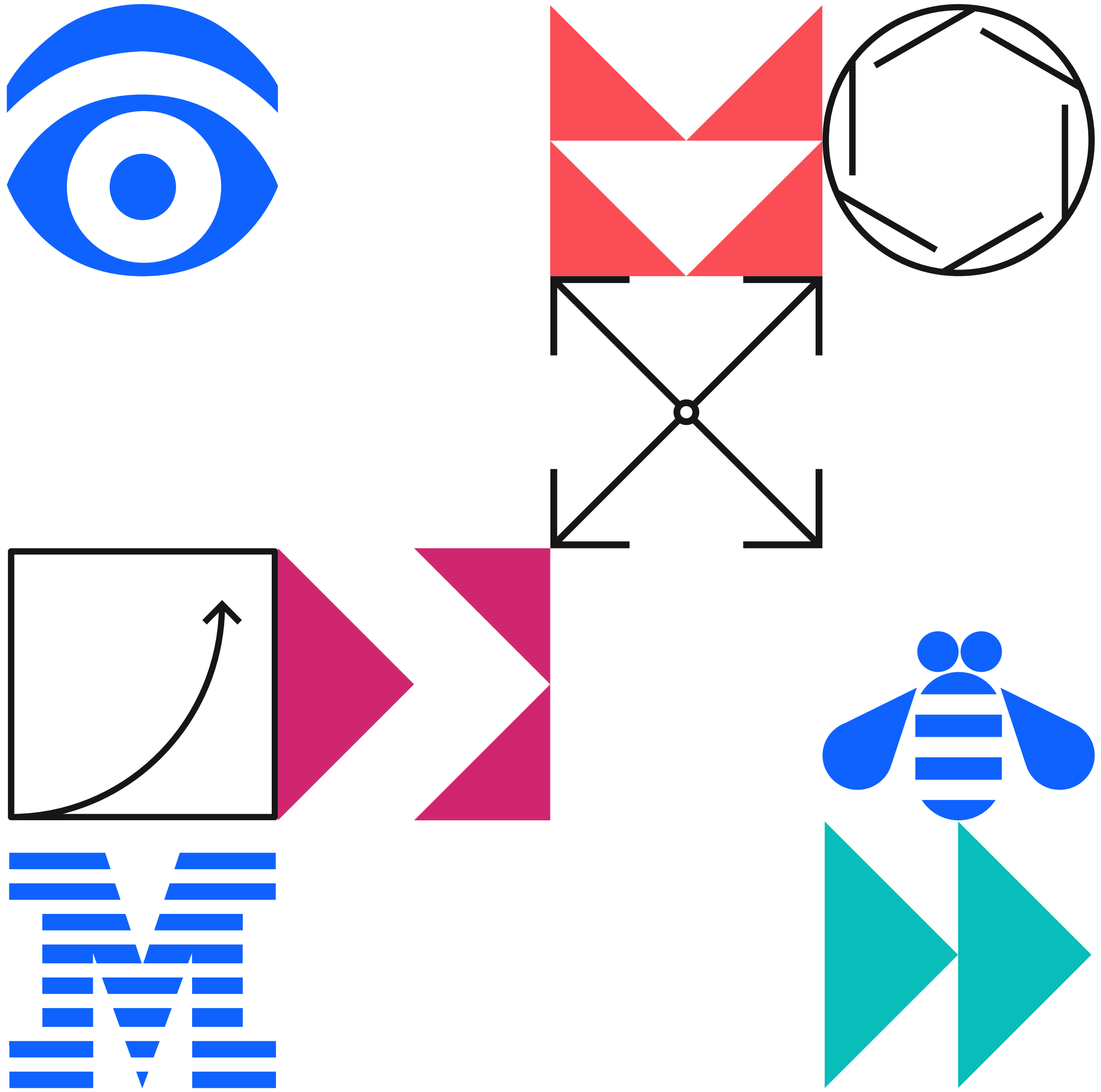
Next Generation Db2 Warehouse: Native Cloud Object Storage and Data Lake Integration

Christian Garcia-Arellano

STSM and Db2 Senior Architect

Session LUW-03



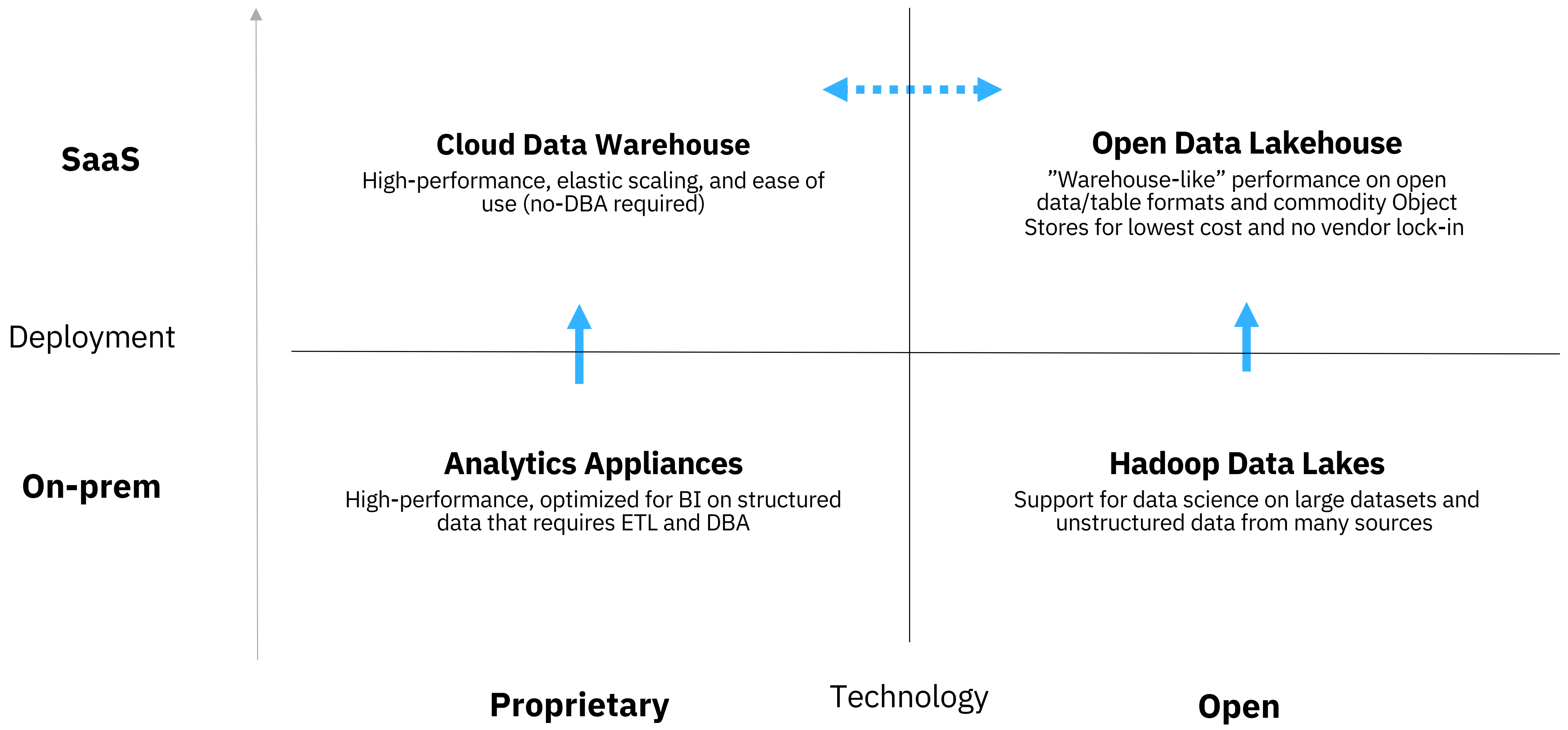
Agenda

Analytic Database Landscape

Analytics Repositories Market Dynamics

Major disruptions are driving the growth in the analytics repositories market **from on-prem to SaaS** and **blending proprietary and open technologies**

Analytics Repositories Market Landscape



Cloud Data Warehouse
High-performance, elastic scaling, and ease of use (no-DBA required)

Open Data Lakehouse
"Warehouse-like" performance on open data/table formats and commodity Object Stores for lowest cost and no vendor lock-in

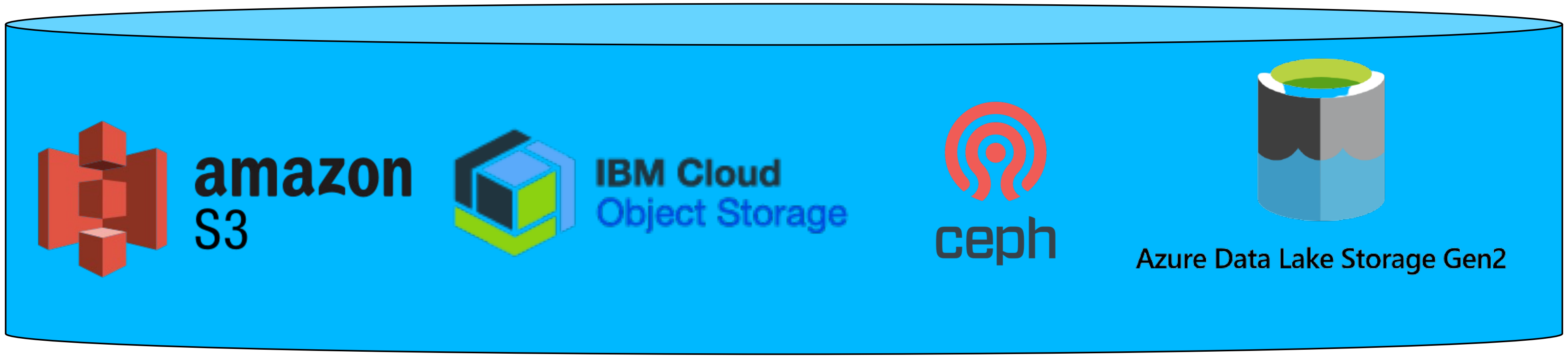
Analytics Appliances
High-performance, optimized for BI on structured data that requires ETL and DBA

Hadoop Data Lakes
Support for data science on large datasets and unstructured data from many sources

Sources: IDC Data Management Forecast (November 2021), IDC BDA Forecast (June 2021), MI modeling

