ADVANCED
DATABASE
SYSTEMS

Modern OLAP Databases

01

Andy Pavlo
CMU 15-721
Spring 2024

Carnegie Mellon University

TODAY'S AGENDA

Background

Architecture Overview

Query Execution

Project Discussion

BACKGROUND

Organizations use **on-line analytical processing** (OLAP) systems to extract new information from existing data sets.

Historically these workloads were run in a monolithic DBMSs that had all an organization's data in centralized managed storage…

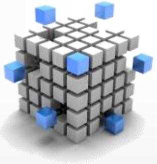
LAKEHOUSE: A NEW GENERATION OF OPEN PLATFORMS THAT UNIFY DATA WAREHOUSING AND ADVANCED ANALYTICS
CIDR 2021

1990s – DATA CUBES

DBMSs would maintain multi-dimensional arrays as pre-computed aggregations to speed up queries.

→ Periodically refreshed materialized views.
→ Administrator had to specify cubes ahead of time.

Data cubes were often introduced in existing operational DBMSs originally designed to operate on row-oriented data.



Microsoft SQL Server

ORACLE

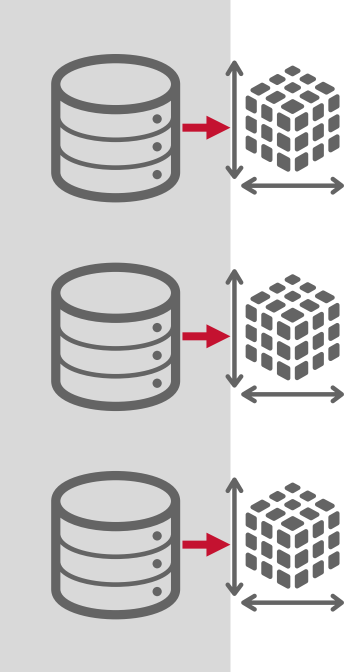
TERADATA

IBM DB2

SYBASE

Informix

1990s – DATA CUBES



OLTP Databases

Data cubes : simply pre-aggregation

```
SELECT product, region, cdate,
       SUM(amount) AS total_sales
  FROM sales
 GROUP BY CUBE (product, region, cdate);
```

2000s – DATA WAREHOUSES

Monolithic DBMSs designed to efficiently execute OLAP workloads using shared-nothing architectures and column-oriented data.

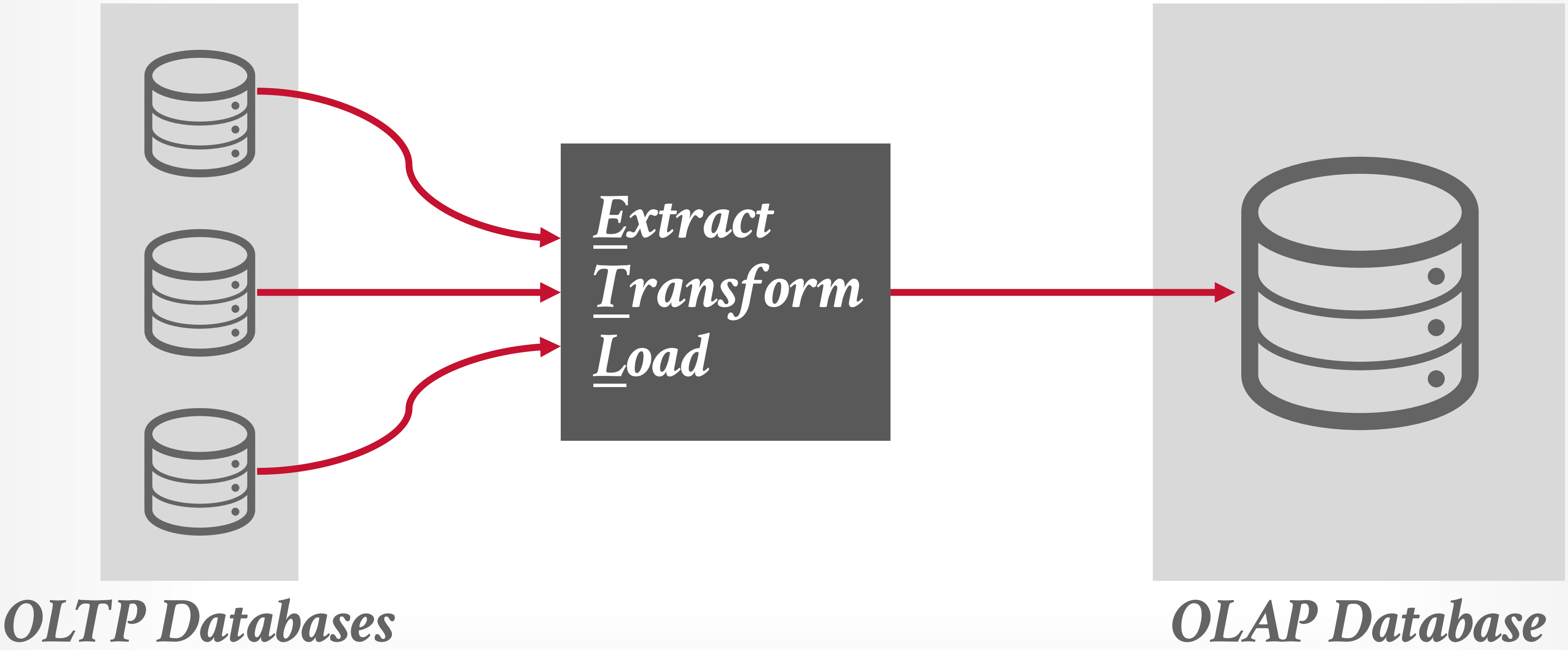
→ Many systems from this era started as forks of Postgres.

