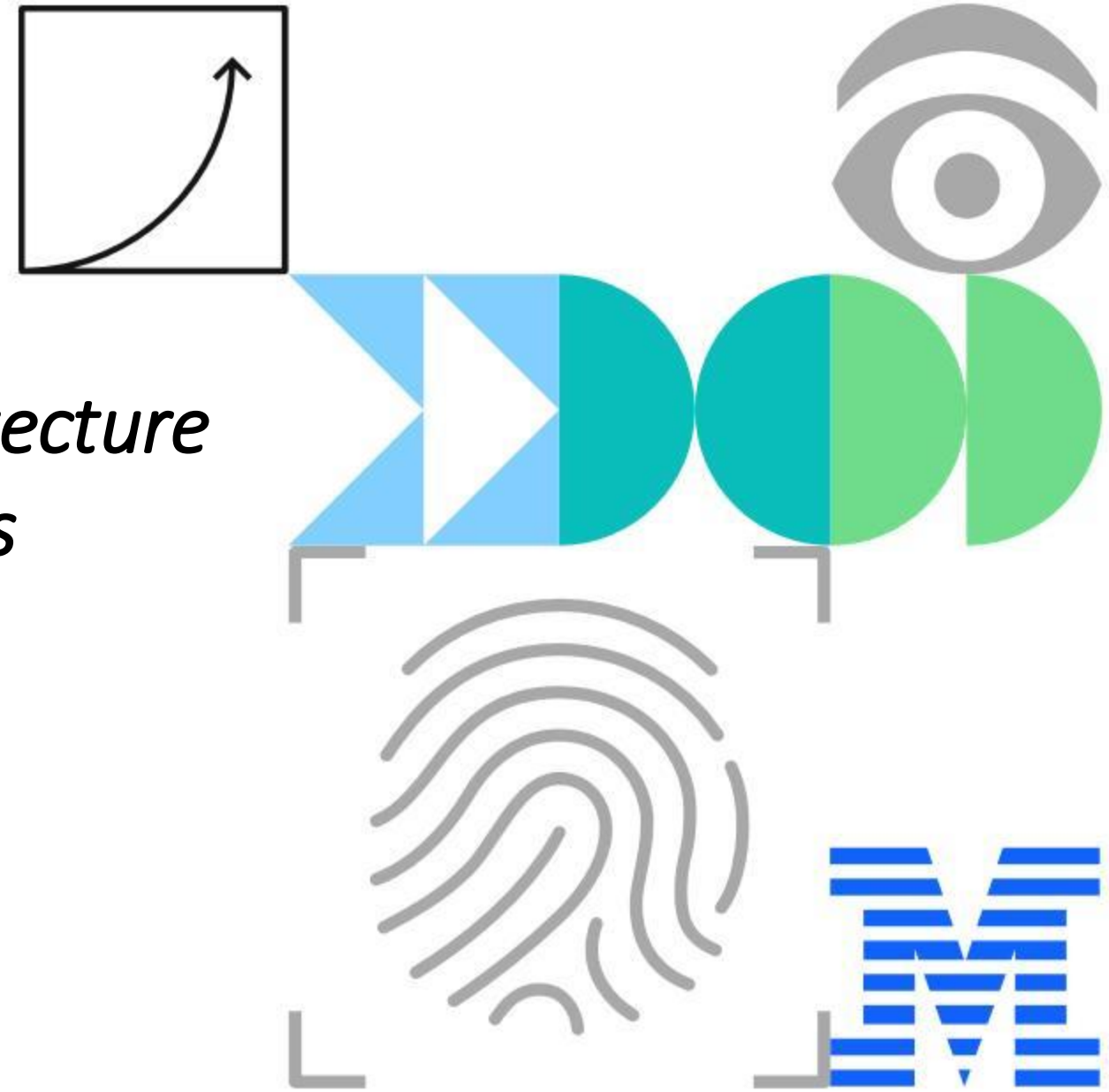
IBM **TechXchange**

IBM Db2 LUW Webinar Series



Db2 Cloud Object Storage Architecture and Performance Considerations

May 7 2024

Robert Hooper
Kostas Rakopoulos

IBM

Notices and disclaimers

This document is distributed "as is" without any warranty, either express or implied. In no event shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.

Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM.

Not all offerings are available in every country in which IBM operates.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

Certain comments made in this presentation may be characterized as forward looking under the Private Securities Litigation Reform Act of 1995.

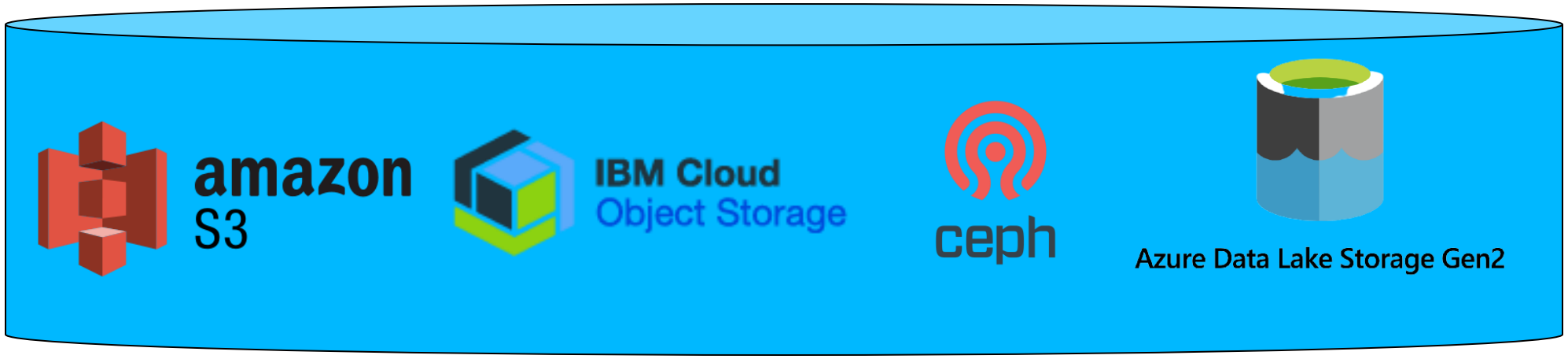
Forward-looking statements are based on the company's current assumptions regarding future business and financial performance. Those statements by their nature address matters that are uncertain to different degrees and involve a number of factors that could cause actual results to differ materially. Additional information concerning these factors is contained in the Company's filings with the SEC.