Common Data Lake Storage

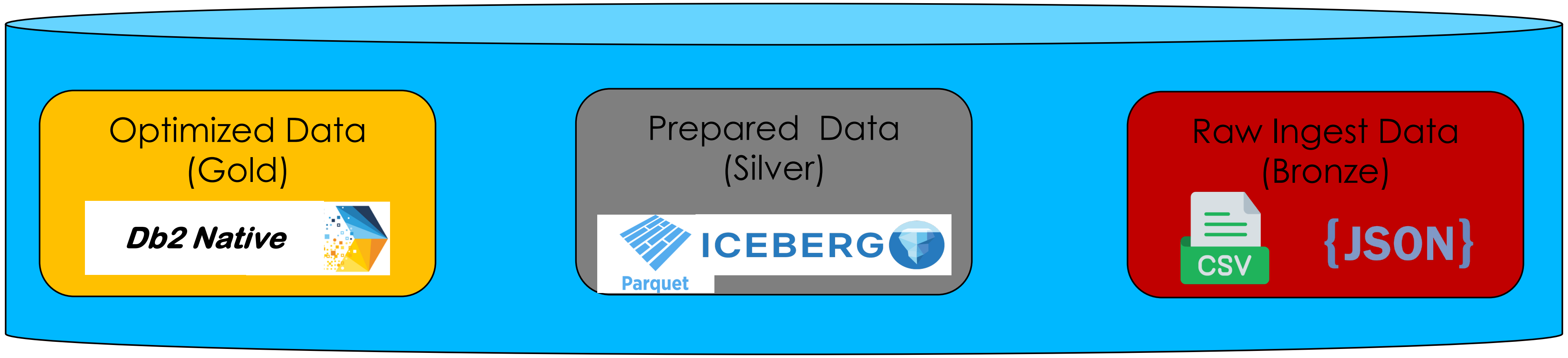
- √ Low cost
- √ Near unlimited scalability
- √ Extreme durability + reliability (99.999999999%)
- √ High throughput
- 🛑 High latency (but can be compensated for)



Data Lake - Cloud Object Storage

Purpose Optimized Data Formats

Proprietary / Optimized Format	Open / Interoperable Format	Source Format
Well Defined Schema	Flexible Schema	Low Performance
Highest Performance	Medium Performance	Zero transformation
High Volume Transactions	Low Volume Transactions	Not transactional



Data Lake - Cloud Object Storage

Purpose Optimized Engines

Business Intelligence | **Predictive Analytics** | **Data Exploration** | **Data Engineering**

Watson Query

Polyglot SQL
Virtualization
Governance

Db2 Warehouse
(Next Generation)

High Performance BI + Analytics
Petabyte Scale
High Concurrency
High Volume Transactions

presto

Interactive Queries + Adhoc Analytics
Petabyte Scale
Lightweight Scalable Engines
Low Volume Batch Transactions

APACHE Spark

Large Scale Batch Analytics
Exabyte Scale
Data Engineering + Transformation
Low Volume Batch Transactions

Optimized Data
(Gold)

Db2 Native

Highest Performance

Prepared Data
(Silver)

Parquet ICEBERG

Multi-Purpose

Raw Ingest Data
(Bronze)

Source Format

More Structured (Defined Schema) ***Data Lake – Cloud Object Storage*** *Less Structured (Schemaless)*

Db2 Warehouse Next Generation

Next Generation Db2 Warehouse

Warehouse

Full warehousing SQL + performance with tables in cloud object storage

Lowers storage costs and simplifies storage tiering with local NVME caching

High performance bulk + streaming IUD with full transactional support

Lake

Data lake integration with open data formats like Iceberg through external tables

Warehouse can access both local and open data directly in the “data lake”

Data lake table access can be optimized using an MQT cache in native format

Next Generation Db2 Warehouse

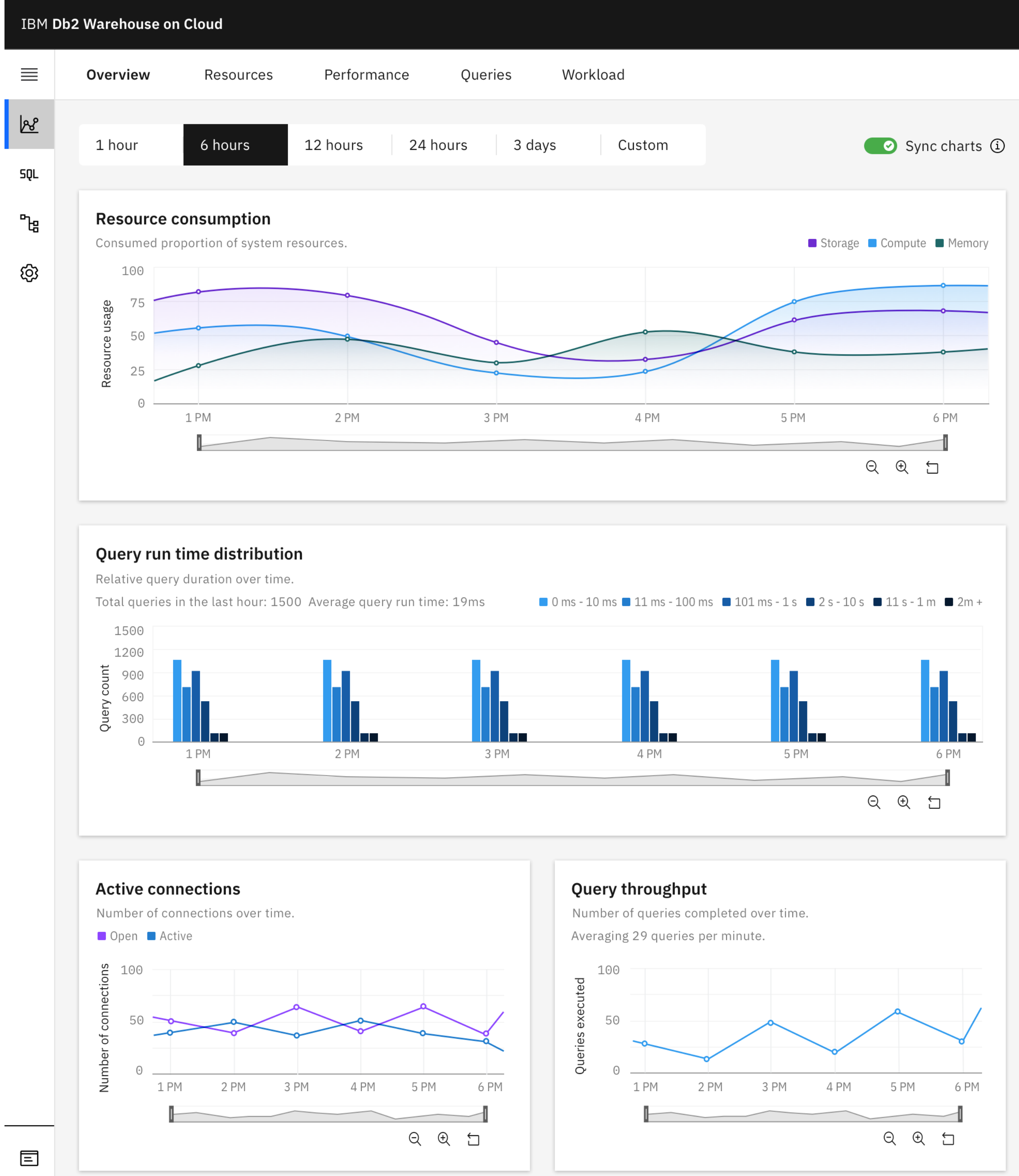
Managed Environments

Key new features:

- **Native column-organized table storage in Amazon S3, significantly decreasing the cost of storing data without sacrificing performance**
- **Query multiple open data formats (Iceberg, Parquet, AVRO, ORC, CSV, JSON and more) leveraging existing compute resources dedicated to the warehouse**
- **Integration with Watsonx.data lakehouse engine with sharing of data catalogs and S3 buckets**

Other features:

- **Fully managed cloud data warehouse scaling up to 1440 cores (2880 vCPUs) per cluster, multi-petabyte-scale, multi-performant storage with seamless transition between compute tiers**
- **Tiered storage support for Amazon S3 and Block Storage**
- **Storage auto-increase for Block Storage on set threshold ensuring you never run out of storage for your workloads**
- **Cross-region snapshot backup to AWS S3 for disaster recovery**
- **Self-service maintenance windows for product and database engine updates**
- **New APIs for scaling, updates, backup/restore, logging**
- **Granular, schema-level backup/restore to S3, restoring only the data you need**


IBM **Db2 Warehouse on Cloud**

Resource consumption

Consumed proportion of system resources.

Query run time distribution

Relative query duration over time.

Total queries in the last hour: 1500 Average query run time: 19ms

Active connections

Number of connections over time.

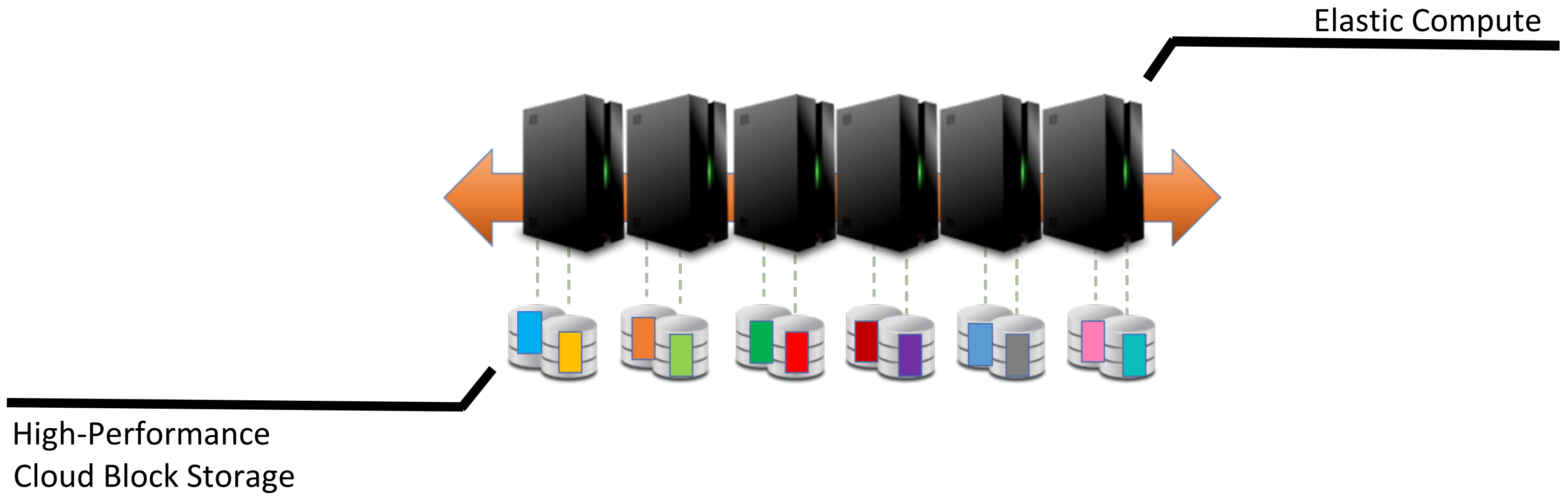
Query throughput

Number of queries completed over time.

Averaging 29 queries per minute.

