Hello!

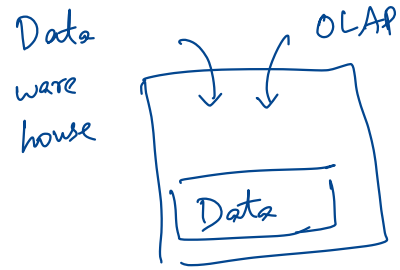
CS 744: SNOWFLAKE

Shivaram Venkataraman

Fall 2022

ADMINISTRIVIA

- Project Proposals – Due 11am tomorrow! → Highly recommend submitting today!
- Midterm on Thursday! Seating layout?
  - ↳ open book, open notes
  - ↳ Send email for extensions



→ Optimizer, Schema

SparkSQL/Scope: Given a query how do you run it efficiently?

→ OLAP

Snowflake: How do you build an elastic data warehouse?

↓

adjust amount of resources used at runtime

CLOUD COMPUTING STACK

AWS / Azure

PyTorch

SCOPE

Machine Learning

SQL

Analytics as a Service!

Snowflake

MapReduce

Spark

Computational Engines

Elastic MapReduce

EMR

EC2

Virtual Machines

GFS

Scalable Storage Systems

S3, EBS

Storage As a service

SNOWFLAKE: GOALS

Software-as-a-Service → Users do not install or deploy anything.
Connecting on your browser

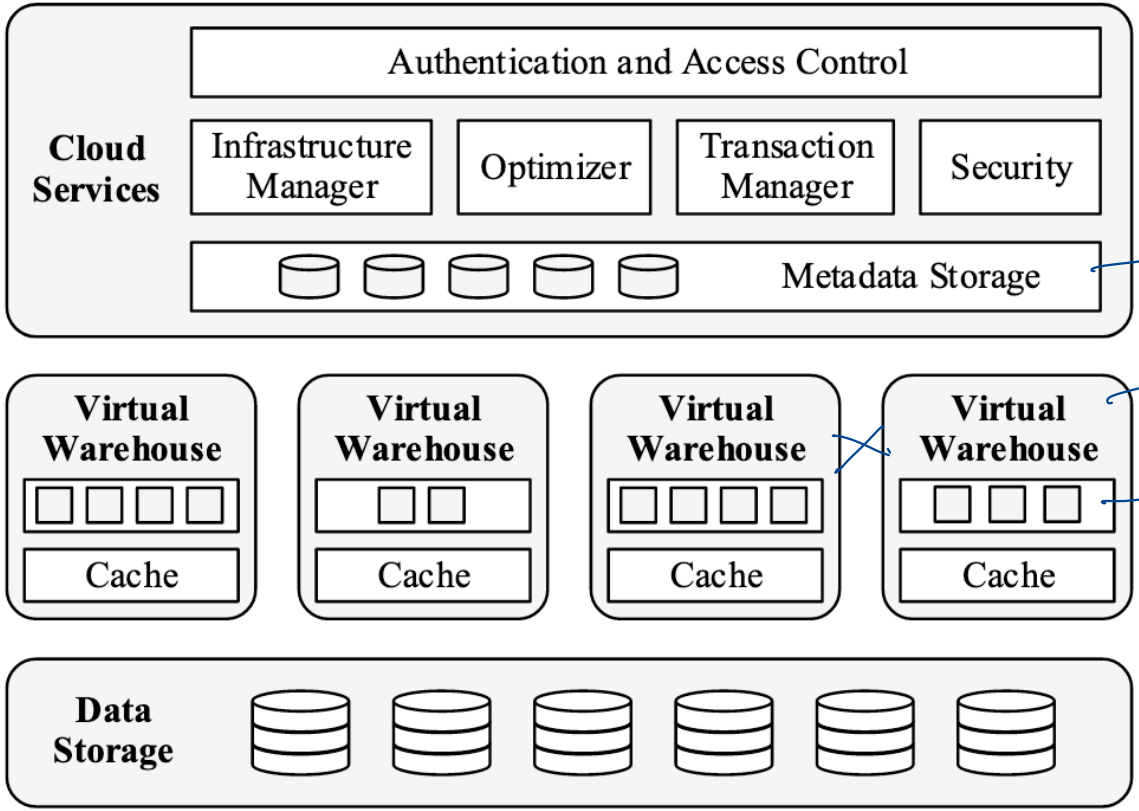
Elastic → Use resources as required.
→ Use CPU / mem when you have work
Shut down otherwise

Highly Available
→ Fault Tolerant

Semi-Structured Data
→ Real data arrives as JSON / XML formats. May not have a schema

SNOWFLAKE DESIGN

Control plane (GFS Master Mesos ...)