DBMS-managed storage using proprietary data encoding / formats.

NETEZZA PARACCEL monetdb

Greenplum DATAllegro VERTICA

2000s – DATA WAREHOUSES



2010s – SHARED-DISK ENGINES

Shared-disk DBMS architectures that relied on third-party distributed storage (object stores) instead of using a custom storage manager.

First generation of these systems managed data files themselves.

E.x: Snowflake data format + data store in S3

Newer systems allow external entities to add new data files to storage without enforcing schema (**lakehouse**).

Idea: Seperate compute and storage layer --> use shared-disk engines

HIVE

APACHE DRILL

druid

snowflake

cloudera IMPALA

pinot

presto

databricks

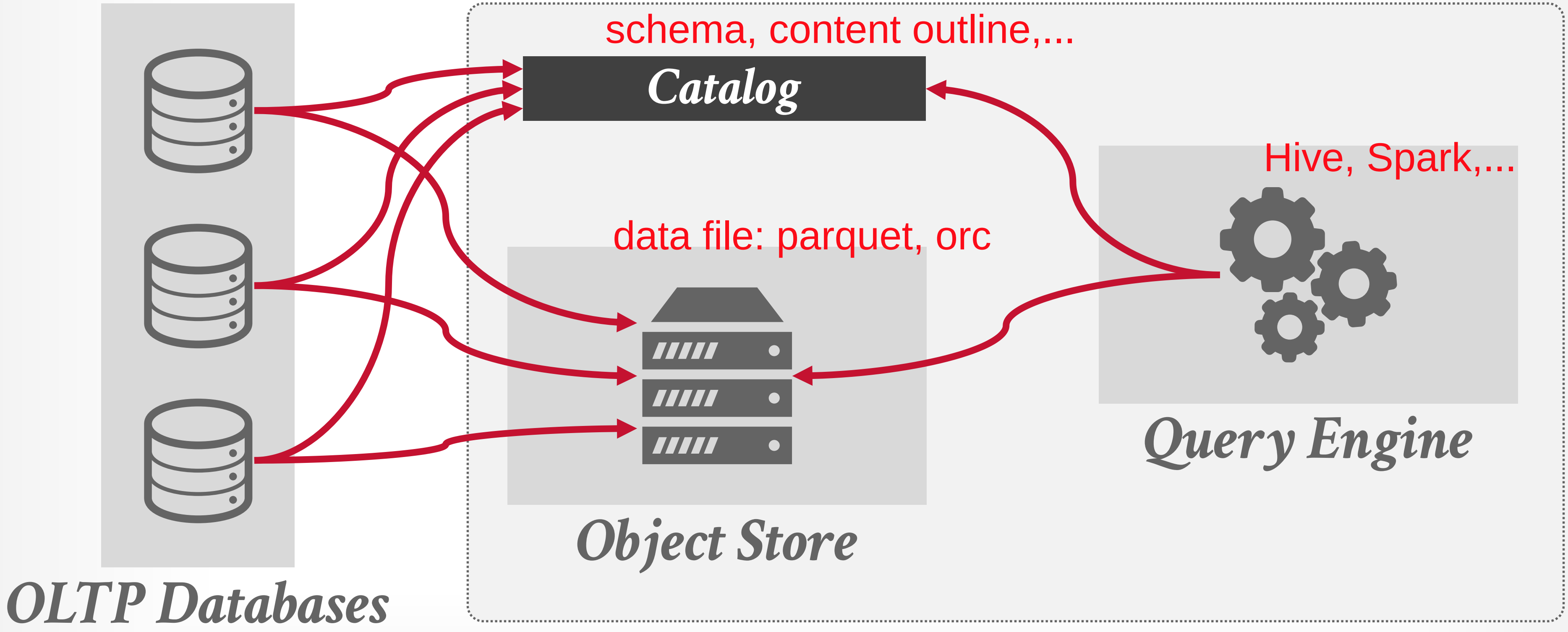
amazon REDSHIFT

Microsoft

APACHE Spark

trino

2010s – SHARED-DISK ENGINES



2020S – LAKEHOUSE SYSTEMS

Middleware for data lakes that adds support for better schema control / versioning with transactional CRUD operations.

→ Store changes in row-oriented log-structured files with indexes.

→ Periodically compact recently added data into read-only columnar files.

Schema evolution
(re-create table)

We will not be covering this aspect of these systems in this course.

2020S – LAKEHOUSE SYSTEMS

Observation #1: People want to execute more than just SQL on data.

Observation #2: Decoupling data storage from DBMS reduces ingest/egress barriers.

Observation #3: Most data is unstructured / semi-structured.