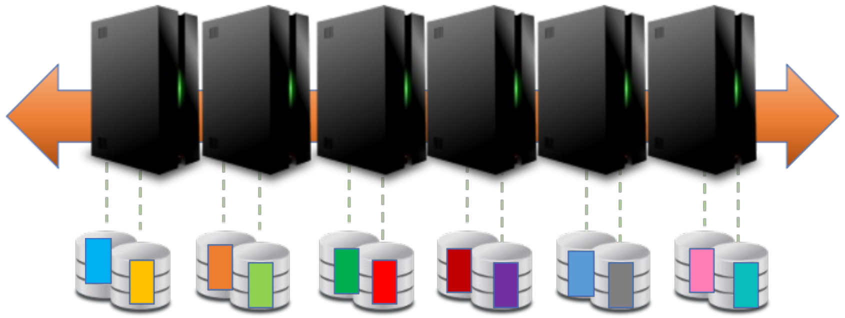
Native Cloud Object Storage

Evolution of the Db2 Warehouse Storage Architecture



Evolution of the Db2 Warehouse Storage Architecture

- ✓ ***High Performance***
- ✓ ***Elastic***
- 🚫 ***Tightly Coupled***
- 🚫 ***Expensive***



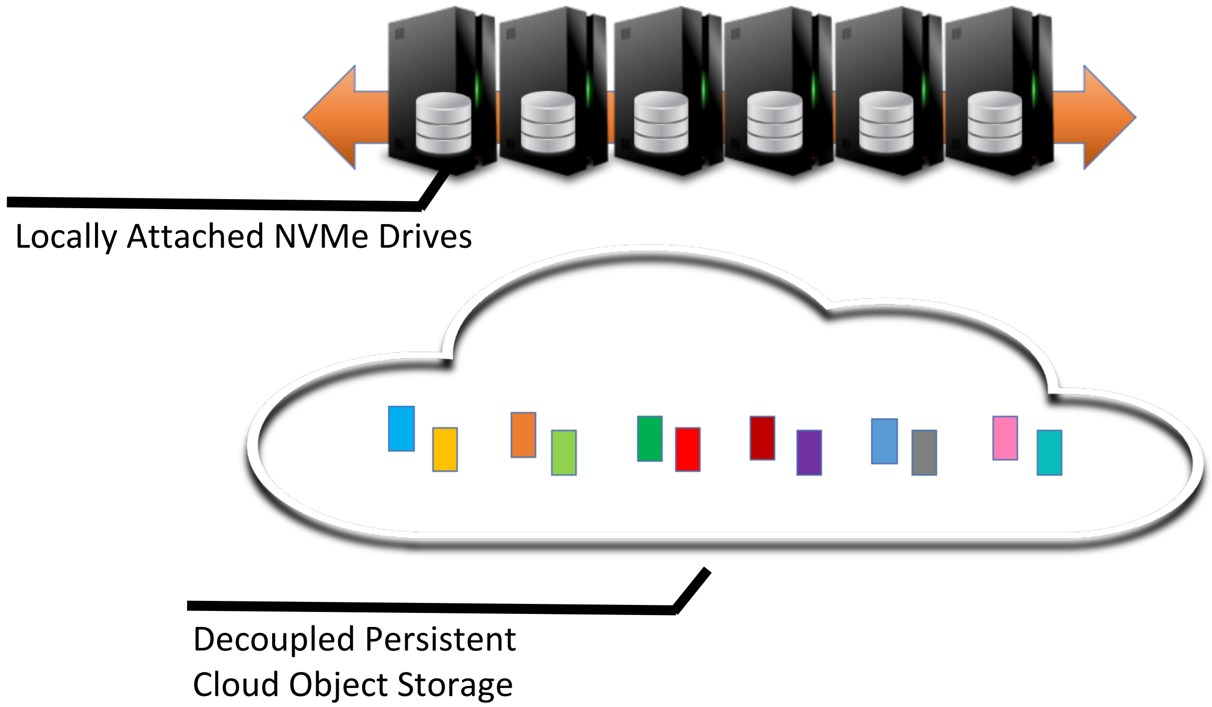
Next Generation Db2 Warehouse Storage Architecture

✓ ***High Performance***

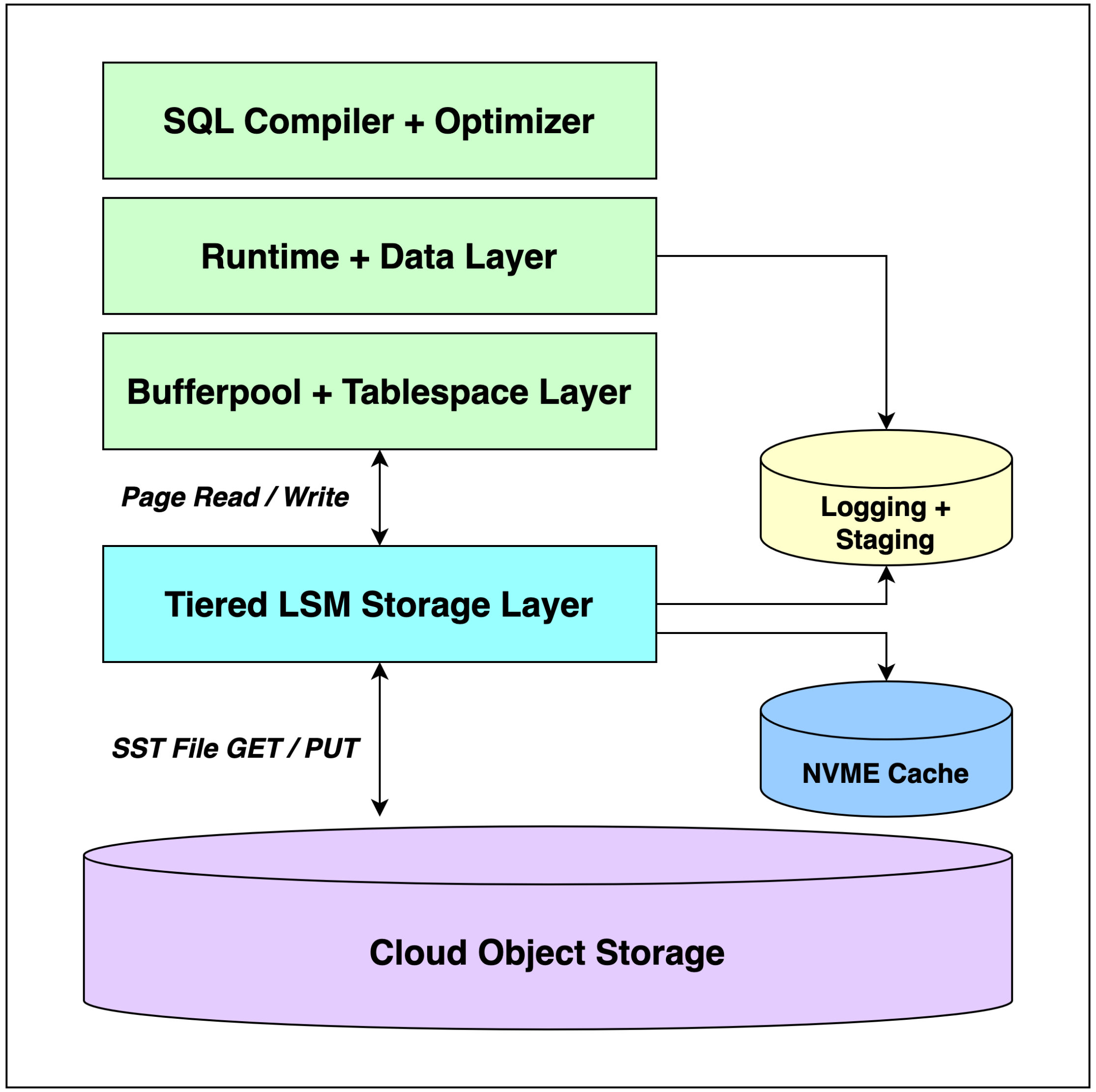
✓ ***Elastic***

✓ ***Decoupled***

✓ ***Cost Efficient***

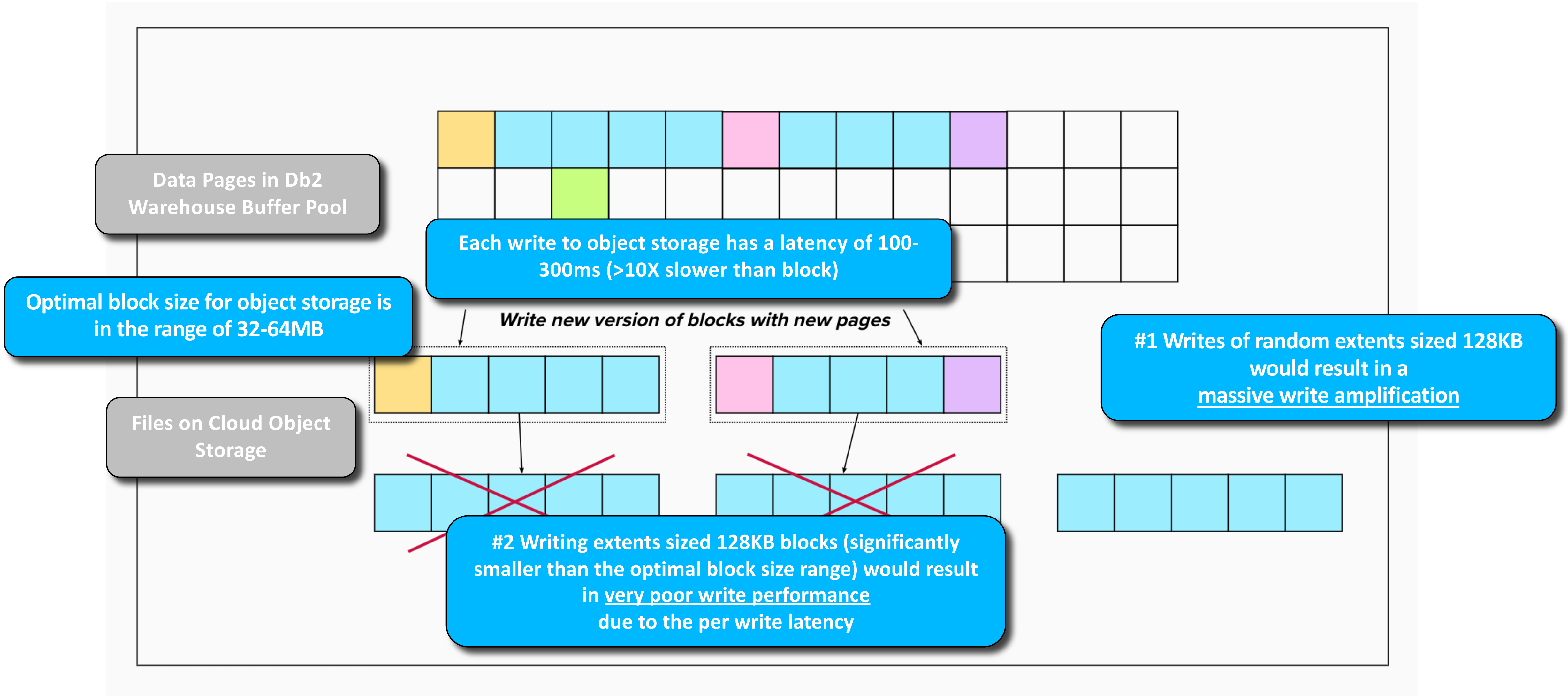

Locally Attached NVMe Drives
Decoupled Persistent Cloud Object Storage

Native Cloud Object Storage architecture



- Existing Db2 component stack down through bufferpool + tablespace layer
- Existing Db2 logging maintains high performance for trickle feed
- Three new elements in new native cloud storage layer:
  1. An LSM tree storage organization to efficiently store Db2 native pages on cloud object storage.
  2. A novel data clustering technique that exploits the self-clustering capabilities of the LSM tree.
  3. A multi-tiered cache that adds a local NVMe component to enable high performance query processing and bulk ingest.