Copies are available from the SEC, from the IBM website, or from IBM Investor Relations.

Any forward-looking statement made during this presentation speaks only as of the date on which it is made. The company assumes no obligation to update or revise any forward-looking statements except as required by law; these charts and the associated remarks and comments are integrally related and are intended to be presented and understood together.

Object Storage

- ✓ **Near unlimited scalability**
- ✓ **Extreme durability + reliability**
- ✓ **High throughput**
- 🛑 **High latency (but can be compensated for) + storage model (immutable)**



Cloud Object Storage (COS)

Evolution of the Storage Architecture

High-Performance
Cloud Block Storage

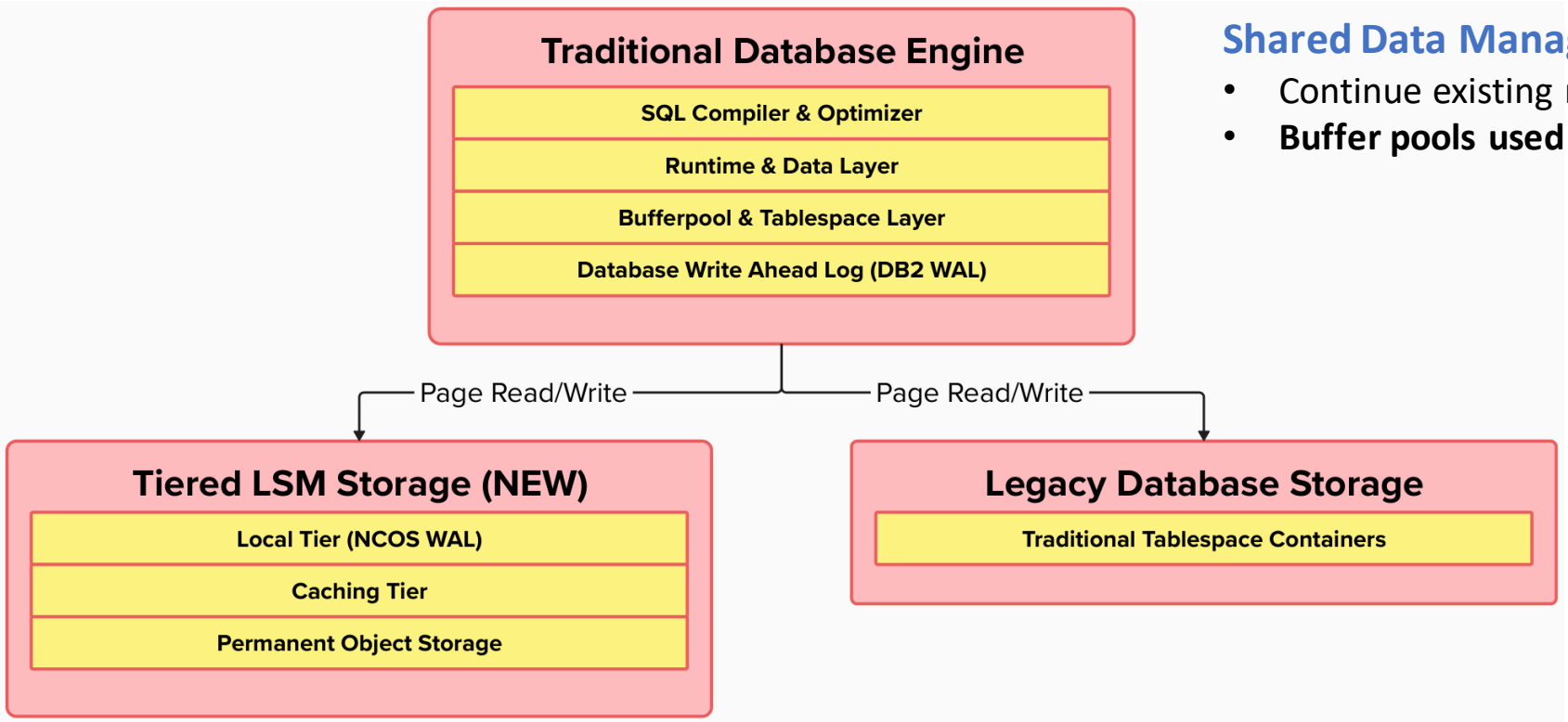
Locally Attached NVMe Drives

Decoupled Persistent
Cloud Object Storage

@ 10-30ms latency each (6 IOPS/GB)