LAKEHOUSE: A NEW GENERATION OF OPEN PLATFORMS THAT UNIFY DATA WAREHOUSING AND ADVANCED ANALYTICS
CIDR 2021

OLAP DBMS COMPONENTS

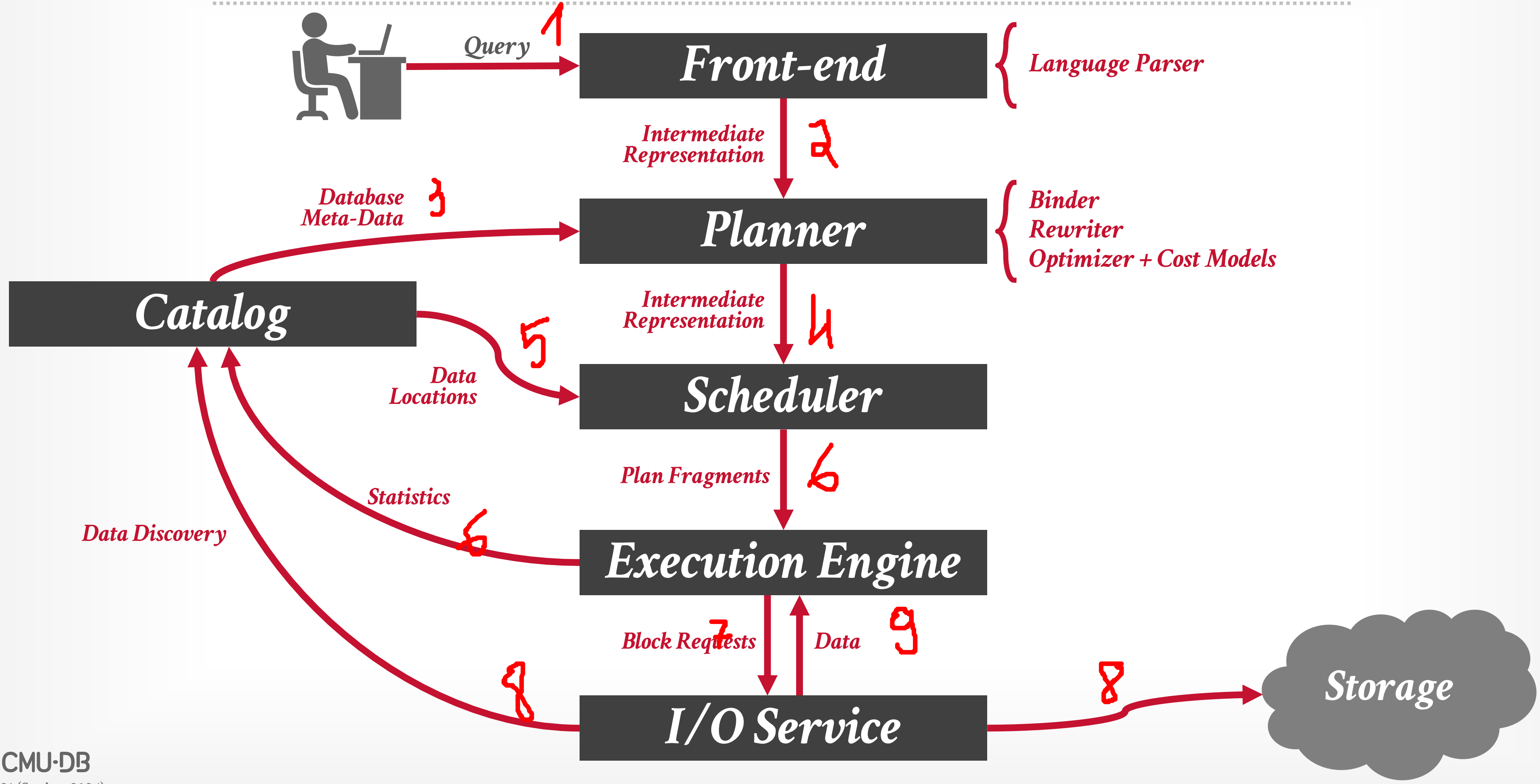
One recent trend of the last decade is the breakout of OLAP DBMS components into standalone services and libraries:

→ System Catalogs
→ Intermediate Representation
→ Query Optimizers
→ File Format / Access Libraries
→ Execution Engines / Fabrics

Lots of engineering challenges to make these components interoperable + performant.