Pitfalls of a naïve storage model



Data Pages in Db2 Warehouse Buffer Pool

Each write to object storage has a latency of 100-300ms (>10X slower than block)

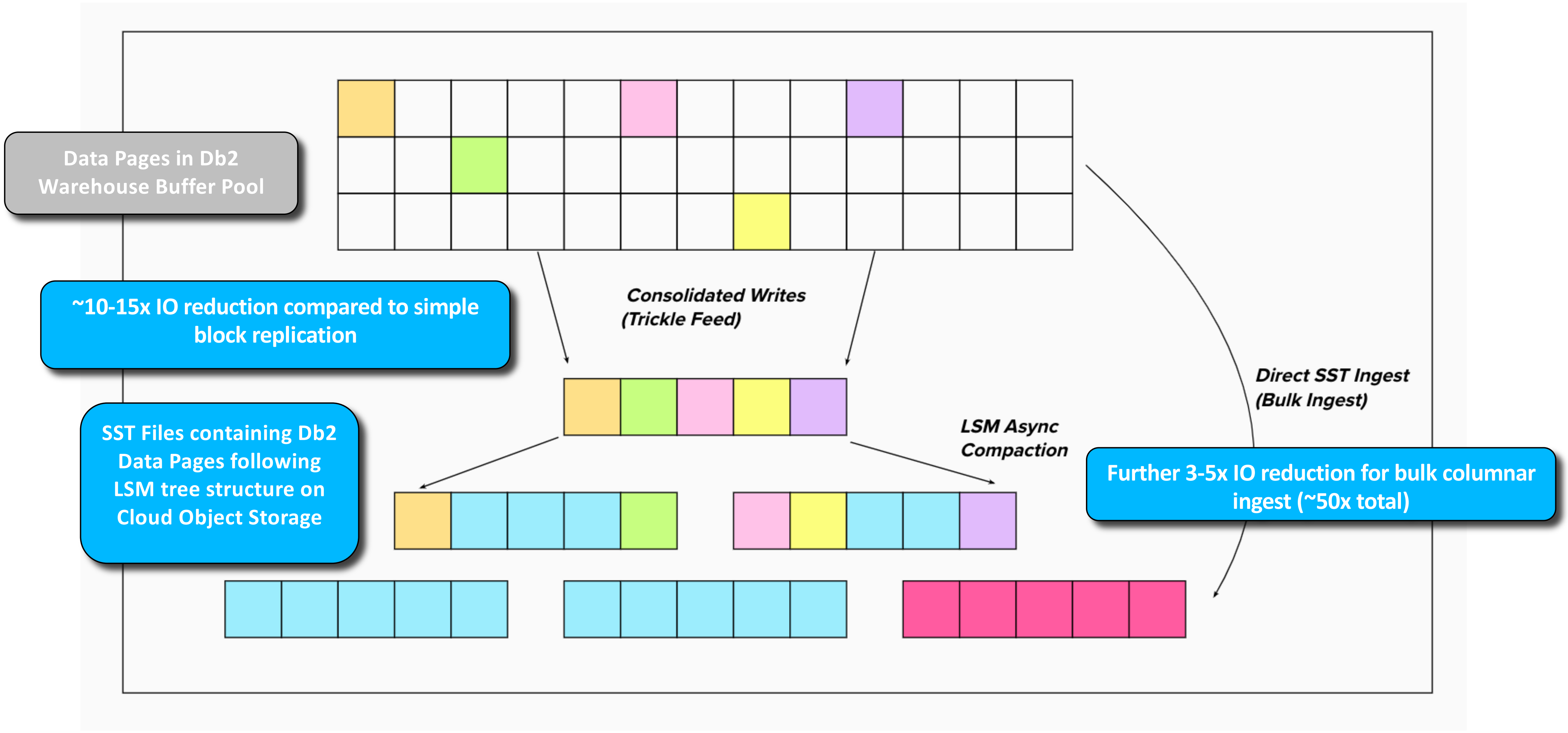
Optimal block size for object storage is in the range of 32-64MB

#1 Writes of random extents sized 128KB would result in a <u>massive write amplification</u>

Files on Cloud Object Storage

#2 Writing extents sized 128KB blocks (significantly smaller than the optimal block size range) would result in <u>very poor write performance</u> due to the per write latency

#1 LSM Tree based page IO

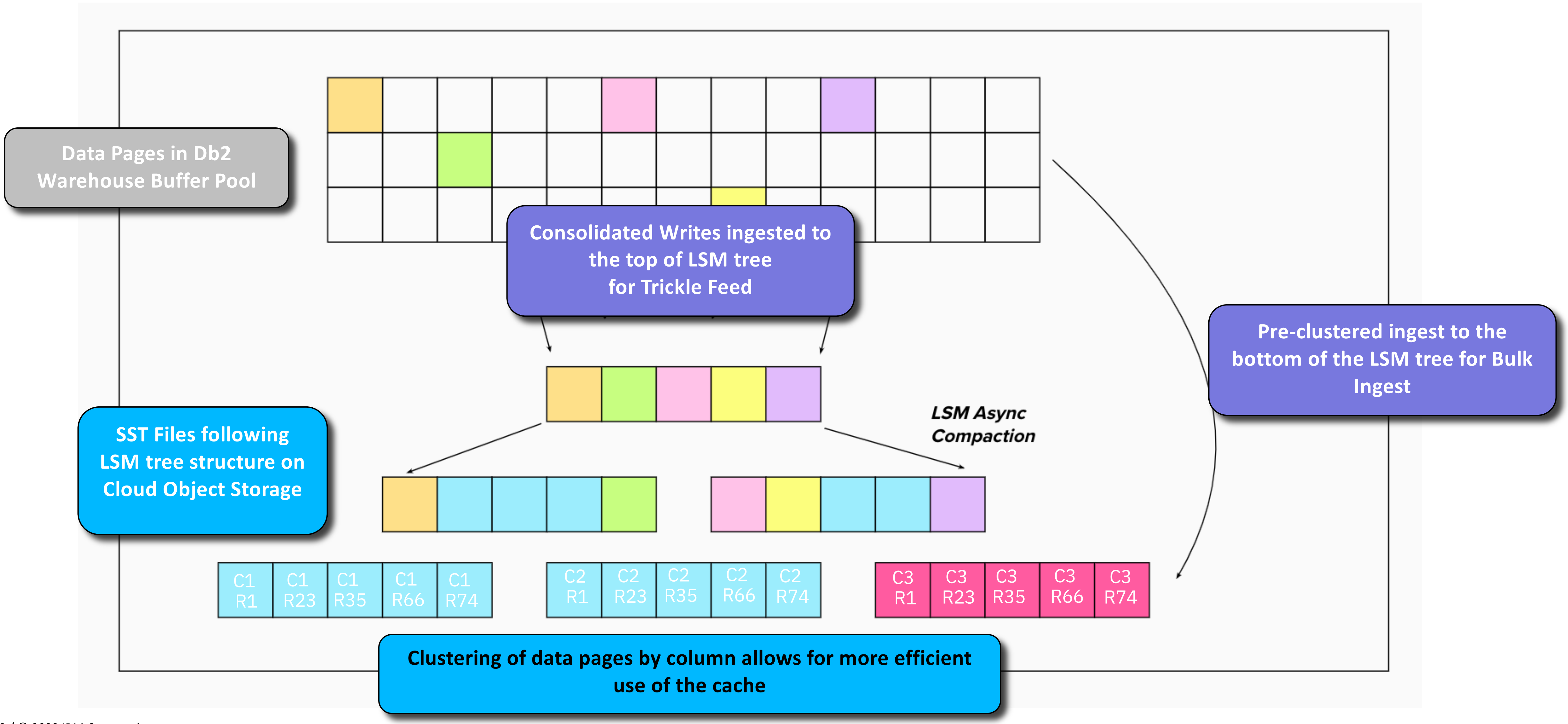

Data Pages in Db2 Warehouse Buffer Pool
Consolidated Writes (Trickle Feed)
Direct SST Ingest (Bulk Ingest)
LSM Async Compaction

~10-15x IO reduction compared to simple block replication

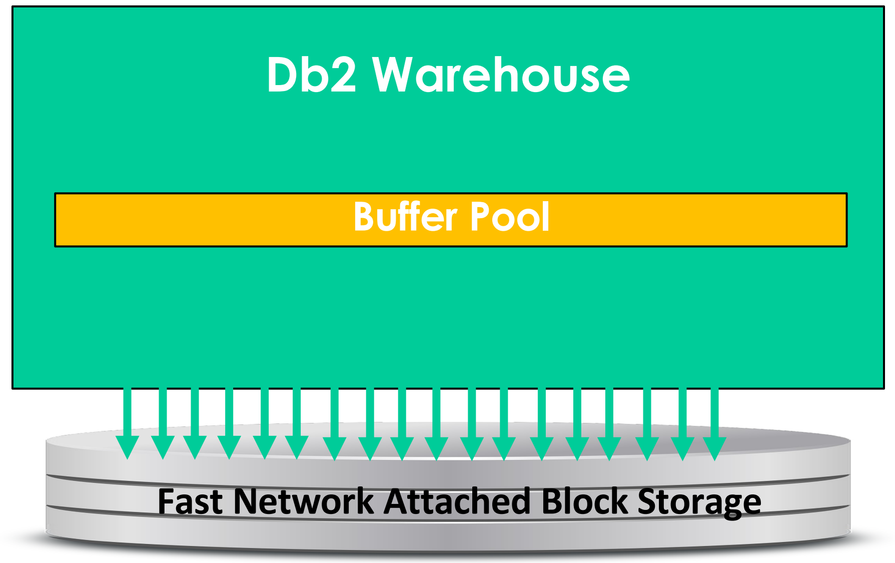
SST Files containing Db2 Data Pages following LSM tree structure on Cloud Object Storage