@ 100-300ms latency per operation

Architecture Overview



Shared Data Management

- Continue existing model
- **Buffer pools used for both**

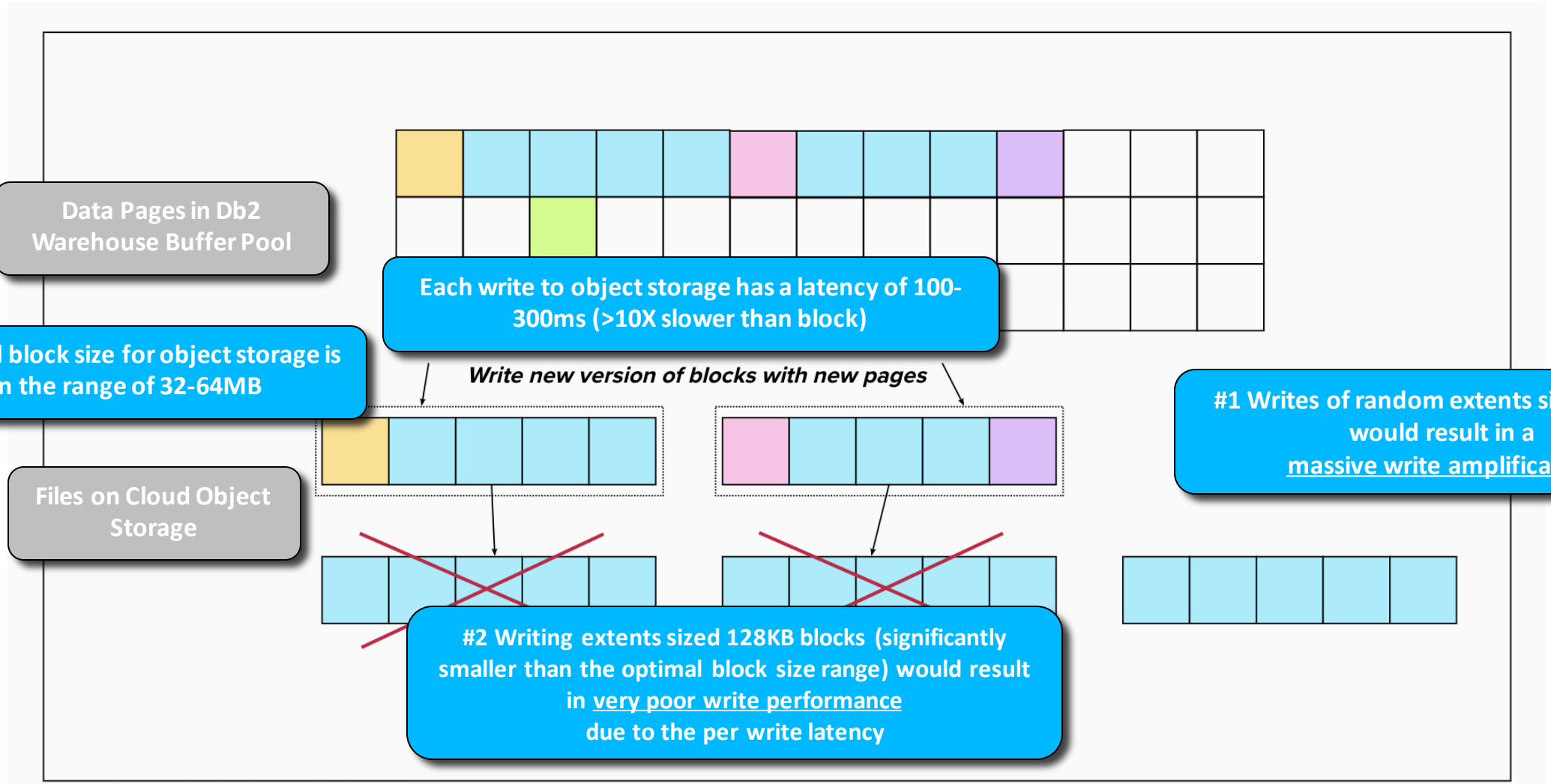
NCOS TABLESPACES

- Write to object storage
- Dedicated NVMe cache
- Columnar only
- **Latency mitigation**

Non-NCOS TABLESPACES

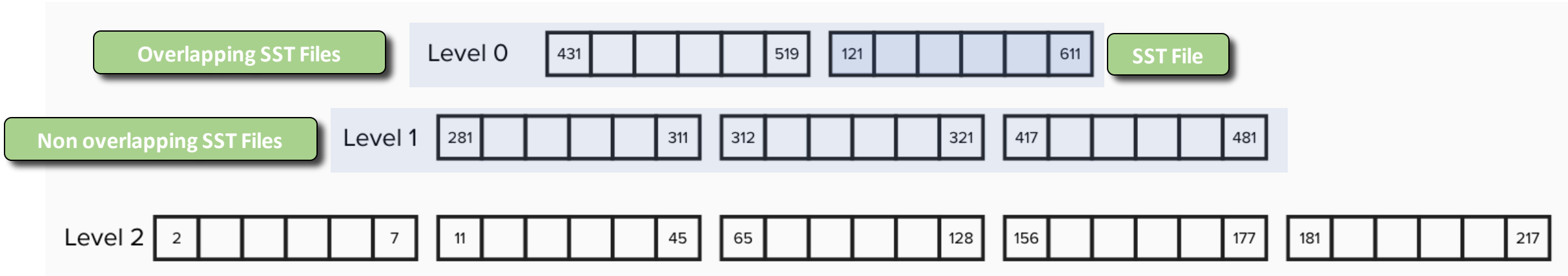
- Continue existing storage model
- **Db2 WAL used for both**