THE COMPOSABLE DATA MANAGEMENT SYSTEM MANIFESTO
VLDB 2023

ARCHITECTURE OVERVIEW



DISTRIBUTED QUERY EXECUTION

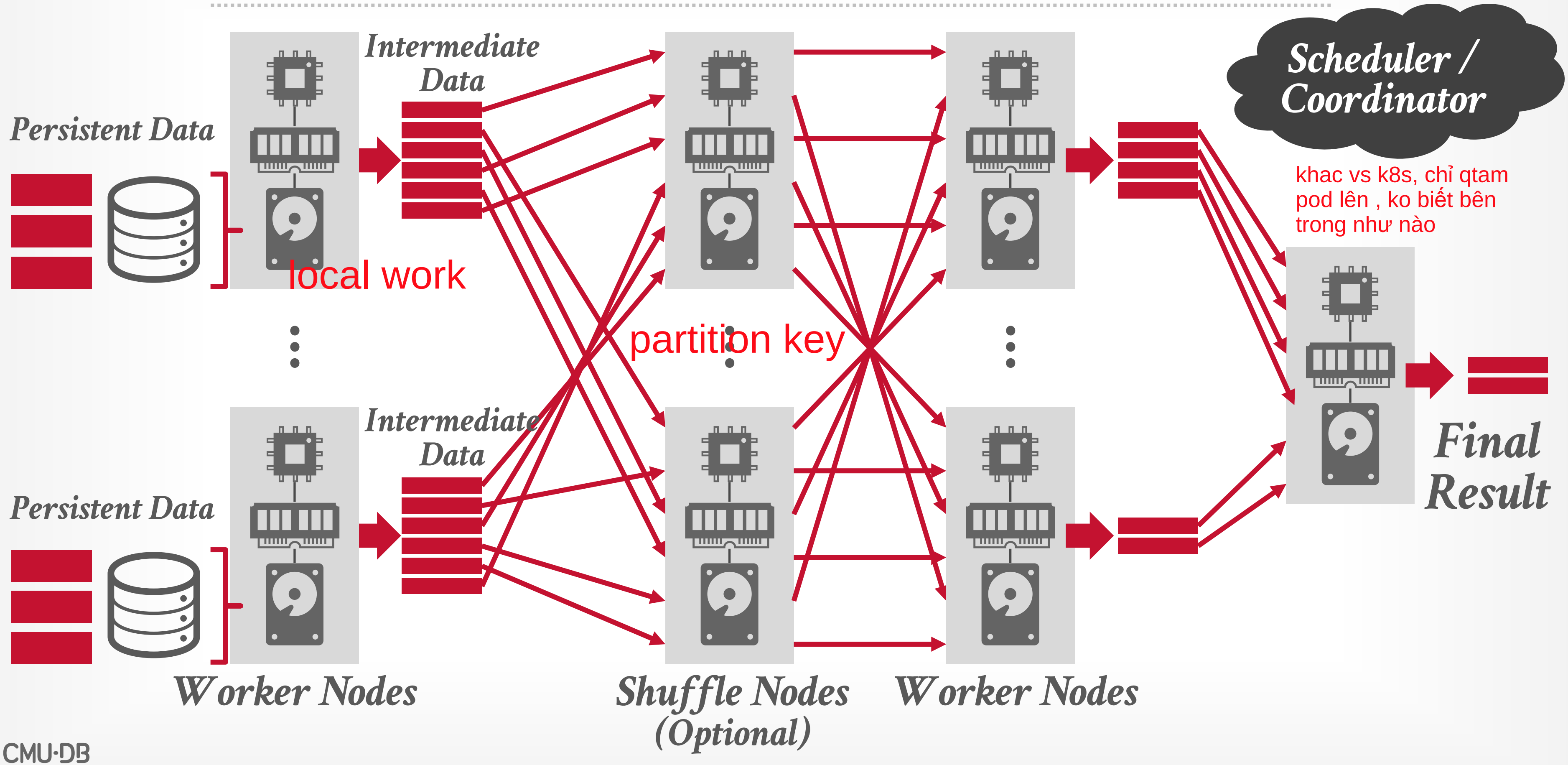
Executing an OLAP query in a distributed DBMS is roughly the same as on a single-node DBMS.

→ Query plan is a DAG of physical operators.

For each operator, the DBMS considers where input is coming from and where to send output.

→ Table Scans
→ Joins
→ Aggregations
→ Sorting

DISTRIBUTED QUERY EXECUTION



DATA CATEGORIES

Persistent Data:

→ The "source of record" for the database (e.g., tables).

→ Modern systems assume that these data files are immutable but can support updates by rewriting them.

Intermediate Data:

→ Short-lived artifacts produced by query operators during execution and then consumed by other operators.

→ The amount of intermediate data that a query generates has little to no correlation to amount of persistent data that it reads or the execution time.

BUILDING AN ELASTIC QUERY ENGINE ON DISAGGREGATED STORAGE
NSDI 2022

DISTRIBUTED SYSTEM ARCHITECTURE

A distributed DBMS's system architecture specifies the location of the database's persistent data files. This affects how nodes coordinate with each other and where they retrieve/store objects in the database.

Two approaches (not mutually exclusive):

→ **Push Query to Data**

→ **Pull Data to Query**

THE CASE FOR SHARED NOTHING
HPTS 1985

PUSH VS. PULL

Approach #1: Push Query to Data

→ Send the query (or a portion of it) to the node that contains the data.

→ Perform as much filtering and processing as possible where data resides before transmitting over network.

Approach #2: Pull Data to Query

→ Bring the data to the node that is executing a query that needs it for processing.