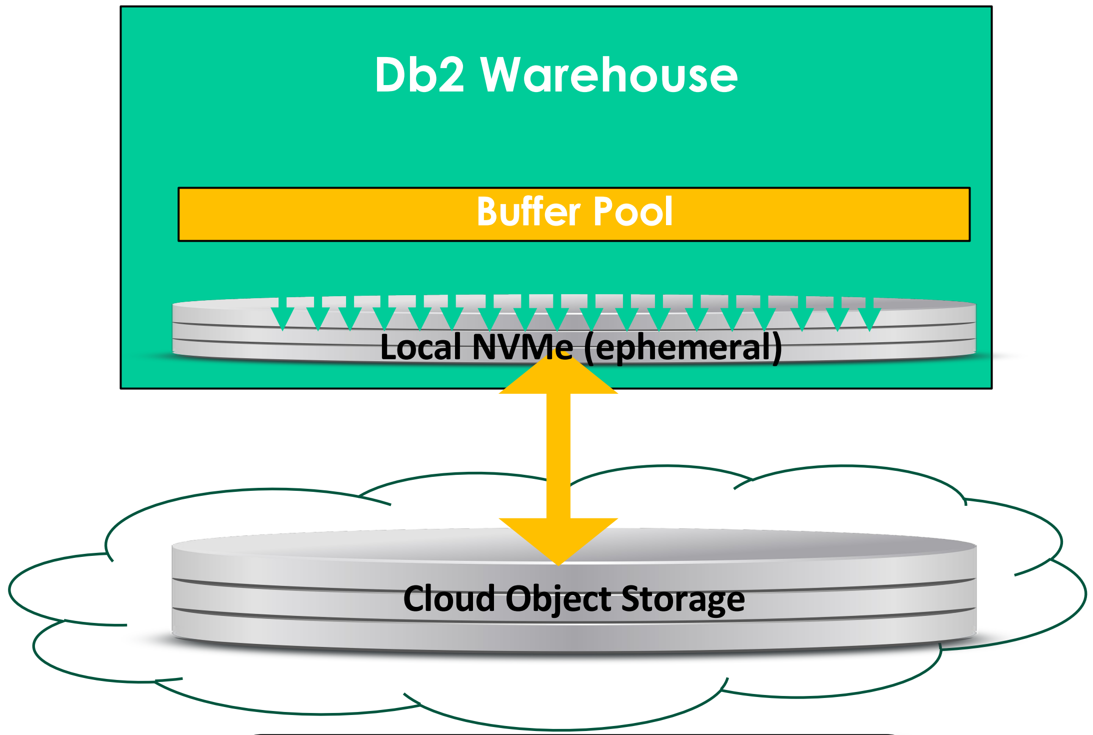
Further 3-5x IO reduction for bulk columnar ingest (~50x total)

#2 Column Group Clustering within LSM tree


Data Pages in Db2 Warehouse Buffer Pool
Consolidated Writes ingested to the top of LSM tree for Trickle Feed
Pre-clustered ingest to the bottom of the LSM tree for Bulk Ingest
LSM Async Compaction
SST Files following LSM tree structure on Cloud Object Storage
C1 R1
C1 R23
C1 R35
C1 R66
C1 R74
C2 R1
C2 R23
C2 R35
C2 R66
C2 R74
C3 R1
C3 R23
C3 R35
C3 R66
C3 R74
Clustering of data pages by column allows for more efficient use of the cache

#3 Multi-tiered Cache on Local NVMe drives





1000 page writes/reads to/from fast network attached block storage @ 10-30ms latency each (6 IOPS/GB)

1000 pages writes/reads to/from local NVME (ephemeral) + 1 PUT/GET to S3 @ 100-300ms latency each

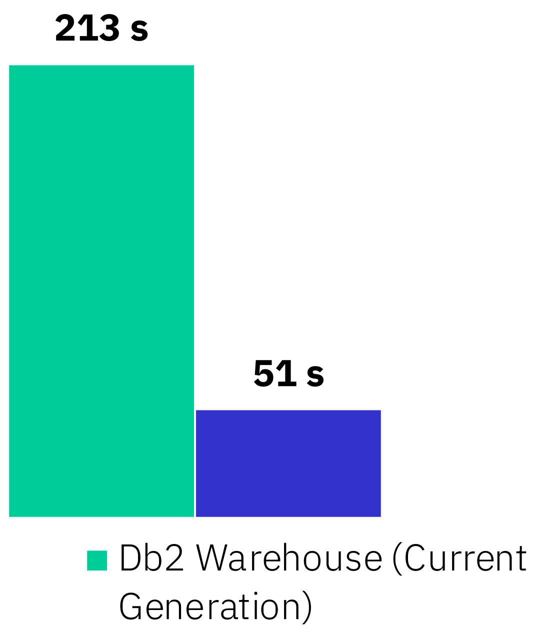
Db2 Warehouse Gen3
Performance numbers comparing Db2 Warehouse current generation vs Gen3

4x

Faster query performance

When Gen3 is compared against the prior generation

Note: Lower number is better



IBM Big Data Insight (BDI) Benchmark
simulates real-world deep analytics, reporting, and dashboard queries

10TB Db2 data warehouse
residing either on block storage (current generation) or object storage (Gen3)

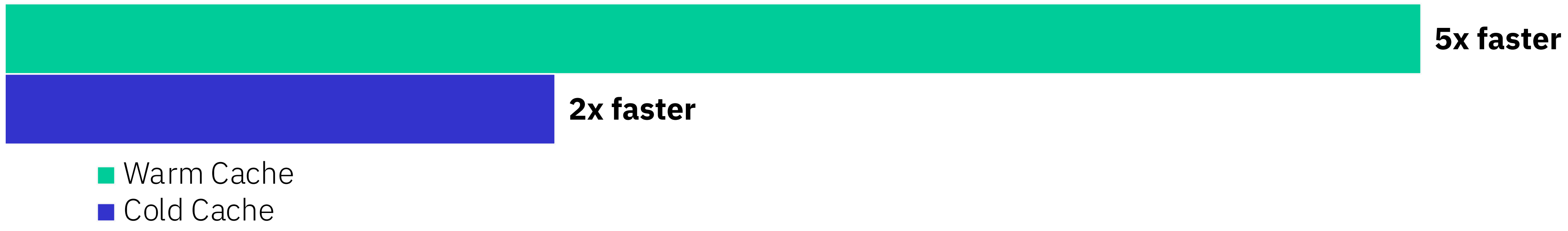
16 concurrent users
running a variety of ML, reporting, and dashboard queries

Cold cache start
for both the in-memory buffer pools or the NVMe cache

Db2 Warehouse Gen3
Performance numbers comparing Db2 Warehouse current generation vs Gen3

4.5x

Average query speed-up ratio



TPC-DS benchmark
running industry standard queries

10TB Db2 data warehouse
residing either on block storage (current generation) or object storage (Gen3)

99 query
serial test running SQL statements sequentially

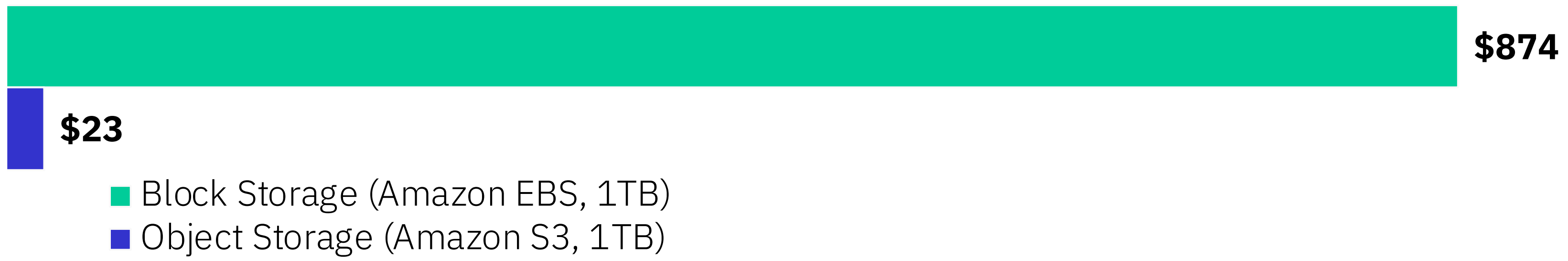
Multi-temperature test
running queries on both a cold and warm cache

Db2 Warehouse Gen3

Performance numbers comparing Db2 Warehouse current generation vs Gen3

38x

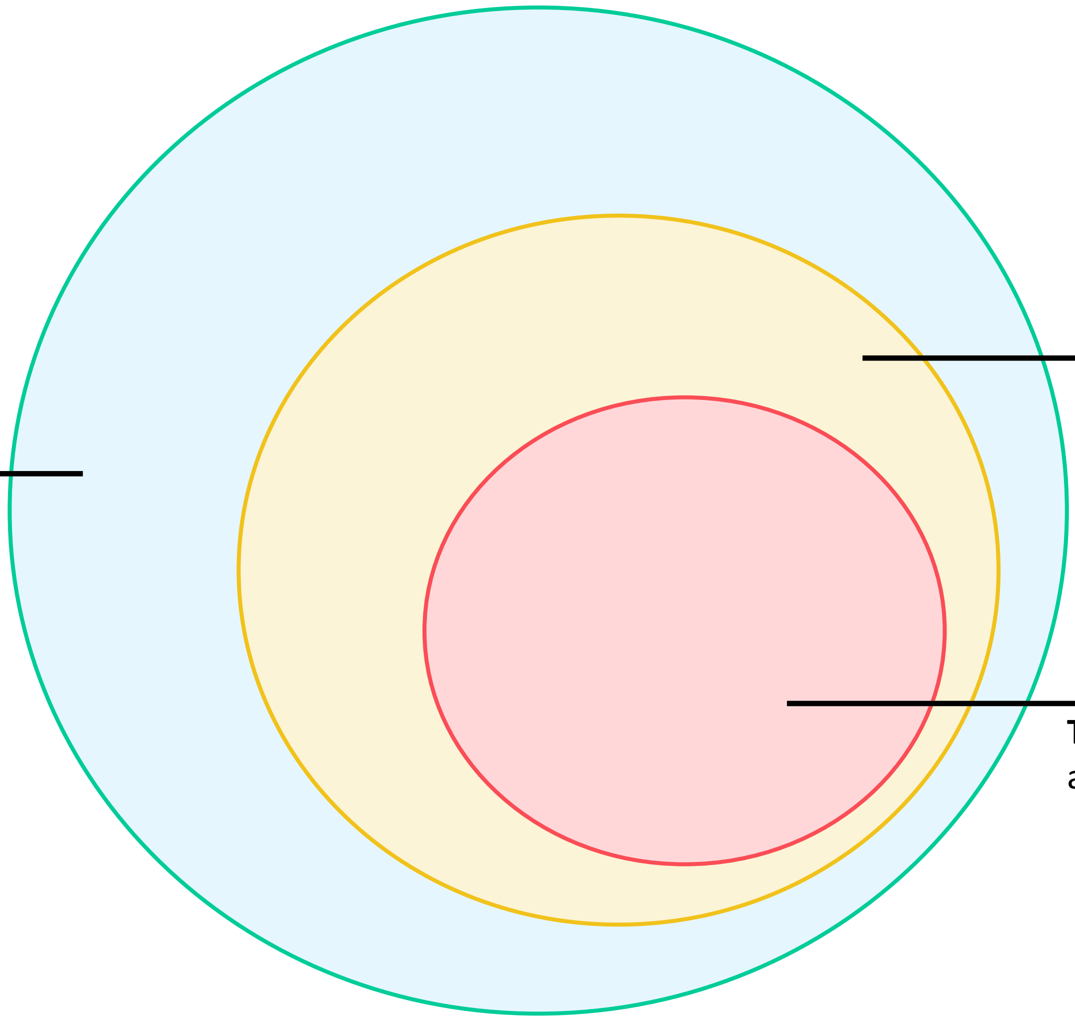
Less expensive to host Db2 data on object vs block storage[1]



Block Storage (Amazon EBS) vs Object Storage (Amazon S3)
Cost reflects Amazon's list price for block storage (various tiers & IOPS levels) required to host an incremental 1TB of Db2 data

[1] Block vs Object Storage comparison depicts difference between published prices for Amazon EBS 1TB of io1 at 6 IOPS/GB (and additional tiers to support Db2 data) vs Amazon S3. This metric is not an indicator of future storage pricing for Db2 Warehouse Gen 3.

