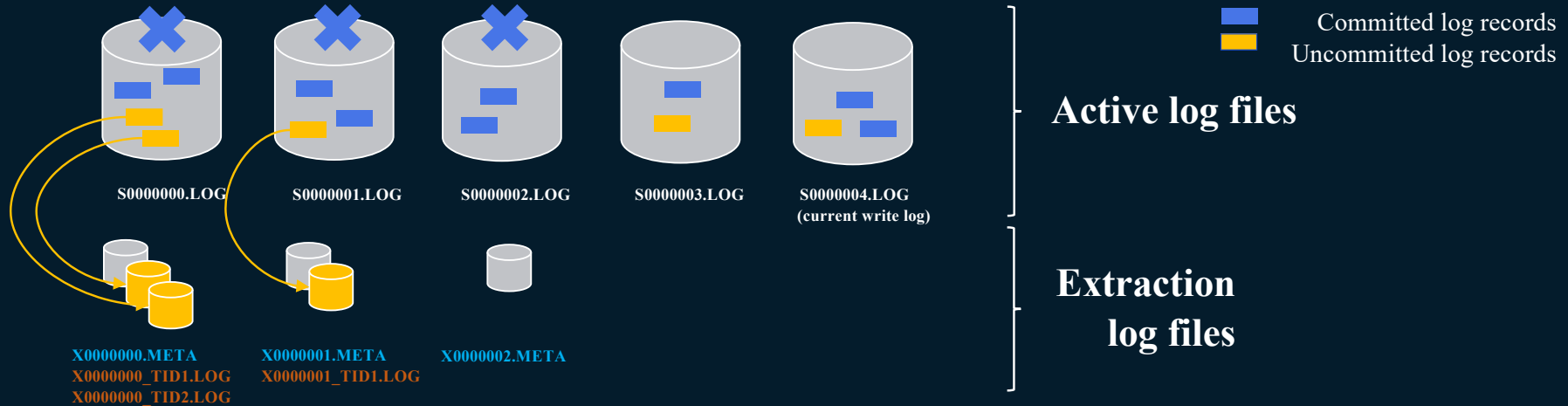
# Logging – Transaction Log Full (8|13)

- 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



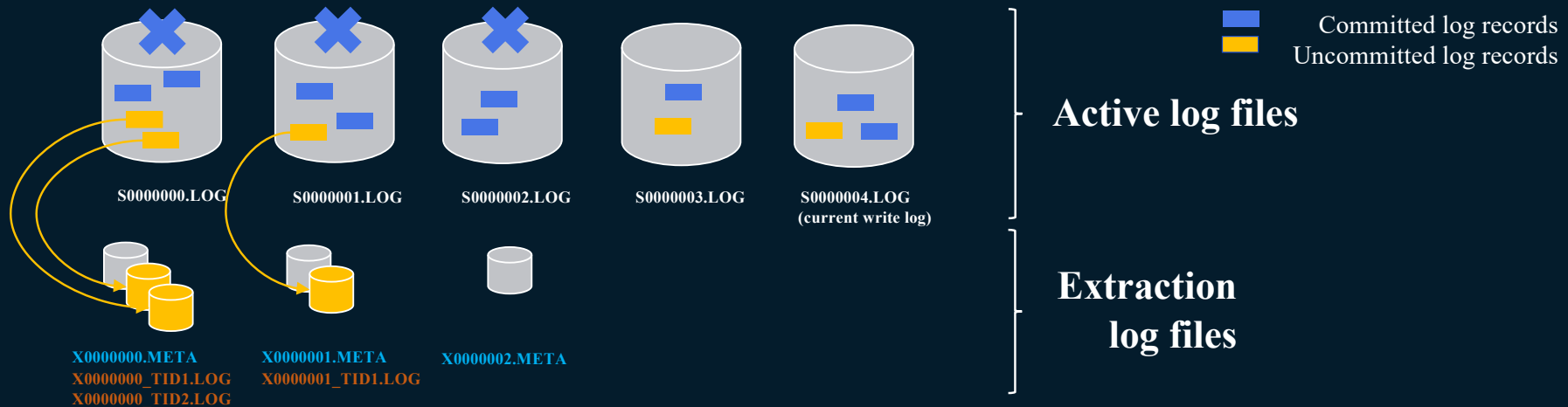
# Logging – Transaction Log Full (9|13)