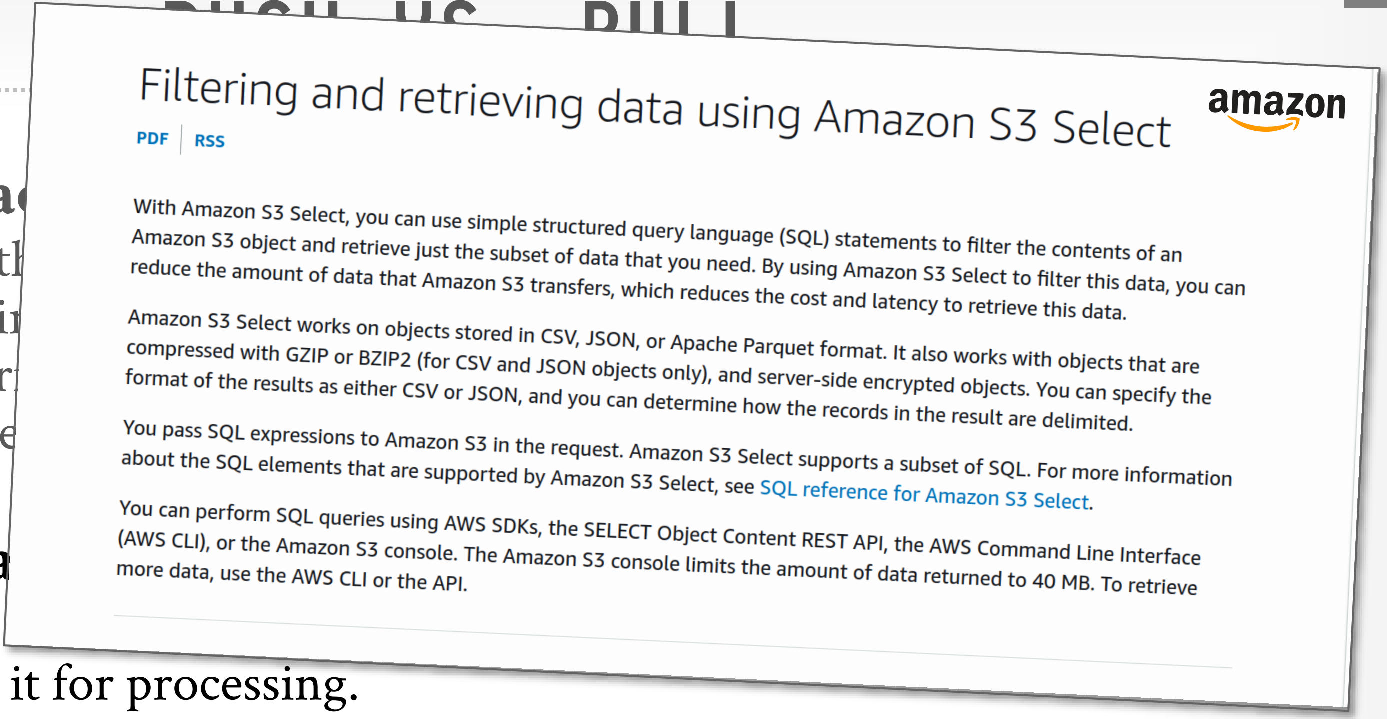
→ This is necessary when there is no compute resources available where persistent data files are located.

PUSH VS. PULL

Approa

→ Send th
contai

→ Perfor
data re



Filtering and retrieving data using Amazon S3 Select

PDF | RSS

With Amazon S3 Select, you can use simple structured query language (SQL) statements to filter the contents of an Amazon S3 object and retrieve just the subset of data that you need. By using Amazon S3 Select to filter this data, you can reduce the amount of data that Amazon S3 transfers, which reduces the cost and latency to retrieve this data.

Amazon S3 Select works on objects stored in CSV, JSON, or Apache Parquet format. It also works with objects that are compressed with GZIP or BZIP2 (for CSV and JSON objects only), and server-side encrypted objects. You can specify the format of the results as either CSV or JSON, and you can determine how the records in the result are delimited.

You pass SQL expressions to Amazon S3 in the request. Amazon S3 Select supports a subset of SQL. For more information about the SQL elements that are supported by Amazon S3 Select, see SQL reference for Amazon S3 Select.

You can perform SQL queries using AWS SDKs, the SELECT Object Content REST API, the AWS Command Line Interface (AWS CLI), or the Amazon S3 console. The Amazon S3 console limits the amount of data returned to 40 MB. To retrieve more data, use the AWS CLI or the API.