Pitfalls of a naïve storage model



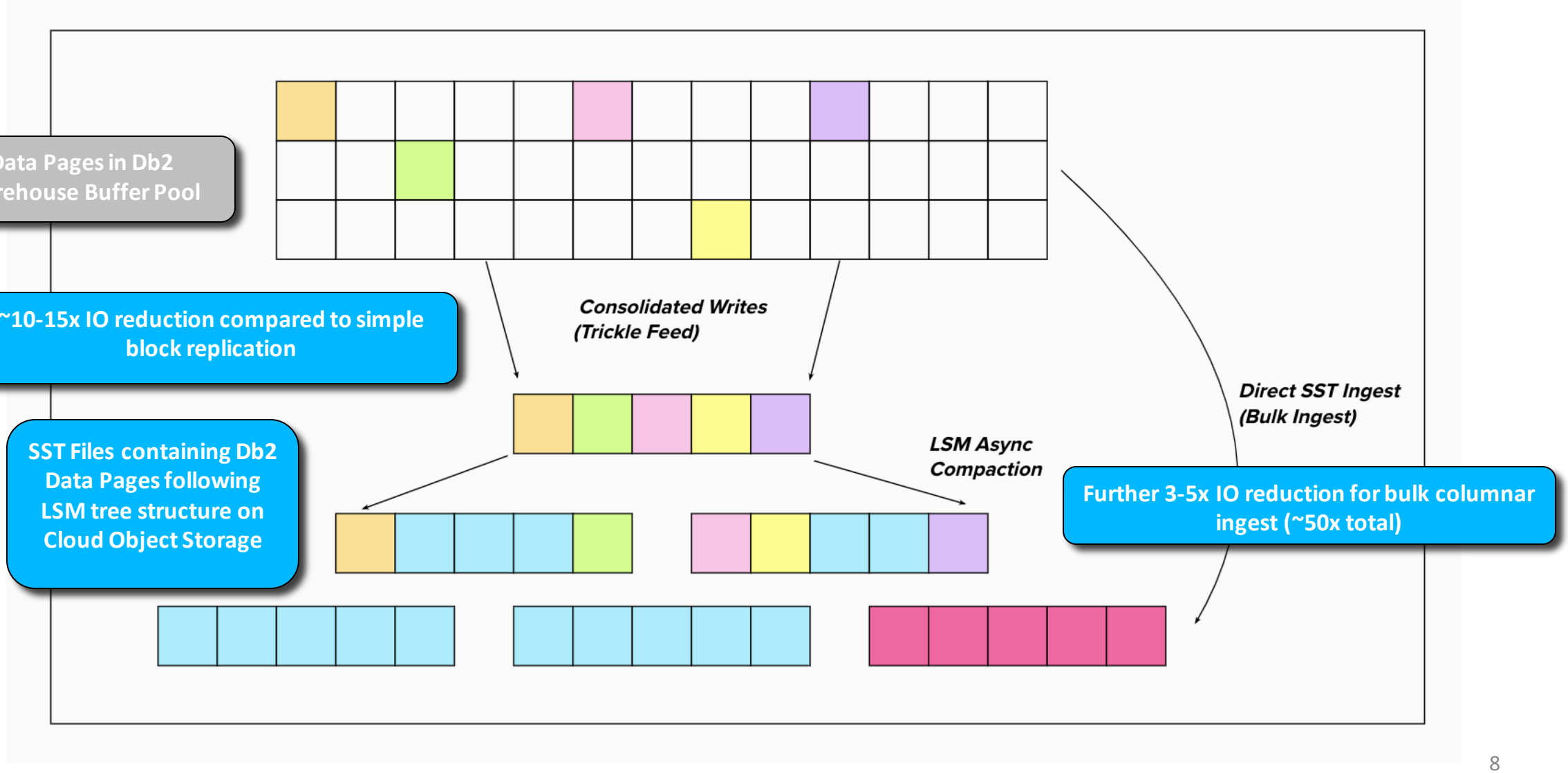
Background On LSM trees

- RocksDB is used under the hood. Log Structured Merge trees (LSM tree) is an index structure designed for on disk low-cost indexing for data with a high insert rate.

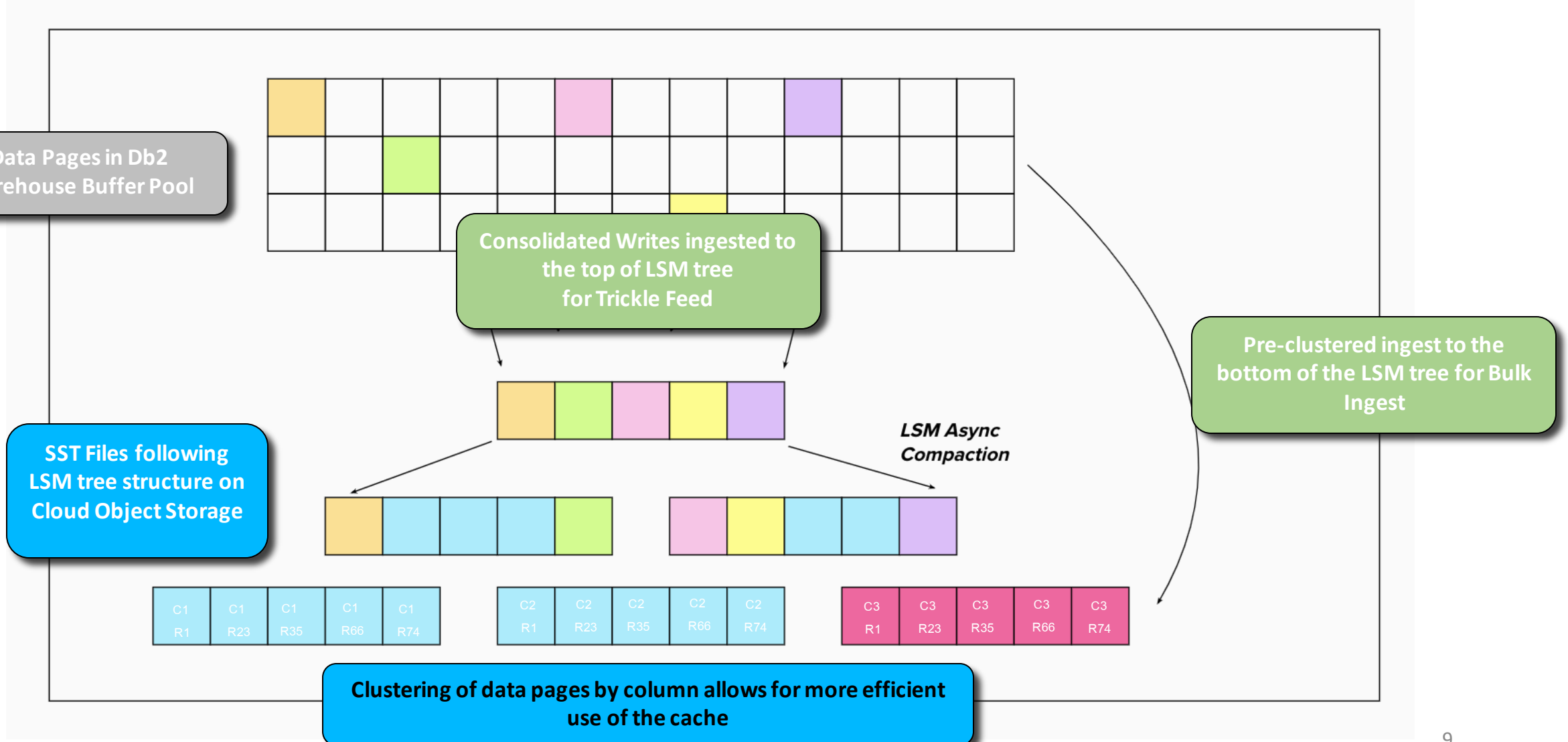


- There are three main characteristics that make it really interesting as a storage model for Db2 Warehouse:
  1. It follows an append-only write mode, where its SST files are only written once, which is ideal for cloud object storage and to simplify cache management.
  2. It is designed for self-optimization, through its background compaction process that moves data through the fully ordered levels.
  3. It is built for a high-volume ingest rate, ideal for data warehouses.