Recap of storage hierarchy in Db2



Storage group defines the set of paths where data can be stored

Table spaces are logical storage groupings using storage defined by the storage group

Tables, indexes and large objects are stored within a table space

User Experience with Native Cloud Object Storage Support

1

A remote storage access alias defines an endpoint, path and credentials in cloud object storage.

2

A remote storage group is associated with a remote storage access alias instead of a set of local paths.

3

A remote table space is defined with a remote storage group

4

A column-organized table is created within a remote storage group

User Experience with Native Cloud Object Storage Support in Db2 Warehouse Gen3

1

The remote storage access alias IBMDEFAULTREMALIAS is pre-created using a pre-provisioned AWS S3 bucket.

2

The remote storage group IBMDEFAULTREMSG1 is pre-created.

3

Two remote table spaces OBJSTORESPACE1 and OBJSTORESPACEUTMP1 are pre-created.

4

Tables and DGTTs can be created within the two pre-created remote storage groups for out-of-the-box exploitation of the Native Cloud Object Storage.

How is this pre-setup done under the hood in Db2 Warehouse Gen3 ?

An AWS S3 bucket is created for the Db2 Warehouse instance by the cloud infrastructure and configured with role-based authentication and other set up required for backup and restore.

A remote storage access alias IBMDEFAULTREMALIAS is created using AWS role-based authentication:

```
db2 CATALOG STORAGE ACCESS ALIAS IBMDEFAULTREMALIAS VENDOR S3 SERVER https://s3.us-east-
2.amazonaws.com CONTAINER db2wh-db2wh-nos-perf-13-cos OBJECT db2u DBUSER db2inst1
```

A storage group IBMDEFAULTSG1 is associated with the default remote storage access alias IBMDEFAULTREMALIAS.

```
db2 CREATE STOGROUP IBMDEFAULTREMSG1 ON 'DB2REMOTE://IBMDEFAULTREMALIAS/'
```

Two remote table spaces are created using the remote storage group IBMDEFAULTSG1.

```
db2 CREATE TABLESPACE OBJSTORESPACE1 USING STOGROUP IBMDEFAULTREMSG1
db2 CREATE USER TEMPORARY TABLESPACE OBJSTORESPACEUTMP1 USING STOGROUP IBMDEFAULTREMSG1
```

EXPLORING REMOTE TABLE SPACES

- To create a column-organized table in the default remote table space, use the following:

```
CREATE TABLE CT1 (c1 INT NOT NULL, c2 INT NOT NULL)
       IN OBJSTORESPACE1
       ORGANIZE BY COLUMN
```

- To create a column-organized Declared Global Temporary table use the following:

```
DECLARE GLOBAL TEMPORARY TABLE GTT1 (c1 int not null, c2 int not null)
    IN OBJSTORESPACEUTMP1
    ORGANIZE BY COLUMN
```

MONITORING REMOTE TABLE SPACES

The remote table spaces are the only table spaces that have the CACHING_TIER column set to ENABLED.

```
SELECT VARCHAR(TBSP_NAME, 30) AS TBSP_NAME,
       MEMBER,
       TBSP_TYPE,
       CACHING_TIER
FROM TABLE(MON_GET_TABLESPACE('',-2)) AS T

TBSP_NAME                      MEMBER TBSP_TYPE  CACHING_TIER
------------------------------ ------ ---------- ------------
…
OBJSTORESPACE1                      0 DMS        ENABLED
OBJSTORESPACEUTMP1                  0 DMS        ENABLED
…
```

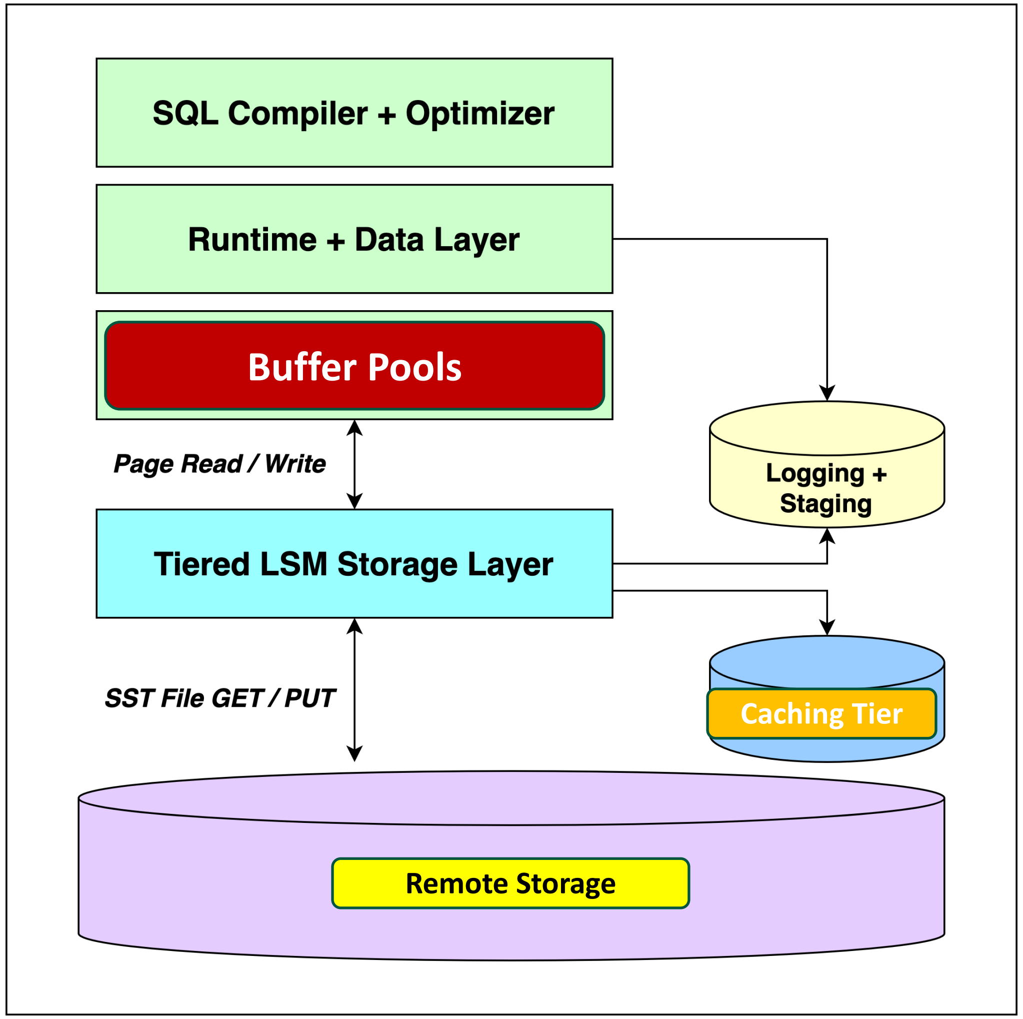
MONITORING REMOTE TABLE SPACES: READS

The Native COS read storage hierarchy has 3 levels:

1. A set of buffer pools for in-memory caching of data pages, shared between remote table spaces and non-remote table spaces.
2. A caching tier layer backed by fast locally-attached NVMe drives, for the extended local caching to maintain a larger working set than in-memory and to amortize the cost of accessing remote storage.

   Note: WAL is not monitored for READS

3. A remote storage layer, in Cloud Object Storage, when reading data pages are not currently cached in either of the two caching layers.



MONITORING REMOTE TABLE SPACES: READS

New monitoring elements were added or changed to expose the additional layers in the storage hierarchy.

Two pairs of examples:

POOL_COL_LBP_PAGES_FOUND: number of pages read (found) in BP.

POOL_COL_CACHING_TIER_PAGES_FOUND: number of pages read (found) in caching tier.

POOL_COL_P_READS: number of pages read from remote storage.

DIRECT_READ_TIME: this is time spent on direct access to the remote storage, excluding the caching tier.

CACHING_TIER_DIRECT_READ_TIME: For remote containers, this is the elapsed time in milliseconds required to perform the direct reads serviced using the caching tier.

MONITORING REMOTE TABLE SPACES: READS

Caching tier hit ratios expose the efficiency of the caching tier, for example:

- CACHING_TIER_DATA_HIT_RATIO_PERCENT: for pages that were found in the caching tier without needing to get them from remote storage.