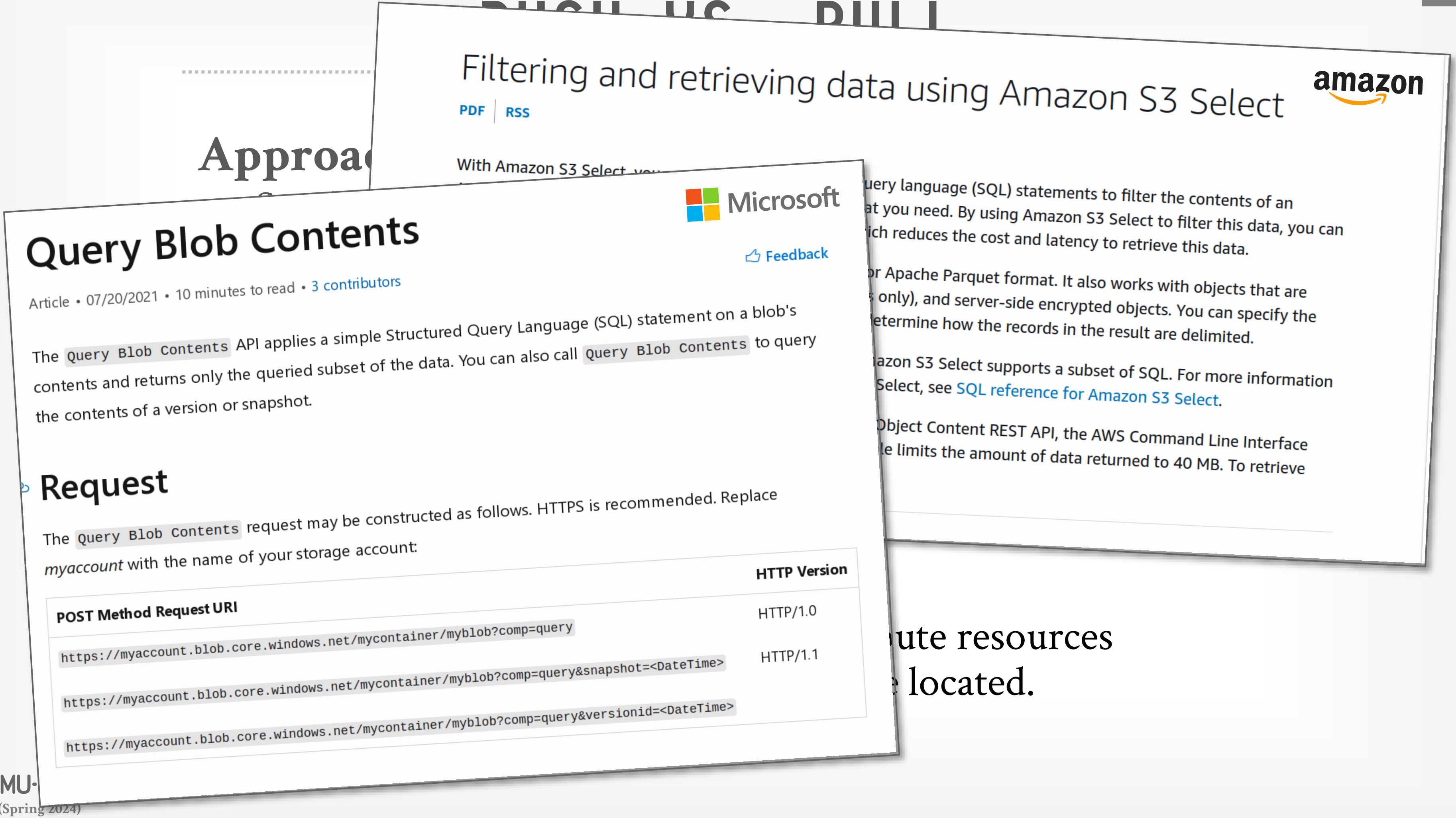
Approa

→ Bring
needs it for processing.

→ T
his is necessary when there is no compute resources available where persistent data files are located.

PUSH VS. PULL

Approac



Filtering and retrieving data using Amazon S3 Select

With Amazon S3 Select, you … uery language (SQL) statements to filter the contents of an … at you need. By using Amazon S3 Select to filter this data, you can … ich reduces the cost and latency to retrieve this data.

… or Apache Parquet format. It also works with objects that are … only), and server-side encrypted objects. You can specify the … etermine how the records in the result are delimited.

… azon S3 Select supports a subset of SQL. For more information … Select, see SQL reference for Amazon S3 Select.

… Object Content REST API, the AWS Command Line Interface … le limits the amount of data returned to 40 MB. To retrieve

Query Blob Contents

Article • 07/20/2021 • 10 minutes to read • 3 contributors

The `Query Blob Contents` API applies a simple Structured Query Language (SQL) statement on a blob's contents and returns only the queried subset of the data. You can also call `Query Blob Contents` to query the contents of a version or snapshot.

Request

The `Query Blob Contents` request may be constructed as follows. HTTPS is recommended. Replace *myaccount* with the name of your storage account:

POST Method Request URI	HTTP Version

`https://myaccount.blob.core.windows.net/mycontainer/myblob?comp=query`
`https://myaccount.blob.core.windows.net/mycontainer/myblob?comp=query&snapshot=<DateTime>`
`https://myaccount.blob.core.windows.net/mycontainer/myblob?comp=query&versionid=<DateTime>`
	HTTP/1.0
	HTTP/1.1

ute resources

located.

SHARED-NOTHING

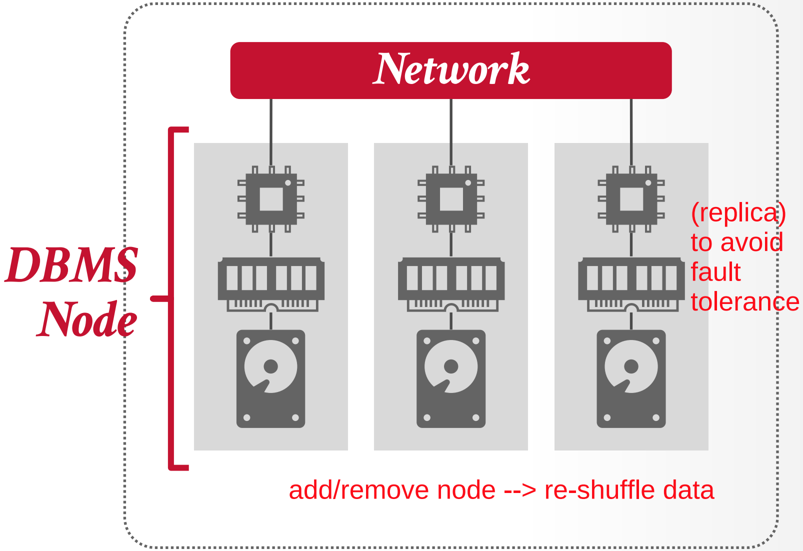
Each DBMS instance has its own CPU, memory, locally-attached disk.

→ Nodes only communicate with each other via network.

Database is partitioned into disjoint subsets across nodes.

→ Adding a new node requires physically moving data between nodes.

Since data is local, the DBMS can access it via POSIX API.