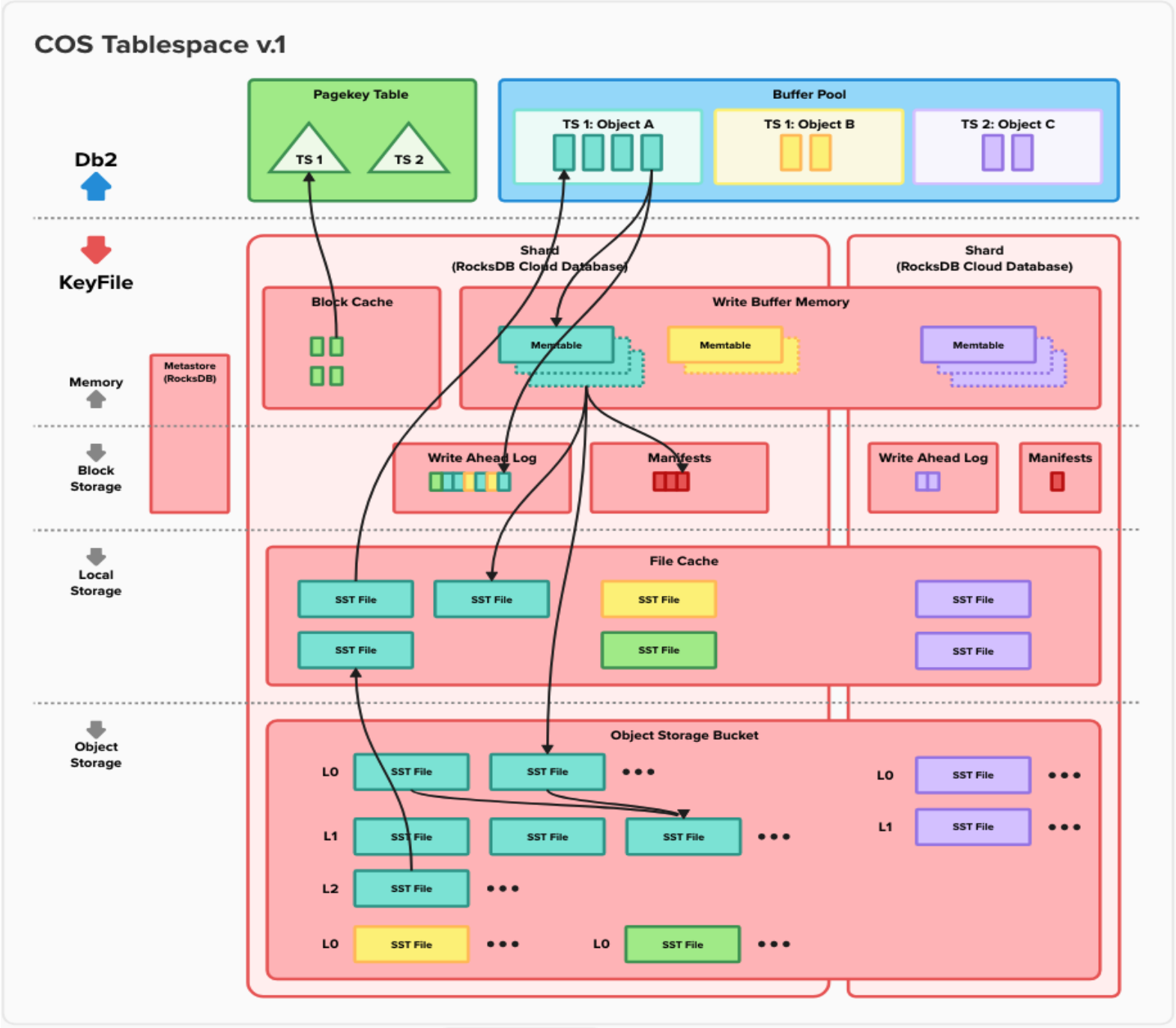
LSM Tree based page IO



Column Group Clustering within LSM tree



Looking Deeper Under the Hood



Using Native COS Tables

These are DBA tasks. Developers can use NCOS tables with no changes to their applications.

Step 1: Catalog Storage Access Alias

```
CATALOG STORAGE ACCESS ALIAS stoaccess1
VENDOR S3
SERVER https://s3host.com
USER db2inst1
PASSWORD db24ever
CONTAINER db2bucket
OBJECT default
DBUSER db2inst1
```

Step 2: Create Storage Group

```
CREATE STOGROUP stogroup1
ON 'DB2REMOTE://stoaccess1'
```

Step 3: Create Tablespace

```
CREATE TABLESPACE tbsp1
USING STOGROUP stogroup1
```

Step 4: Create Table

```
CREATE TABLE t1 (c1 INTEGER)
IN tbsp1
ORGANIZE BY COLUMN
```