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|13)




- **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 | 13)

- 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 (12|13)

## Extraction throttling policies:

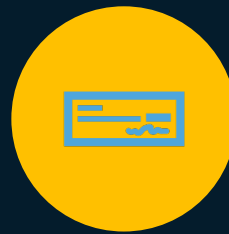
- **Log archiving not healthy**
  - Log data from the active log files that is not archived yet is not extracted
  - Ensure log archiving is healthy and/or a FAILARCHPATH is configured
- **Buffer pool flushing is slow**
  - Active log data that is at or above what has been flushed from the buffer pools 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 (13 | 13)

- **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
    - 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 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

# Logging – Disk Full in Archives (4 | 5)

- **If still does not help:**

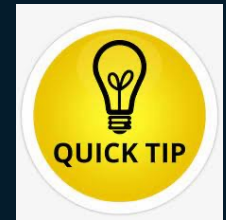
- Consider whether turning on log archive compression can help to decrease storage requirement
  - `LOGARCHCOMPR1/2` database configuration parameter → [ON; NX842; ZLIB]
- Consider using a vendor product as your log archive method type to defer storage management
- Determine your recovery and/or replication 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>/<db 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



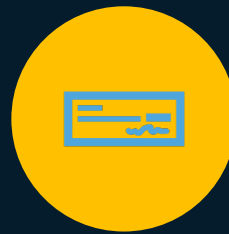
# Agenda

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





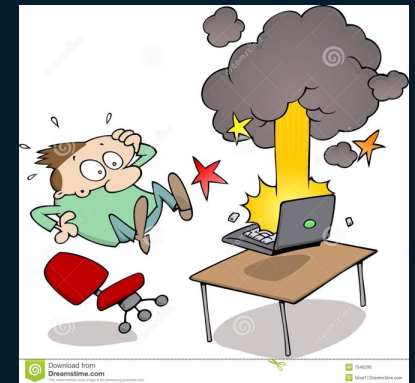
**Question 5:**



**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?

# Thank you

- **Speakers:**

- Wei Cao, IBM Toronto Lab
  - [linkedin.com/in/wei-cao-56790876](https://www.linkedin.com/in/wei-cao-56790876)
  - weicao@ca.ibm.com
- Roger Zheng, IBM Toronto Lab
  - [linkedin.com/in/roger-zheng-72a47b4](https://www.linkedin.com/in/roger-zheng-72a47b4)
  - rzheng@ca.ibm.com

- **Slides courtesy of Michael Roecken**

- @roecken

# Backup slides

# What's New in Version 11.5.8.0 (1|3)

## Log replay performance for crash recovery, database rollforward and HADR standby

- New registry variable `DB2_RECOVERY_SETTINGS`
  - `NUM_AGENTS`
  - `NUM_QSETS`
  - `QSETSIZE`
- New registry variable `DB2_DPS_LFR_PAGES_PER_IO`
  - Specifies the buffer size used by the db2lfr EDU to read log pages from disk

# What's New in Version 11.5.8.0 (2|3)

## Upload load copy images from local disk to TSM using db2adult

- The **db2adutl UPLOAD option** is enhanced to upload load copy images from a local disk to TSM.
- The newly supported **LOADCOPY** keyword can be used to upload load copy images based on the following criteria:
  - A file list
  - a time stamp
  - The most recent image in a history file
- History file entry is updated to facilitate future recovery of such images.

# What's New in Version 11.5.8.0 (3|3)

## New fields added to the db2ReadLog API and related structures

- Support mapping of a start time to an LRI.
- The following field changes have been made to the API:
  - **piStartTime** is added to the db2ReadLogStruct structure
  - **finalLRI** is added to the db2ReadLogInfoStruct structure
  - **nextStartLRI** is enhanced in the db2ReadLogInfoStruct structure