(replica) to avoid fault tolerance

add/remove node --> re-shuffle data

object store and NFS (network file system) provide same interface except that object store provide more

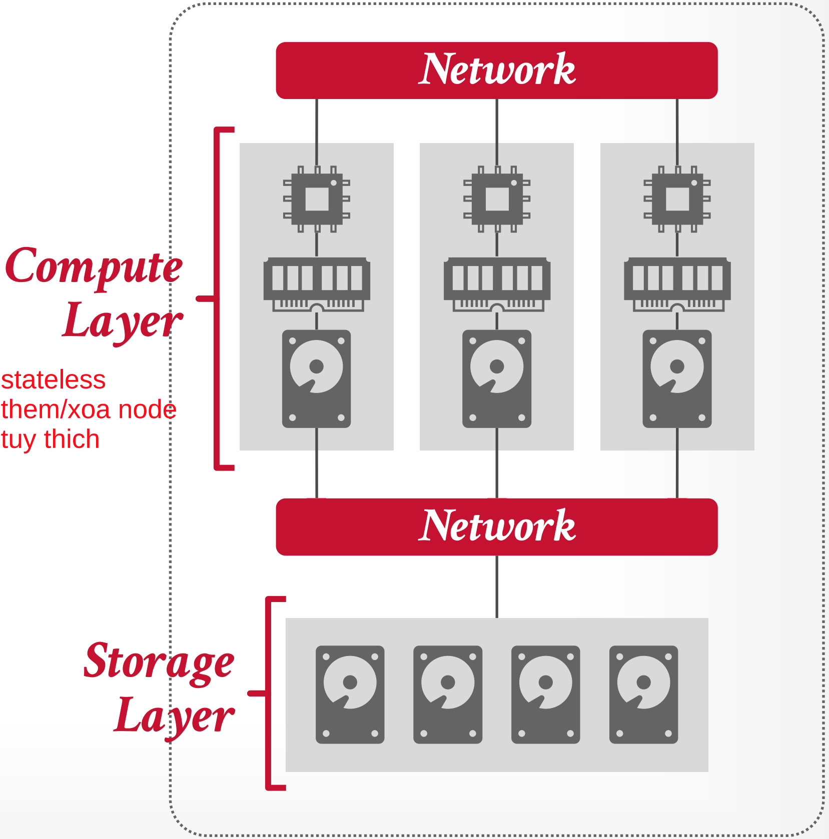
f.write | f.read
do not know the data is actually be stored

SHARED-DISK

Each node accesses a single logical disk via an interconnect, but also have their own private memory and tmp ephemeral storage.

→ Must send messages between nodes to learn about their current state.

Instead of a POSIX API, the DBMS accesses disk using a userspace API.



stateless
them/xoa node
tuy thich

SYSTEM ARCHITECTURE

Choice #1: Shared-Nothing:

→ Harder to scale capacity due to data movement.
→ Potentially better performance & efficiency.
→ Apply filters where the data resides before transferring.

Choice #2: Shared-Disk:

→ Scale compute layer independently from the storage layer.
→ Easy to shutdown idle compute layer resources.
→ May need to pull uncached persistent data from storage layer to compute layer before applying filters.

SHARED-DISK IMPLEMENTATIONS

Traditionally the storage layer in shared-disk DBMSs were dedicated on-prem NAS.

→ Example: Oracle Exadata

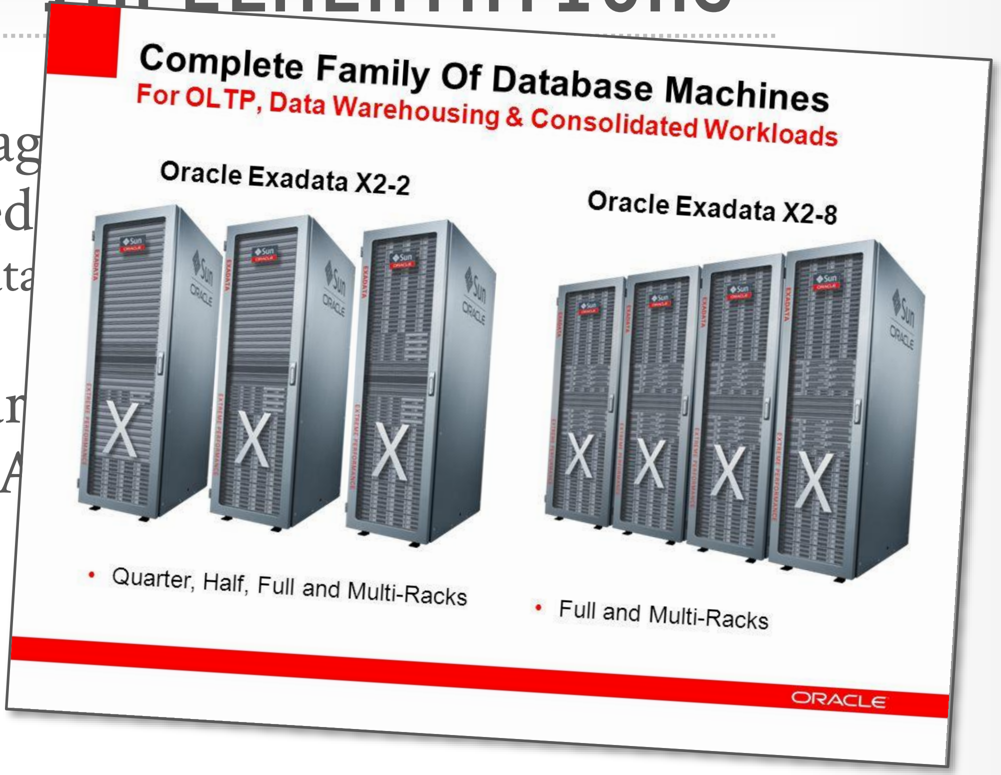
Cloud **object stores** are now the prevailing storage target for modern OLAP DBMSs because they are "infinitely" scalable.

→ Examples: Amazon S3, Azure Blob, Google Cloud Storage

SHARED-DISK IMPLEMENTATIONS

Traditionally the storag
DBMSs were dedicated
→ Example: Oracle Exadata

Cloud **object stores** ar
target for modern OLA
"infinitely" scalable.
→ Examples: Amazon S3,



Complete Family Of Database Machines

For OLTP, Data Warehousing & Consolidated Workloads

Oracle Exadata X2-2

- Quarter, Half, Full and Multi-Racks

Oracle Exadata X2-8

- Full and Multi-Racks

OBJECT STORES

Partition the database's tables (persistent data) into large, immutable files stored in an object store.