Demo: Setup Steps

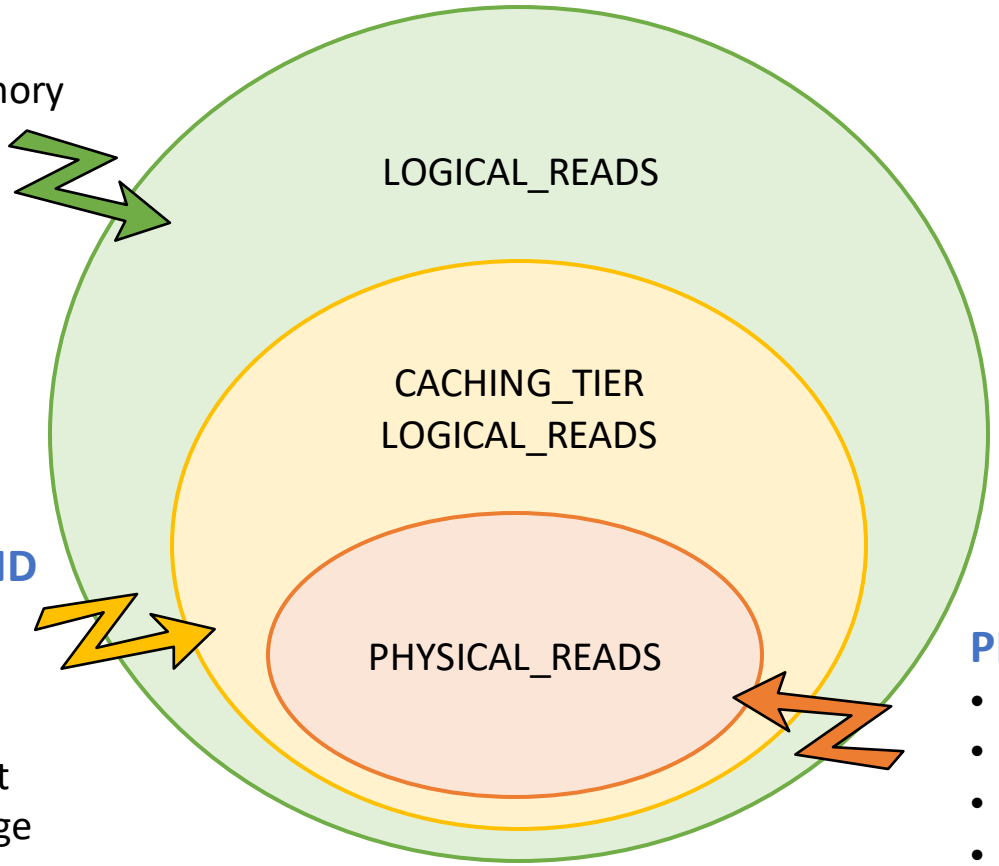
Native COS Monitoring – Writes

- New types of writes
  - LOCAL_TIER_WRITE
    - Persistent storage (e.g., network attached block storage) write
  - CACHING_TIER_WRITE
    - Very fast, ephemeral storage (e.g., local NVMe) write
  - REMOTE_STORAGE_TIER_WRITE
    - Object storage write (excludes writes due to compaction)
  - COMPACTION_WRITE
    - Object storage write due to compaction
- For each write type we record time, bytes and number of requests

Demo: Monitoring Writes

Native COS Monitoring – Reads

LBP_PAGES_FOUND
- Bufferpool hit
- Read from memory
- Very fast



Metrics are collected at various levels (e.g., database, tablespace, statement). This allows for easy drill down to find bottlenecks.

CACHING_TIER_PAGES_FOUND
- Buffer Pool miss
- Caching Tier hit
- Read from Local Disk (NVMe)
- Much slower than memory but much faster than Object Storage

PHYSICAL_READS
- Buffer Pool miss
- Caching Tier miss
- Read from Object Storage
- Very slow

Demo: Monitoring Reads

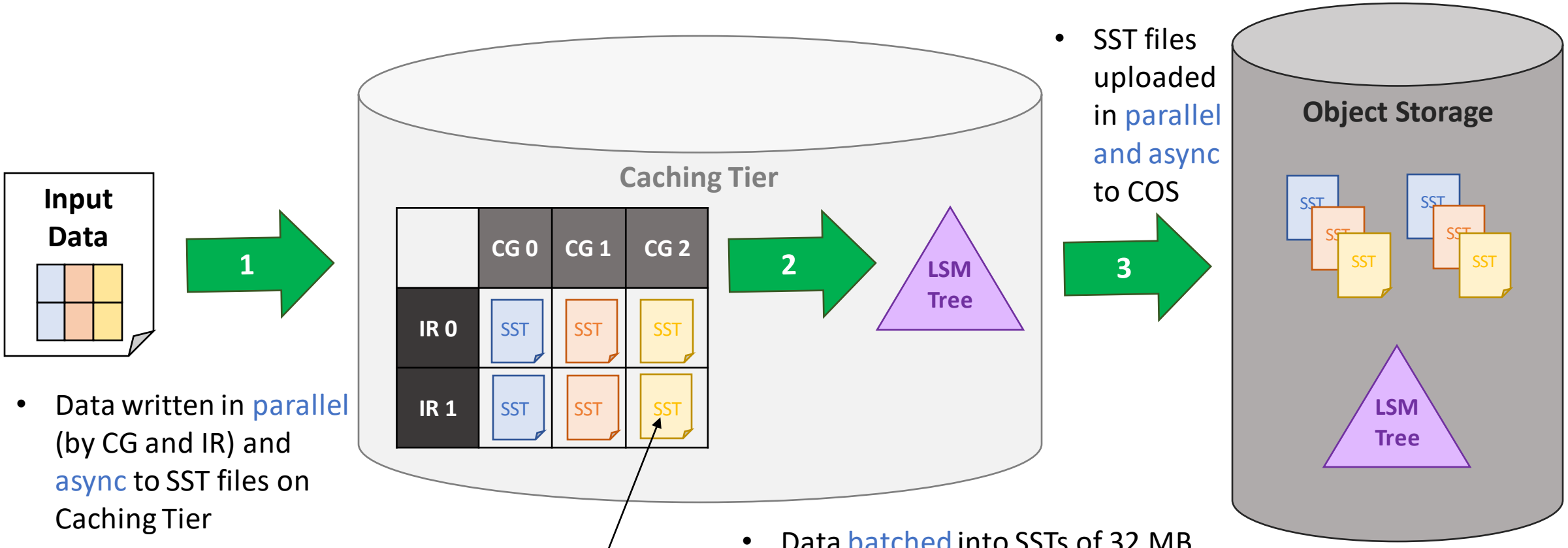
Native COS Performance – Key Warehouse Use Cases

- Bulk Insert
  - Millions of rows per transaction
  - Inserts directly into LSM tree
- Trickle Feed Insert
  - Tens to Thousands of rows per transaction
  - Utilizes Write Ahead Log (persistent block storage) to delay writing to COS
  - Optimized for Db2 use case
    - Multiple RocksDB Column Families per Table to avoid LSM L0 compaction bottleneck
    - Refined RocksDB Write Buffer Manager MemTable victimization to avoid early flushes
- Queries
  - Concurrent queries of varying complexity
  - Utilizes Db2's prefetching to avoid synchronous reads from COS