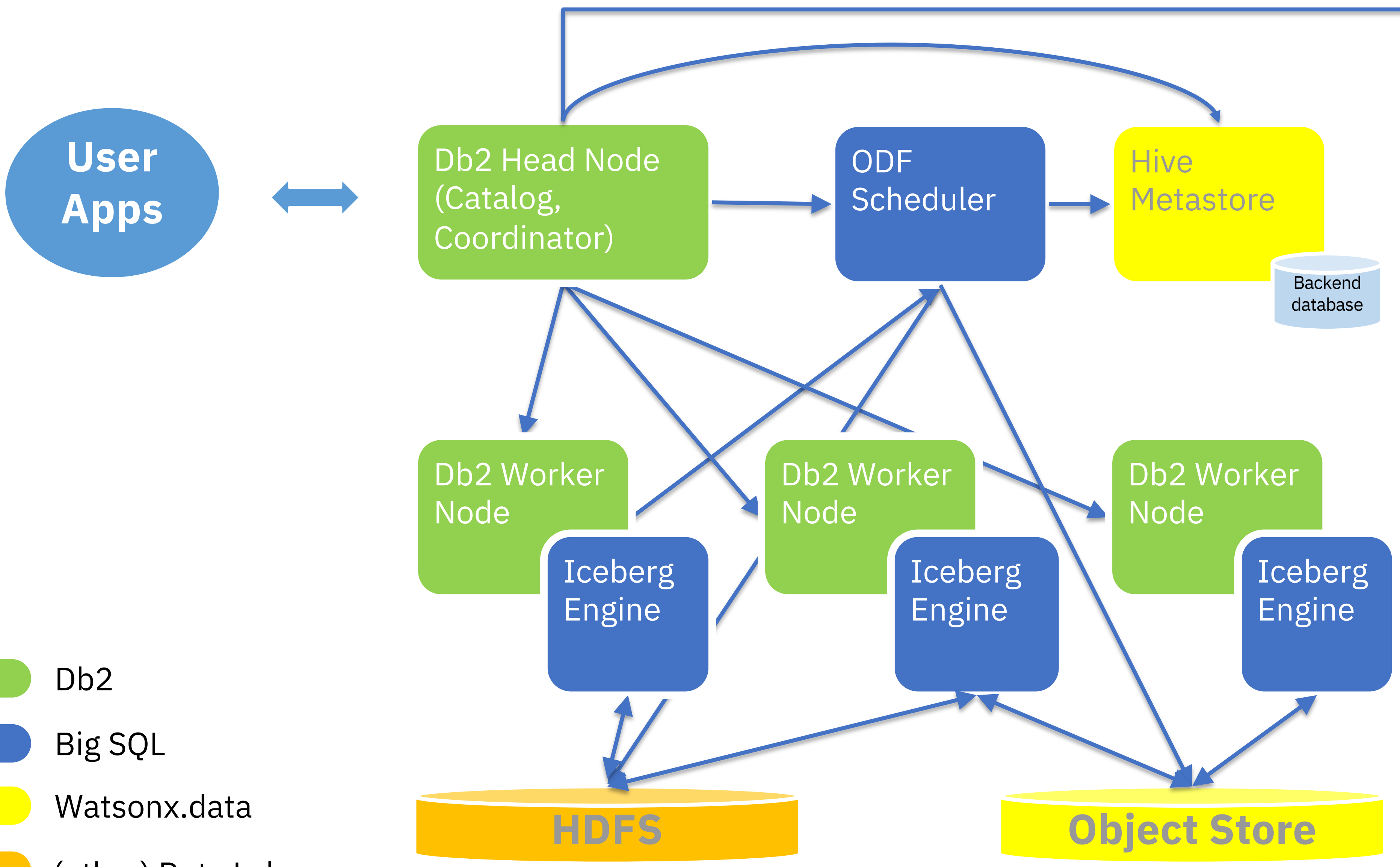
As usual with cache hit ratios, the higher the ratio the better the cache efficiency.

```
SELECT VARCHAR(TBSP_NAME, 30) AS TBSP_NAME,
       MEMBER,
       CACHING_TIER_DATA_HIT_RATIO_PERCENT
FROM SYSIBMADM.MON_TBSP_UTILIZATION

TBSP_NAME                      MEMBER CACHING_TIER_DATA_HIT_RATIO_PERCENT
------------------------------ ------ -----------------------------------
…
OBJSTORESPACE1                      0                              100.00
OBJSTORESPACEUTMP1                  0                              100.00
…
```

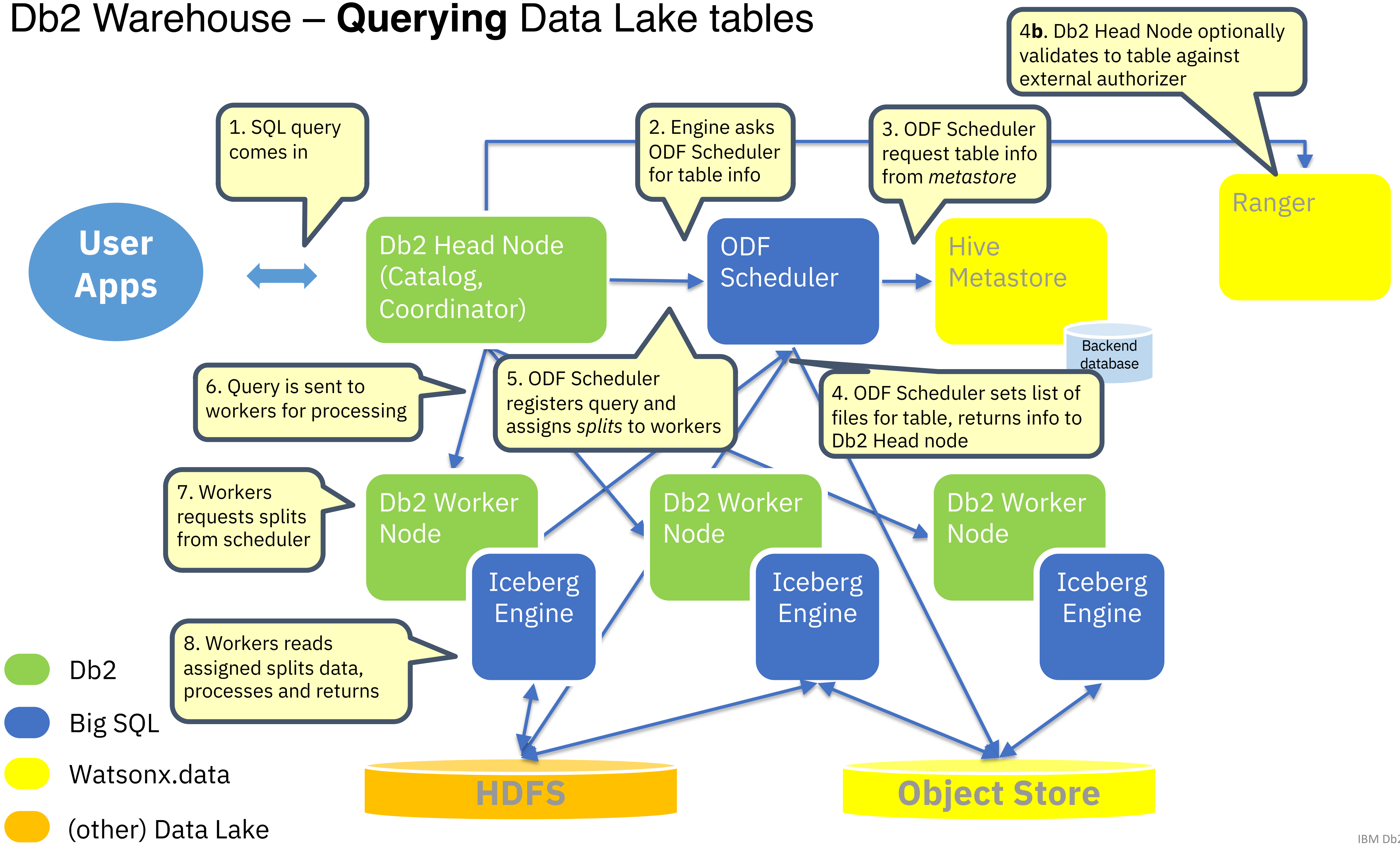
Data Lakehouse Integration

Db2 Warehouse – Data Lakehouse **Integration Components**



🟩	Db2
🟦	Big SQL
🟨	Watsonx.data
🟧	(other) Data Lake

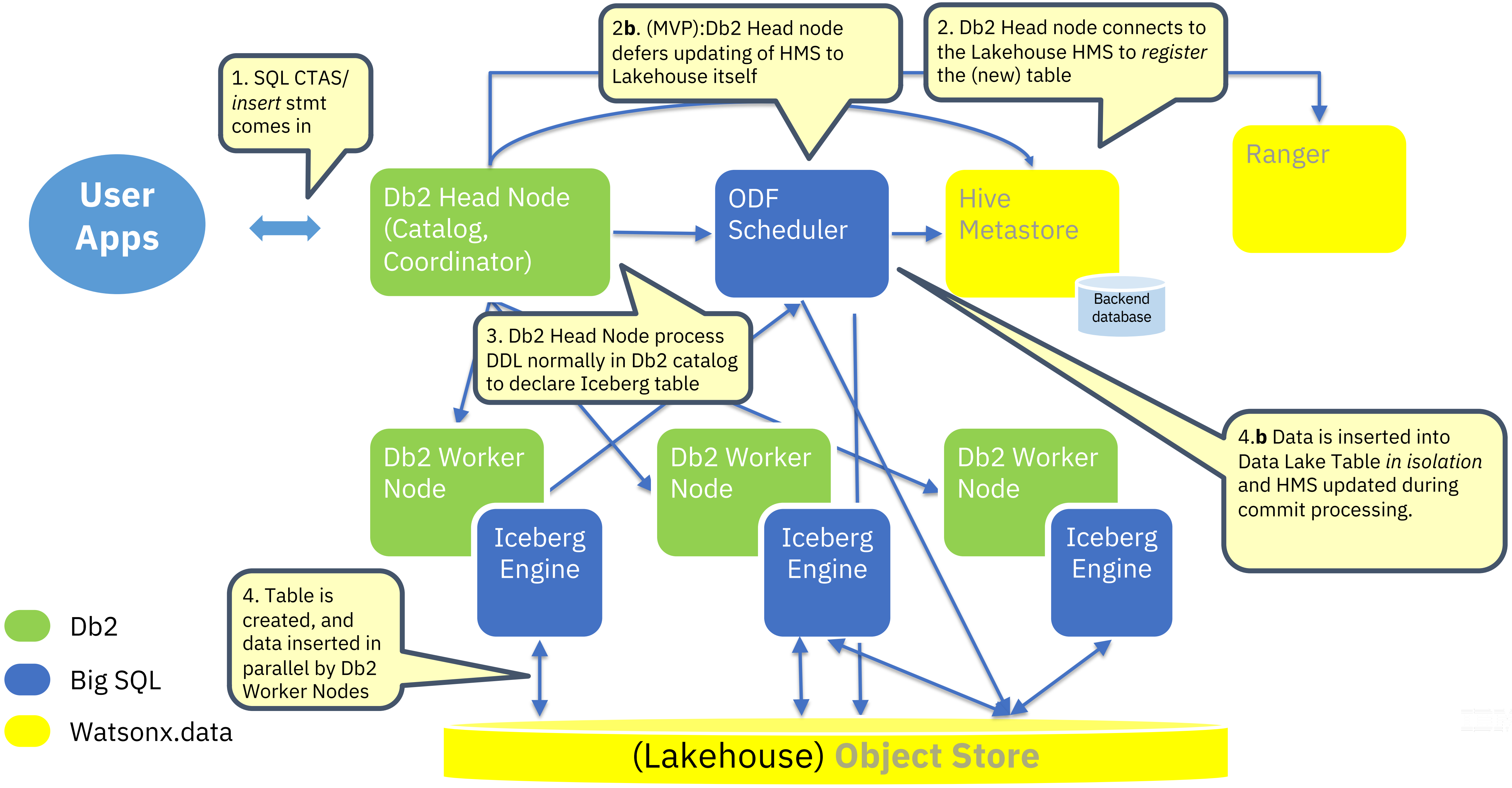
Db2 Warehouse – **Querying** Data Lake tables



User Apps
1. SQL query comes in
Db2 Head Node (Catalog, Coordinator)
2. Engine asks ODF Scheduler for table info
ODF Scheduler
3. ODF Scheduler request table info from *metastore*
Hive Metastore
Backend database
4**b**. Db2 Head Node optionally validates to table against external authorizer
Ranger
4. ODF Scheduler sets list of files for table, returns info to Db2 Head node
5. ODF Scheduler registers query and assigns *splits* to workers
6. Query is sent to workers for processing
7. Workers requests splits from scheduler
8. Workers reads assigned splits data, processes and returns
Db2 Worker Node
Iceberg Engine
Db2 Worker Node
Iceberg Engine
Db2 Worker Node
Iceberg Engine
HDFS
Object Store