→ All attributes for a tuple are stored in the same file in a columnar layout (PAX).

→ Header (or footer) contains meta-data about columnar offsets, compression schemes, indexes, and zone maps.

The DBMS retrieves a block's header to determine what byte ranges it needs to retrieve (if any).

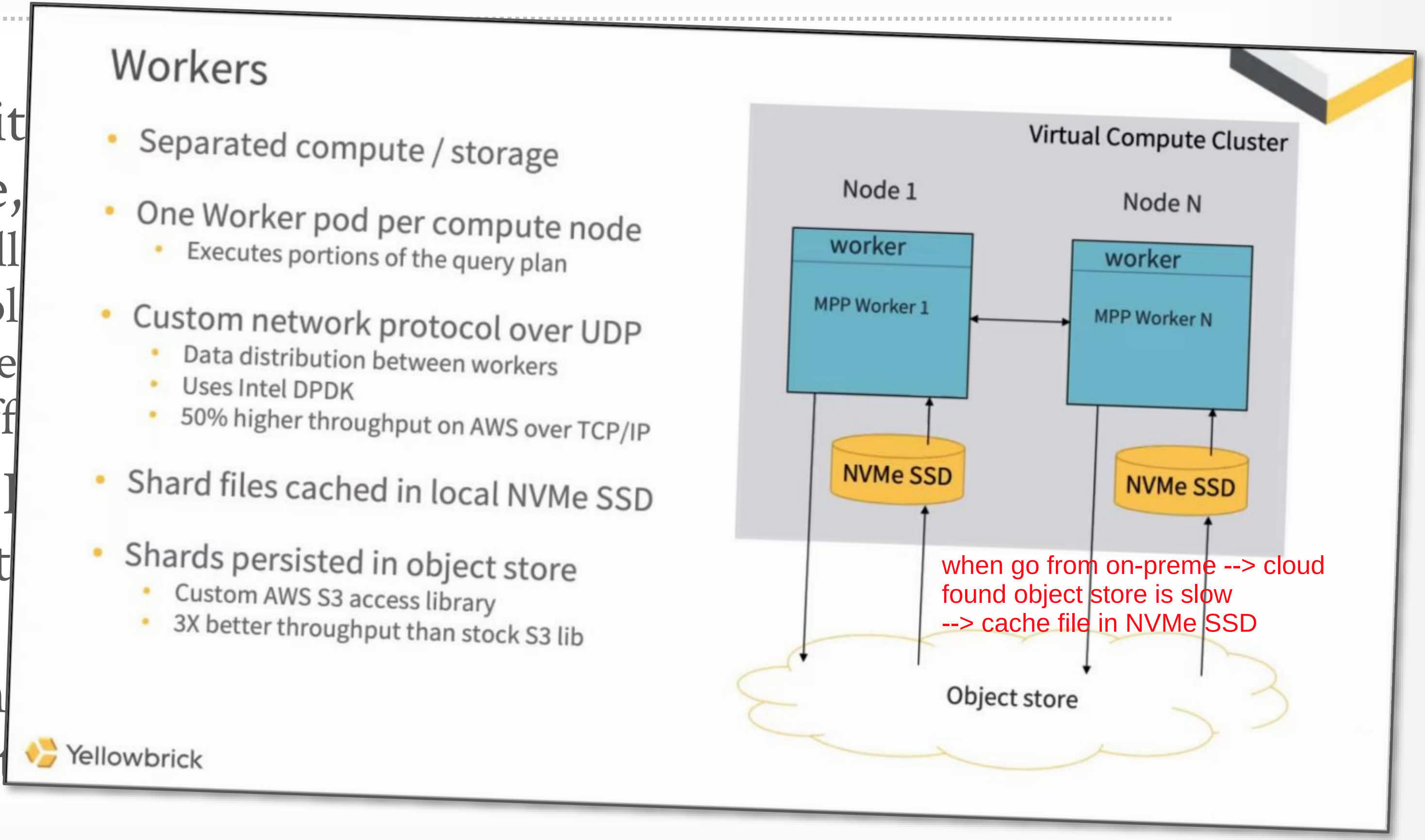
Each cloud vendor provides their own proprietary API to access data (**PUT**, **GET**, **DELETE**).

OBJECT STORES

Partit
large,
→ All
col
→ He
off

The I
what

Each
API



Workers

- Separated compute / storage
- One Worker pod per compute node
  - Executes portions of the query plan
- Custom network protocol over UDP
  - Data distribution between workers
  - Uses Intel DPDK
  - 50% higher throughput on AWS over TCP/IP
- Shard files cached in local NVMe SSD
- Shards persisted in object store
  - Custom AWS S3 access library
  - 3X better throughput than stock S3 lib

when go from on-preme --> cloud
found object store is slow
--> cache file in NVMe SSD

Yellowbrick

CONCLUSION

Today was about understanding the high-level context of what modern OLAP DBMSs look like.

→ Fundamentally these new DBMSs are not different than previous distributed/parallel DBMSs except for the prevalence of a cloud-based object store for shared disk.

Our focus for the rest of the semester will be about state-of-the-art implementations of these systems' components.

NEXT CLASS

Storage Models

Data Representation

Encoding

Compression