High Performance with COS Latency

	Parallelism	Batching	Asynchronous	Caching
Bulk Insert	• SST file generation • SST file uploads to COS	• SST file generation target size of 32 MB	• SST file generation • SST file upload (before commit)	• SST file generation • Write through cache
Trickle Insert	• Uploads to COS	• Write buffer size of 32 MB (leading to SST files of 32 MB)	• Write buffer flushes • Uploads to COS (durability via WAL)	• Write through cache
Queries	• Reads from COS and Caching Tier	• Compaction • Prefetching from COS and Caching Tier	• Prefetching from COS and Caching Tier	• Reads from Caching Tier

Optimizing Bulk Insert: Bulk Write Mode



- Data written in parallel (by CG and IR) and async to SST files on Caching Tier

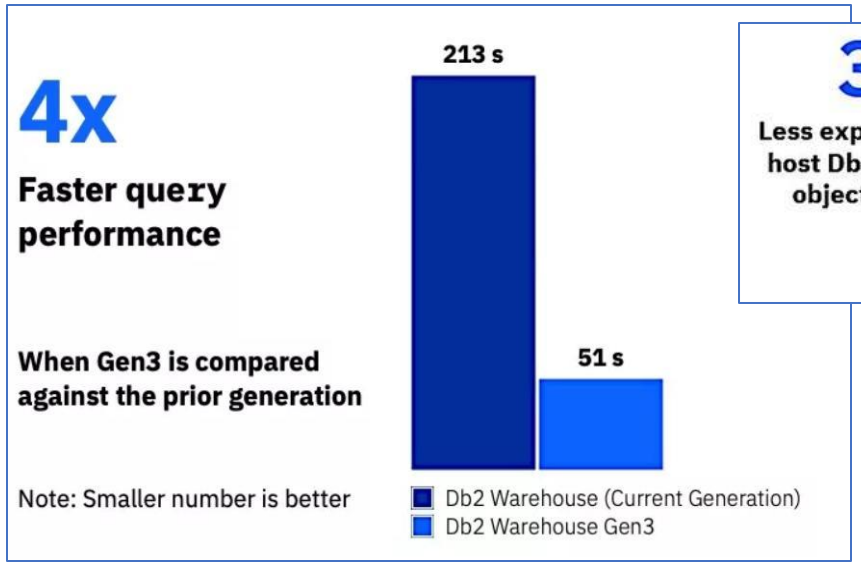
SST file for Insert Range 1 of Column Group 2

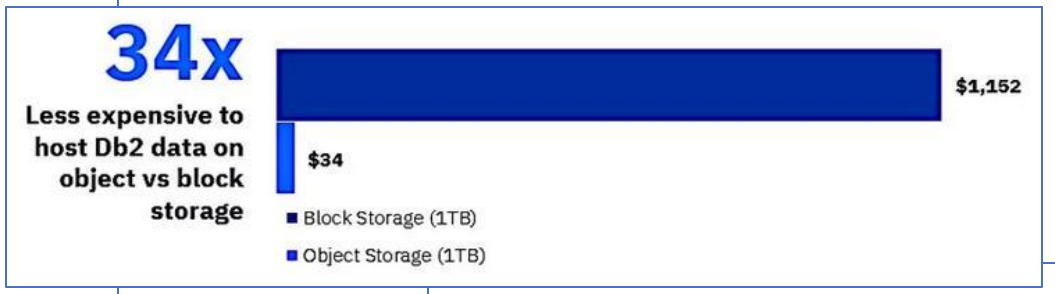
- SST files uploaded in parallel and async to COS

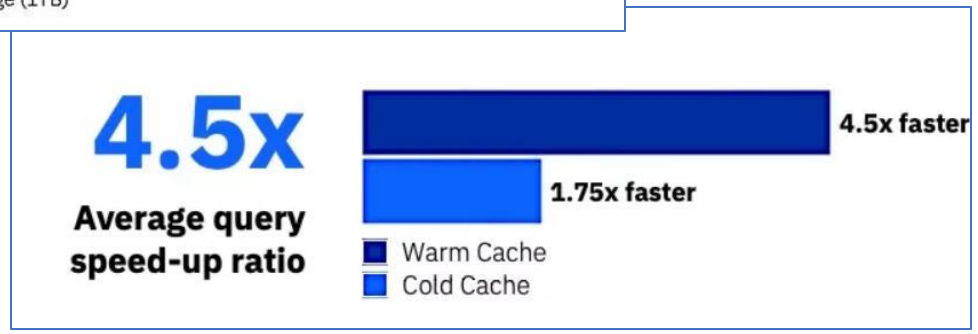
- Data batched into SSTs of 32 MB
- SST files ingested into lowest level of LSM tree, avoiding local tier, flush logic and compaction

NCOS Tables Outperform Previous Gen

- Big performance gains and massive storage cost savings over previous generation of Db2 Warehouse on Cloud
  - 4x faster query performance
  - 4.5x average query speed up (cold cache)
  - 34x less expensive storage costs
  - Read more here: https://www.ibm.com/blog/db2-warehouse-delivers-4x-faster-query-performance-than-previously-while-cutting-storage-costs-by-34x/