🟩	Db2
🟦	Big SQL
🟨	Watsonx.data
🟧	(other) Data Lake

Db2 Warehouse – **Writing** to Data Lake tables


1. SQL CTAS/ *insert* stmt comes in
2**b**. (MVP):Db2 Head node defers updating of HMS to Lakehouse itself
2. Db2 Head node connects to the Lakehouse HMS to *register* the (new) table
User Apps
Db2 Head Node (Catalog, Coordinator)
ODF Scheduler
Hive Metastore
Ranger
Backend database
3. Db2 Head Node process DDL normally in Db2 catalog to declare Iceberg table
Db2 Worker Node
Iceberg Engine
Db2 Worker Node
Iceberg Engine
Db2 Worker Node
Iceberg Engine
4.**b** Data is inserted into Data Lake Table *in isolation* and HMS updated during commit processing.
4. Table is created, and data inserted in parallel by Db2 Worker Nodes
(Lakehouse) Object Store
Db2
Big SQL
Watsonx.data

Exploring Datalake Tables

- Creating the storage access alias as administrator (or member of DASHDB_ENTERPRISE_ADMIN):

```
CALL SYSIBMADM.STORAGE_ACCESS_ALIAS.CATALOG('mybucket-alias', 'S3’,
    's3.us-east- 1.amazonaws.com', '****', '******,
    'mybucket', 'tables’,
    'R', 'DASHDB_ENTERPRISE_USER’)”
```

- Parameters:

```
Alias-name, vendor, endpoint, access & secret key, bucket name, path,
grantee-type (user / group / role), group or role
```

Exploring Datalake Tables

- Creating / deleting a regular Datalake table:

```
CREATE DATALAKE TABLE my_datalake_table(id INT, name VARCHAR(8))
   STORED AS PARQUET LOCATION 'DB2REMOTE://mybucket-alias//my_datalake_table’

DROP DATALAKE TABLE my_datalake_table DELETE DATA PURGE
```

- Optionally use external.table.purge=true to ensure data is deleted:

```
CREATE DATALAKE TABLE my_datalake_table(id INT, name VARCHAR(8))
  STORED AS PARQUET TBLPROPERTIES ('external.table.purge'='true’)
  LOCATION 'DB2REMOTE://mybucket-alias//my_datalake_table’

DROP DATALAKE TABLE my_datalake_table
```

Exploring Datalake Tables

- Creating an Apache Iceberg table:

```
CREATE DATALAKE TABLE my_datalake_table (id INT, name VARCHAR(8))
   STORED AS PARQUET
   STORED BY ICEBERG
   TBLPROPERTIES ('external.table.purge'='true’)
   LOCATION 'DB2REMOTE://mybucket-alias//my_datalake_table'
```

- Benefits of Apache Iceberg tables:
  - ACID table consistency
  - Update / delete (future)
  - Time travel snapshots (future)

Optimizing Query Performance of Datalake Tables

- Collect statistics with ANALYZE TABLE
  - Statistics help the Db2 Optimizer determine the most optimal access plan.
  - Auto-analyze can automatically run an ANALYZE TABLE statement on tables when it is determined to be necessary.

```
ANALYZE TABLE my_datalake_table
    COMPUTE STATISTICS FOR ALL COLUMNS
    TABLESAMPLE SYSTEM(10)
```

Optimizing Query Performance of Datalake Tables

- Create column-organized MQTs in a remote table space
  - Benefits from the multi-tier cache of remote table spaces
  - Allows for higher concurrency in query workload accessing Datalake table

```
CREATE TABLE my_datalake_table_MQT AS (SELECT * FROM from my_datalake_table)
    DATA INITIALLY DEFERRED REFRESH DEFERRED
    MAINTAINED BY USER DISABLE QUERY OPTIMIZATION
    ORGANIZE BY COLUMN IN OBJSTORESPACE1
```

  - Populate the MQT and gather statistics

```
INSERT INTO my_datalake_table_MQT SELECT * FROM my_datalake_table
CALL SYSPROC.ADMIN_CMD('runstats on table my_datalake_table_MQT on all columns')
```

  - Enable the MQT

```
ALTER MATERIALIZE QUERY my_datalake_table_MQT SET ENABLE QUERY OPTIMIZATION
```

Watsonx.data integration

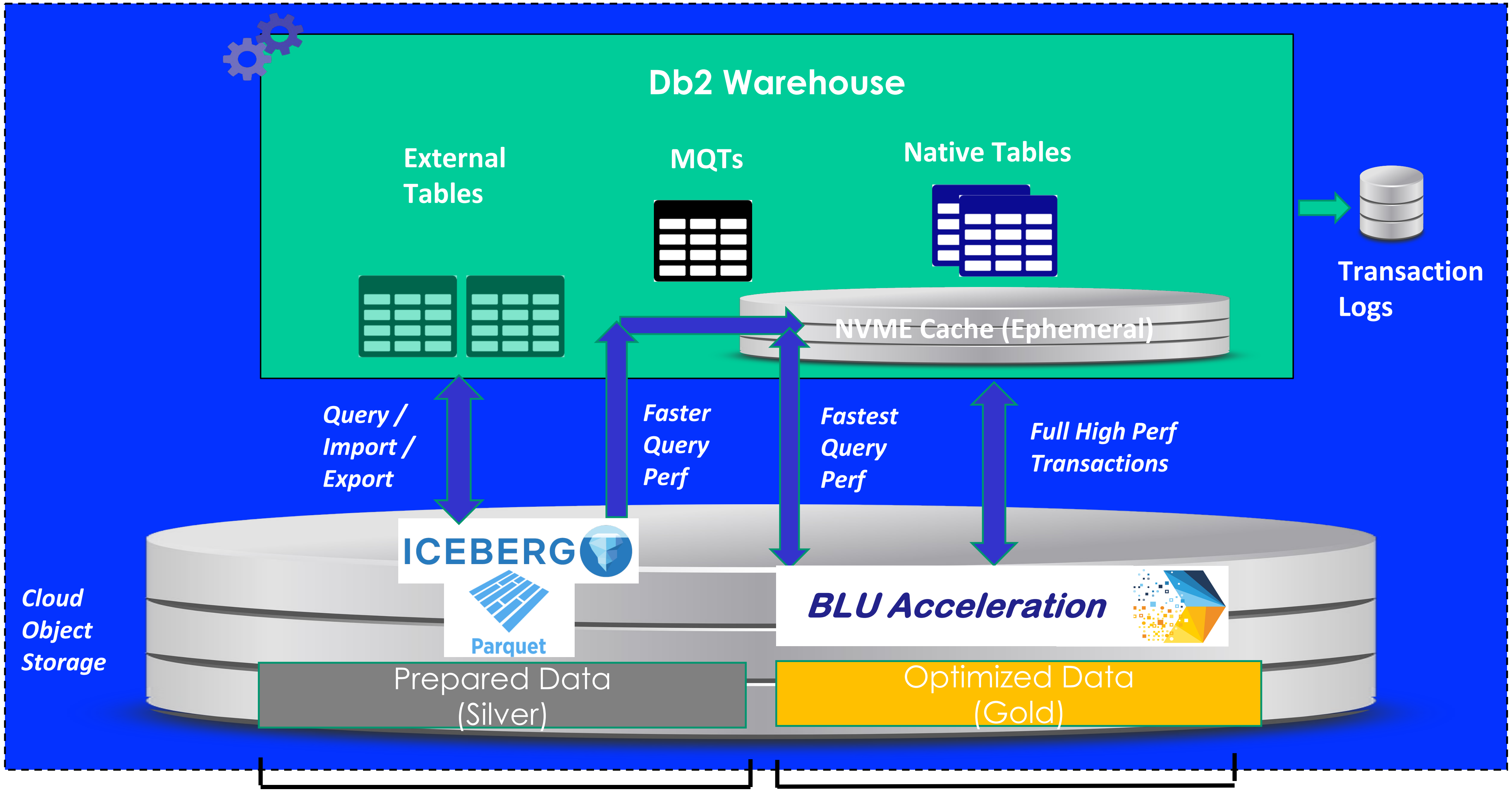
- With the watsonx.data integration
  - Import Apache Iceberg tables defined in watsonx.data into Db2 WH as a DATALAKE tables
  - Create a DATALAKE Iceberg table in both the Db2 and the watsonx.data Iceberg catalog with a single SQL statement.
- Connecting to the watsonx.data metastore

```
CALL REGISTER_EXT_METASTORE('watsonxdata’,
            'type=watsonx-data,uri=thrift://hmsauth1.fyre.ibm.com:9083', ?, ?)
CALL SET_EXT_METASTORE_PROPERTY('watsonxdata', 'use.SSL', 'true', ?, ?)
CALL SET_EXT_METASTORE_PROPERTY('watsonxdata', 'auth.mode', 'PLAIN', ?, ?)
CALL SET_EXT_METASTORE_PROPERTY('watsonxdata', 'auth.plain.credentials’,
                                'ibmlhapikey:<password>', ?, ?)
```

- Importing tables from watsonx.data

```
CALL EXTERNAL_CATALOG_SYNC('watsonxdata', 'iceberg_schema', '.*’,
                           'SKIP', 'CONTINUE', NULL)
```

Next Generation Db2 Warehouse - Summing Up



Open Format / Externally Managed (Interoperability)

Optimized Native Format (Performance)

Thank You!

Speaker: **Christian Garcia-Arellano**

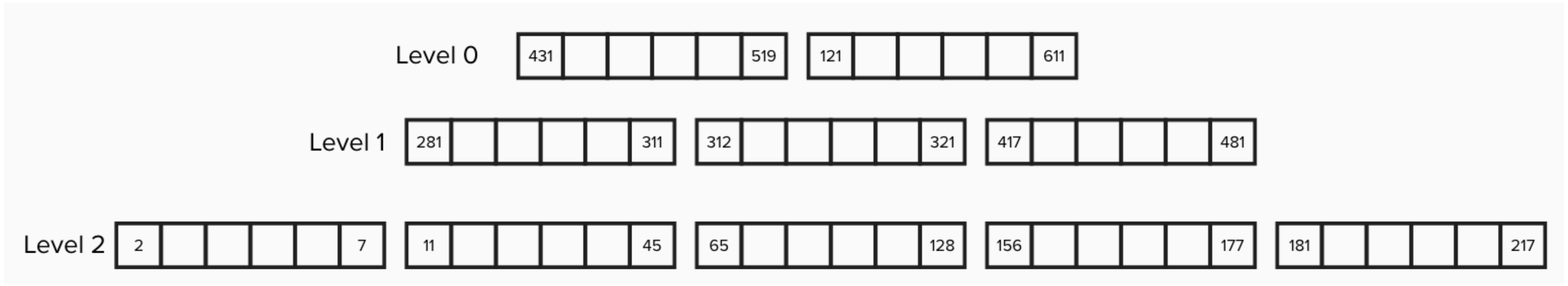
Company: **IBM**

Email Address: **cmgarcia@ca.ibm.com**

Session Code: **LUW-03**

Background On LSM trees

- 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.


Level 0
431
519
121
611
Level 1
281
311
312
321
417
481
Level 2
2
7
11
45
65
128
156
177
181
217