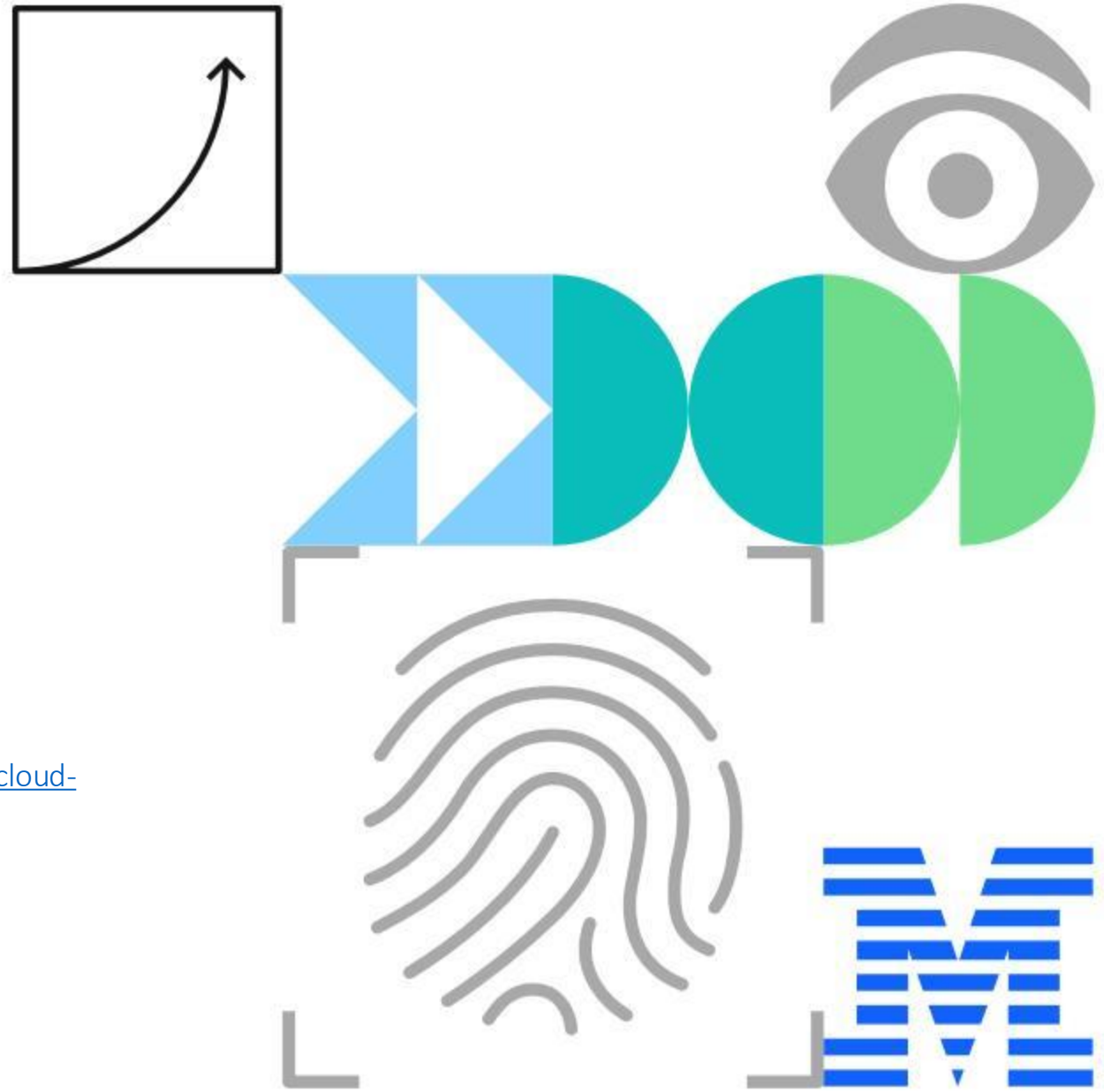






IBM **TechXchange**

IBM Db2 LUW Webinar Series



Thank you!

Continue the Conversation:

Robert Hooper (robert.christopher.hooper@ibm.com)
Kostas Rakopoulos (kostasr@ca.ibm.com)

Discussion Forum:
https://community.ibm.com/community/user/datamanagement/discussion/db2-cloud-object-storage-architecture-and-performance-considerations-webinar

Join the Db2 Community!
https://community.ibm.com/community/user/datamanagement/home

IBM

Backup

Demo Screenshot – Trickle Insert Monitoring

MEMBER	LOCAL_TIER_WRITES	CACHING_TIER_WRITES	COMPACTION_WRITES	REMOTE_TIER_WRITES
0	7836	3543	10	13
1	8045	9193	59	50
2	7971	3985	25	28
3	8067	3686	26	28
4	7997	3740	16	20
5	8079	4035	27	27
6	7963	4555	28	33
7	7909	5466	32	37

MEMBER	AVG_LOCAL_TIER_WRITE_TIME	AVG_CACHING_TIER_WRITE_TIME	AVG_COMPACTION_WRITE_TIME	AVG_REMOTE_TIER_WRITE_TIME
0	1.11	0.18	552.70	416.00
1	1.07	0.20	437.23	213.36
2	1.09	0.25	315.24	290.64
3	1.08	0.25	279.53	268.96
4	1.10	0.26	455.00	319.60
5	1.08	0.19	300.29	295.25
6	1.08	0.20	344.28	234.72
7	1.12	0.22	367.53	224.67

Demo Screenshot – Query Monitoring (Cold Buffer Pool)

MEMBER	LOGICAL_READS	PHYSICAL_READS	AVG_PHYSICAL_READ_TIME
0	28417	0	0.00
1	28816	0	0.00
2	28622	0	0.00
3	28697	0	0.00
4	28639	0	0.00
5	28584	0	0.00
6	28685	0	0.00
7	28640	0	0.00

MEMBER	CACHING_TIER_LOGICAL_READS	CACHING_TIER_PAGES_FOUND	AVG_CACHING_TIER_P_READ_TIME	CACHING_TIER_HITRATIO
0	2488	2488	0.57	100.00
1	2489	2489	0.38	100.00
2	2483	2483	0.52	100.00
3	2498	2498	0.42	100.00
4	2483	2483	0.41	100.00
5	2482	2482	0.41	100.00
6	2494	2494	0.41	100.00
7	2491	2491	0.43	100.00

Demo Screenshot – Query Monitoring (Cold Caching Tier)

MEMBER	LOGICAL_READS	PHYSICAL_READS	AVG_PHYSICAL_READ_TIME
0	28849	7	543.28
1	28836	3	702.66
2	28632	5	604.40
3	28409	5	531.80
4	28307	3	1103.00
5	28670	4	678.00
6	28765	5	465.60
7	28423	6	464.16

MEMBER	CACHING_TIER_LOGICAL_READS	CACHING_TIER_PAGES_FOUND	AVG_CACHING_TIER_P_READ_TIME	CACHING_TIER_HITRATIO
0	2485	2478	1.76	99.71
1	2486	2483	0.68	99.87
2	2480	2475	1.17	99.79
3	2496	2491	0.69	99.79
4	2480	2477	0.99	99.87
5	2482	2478	0.95	99.83
6	2491	2486	0.73	99.79
7	2491	2485	0.77	99.75