Metadata Storage separate module

Data plane

Compute

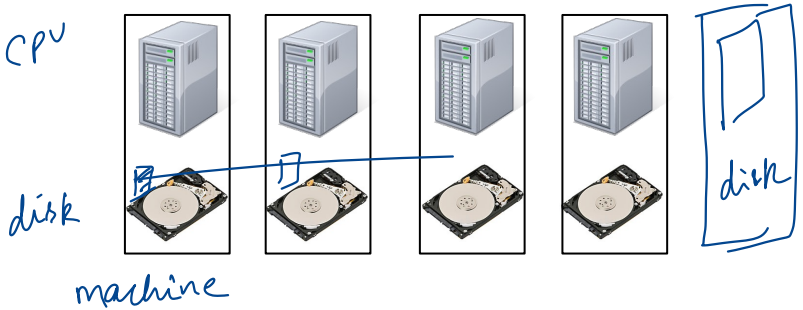
separate for user workers

Isolation

Shared across VWs Storage

STORAGE VS COMPUTE

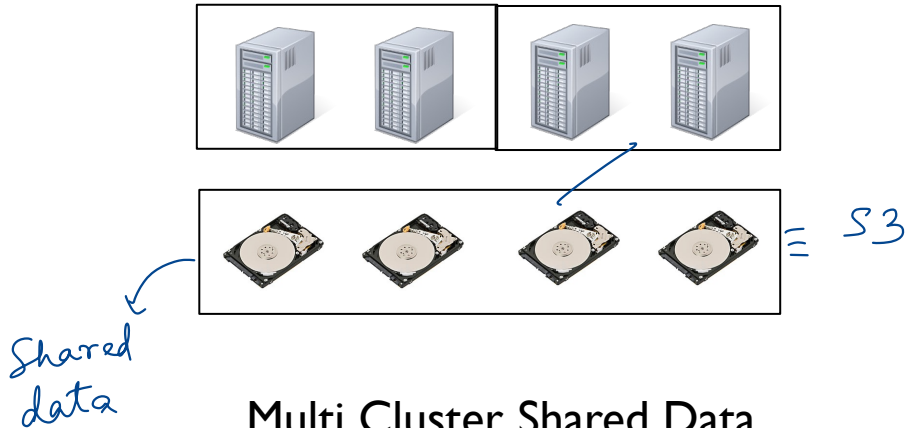
- ties together CPU and disk
  cannot independently scale them
- load balancing
- Data locality ↔ rack / machine locality



Shared Nothing

- Scale independently
- lose locality!

Every storage access is over the network



Multi Cluster, Shared Data

When is shared nothing

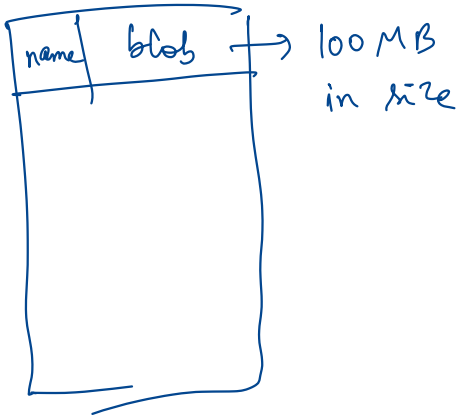
- Locality is very important
- Data isolation
  - ↳ Storage: Policies on data types
  - ↳ Access policies

Shared data

- Network usage is low
- Irregular arrival of queries but data is stored permanently
- Replication across data centers make utilization even more important

STORAGE: HYBRID COLUMNAR

S3
KV store



Alice	32
Bob	22
Eve	24
Victor	27

Access only part of file for a query

Compress much better

Within a file / partition use columnar storage

Chunk size = 2 rows

Alice,32,Bob,22

Eve,24,Victor,27

Row-oriented

Alice, Bob, 32,22

Eve,Victor,24,27

Hybrid Columnar

VIRTUAL WAREHOUSES

Elasticity, Isolation → VW : collection of worker VMs

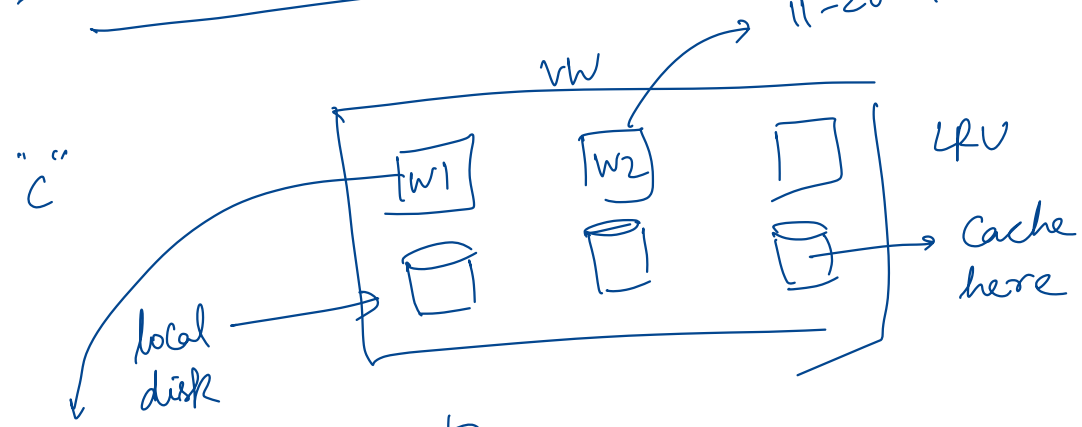
- Each user can launch a new VW. Diff set of machines
- Increase / Decrease / Destroy without affecting DB.

Local caching, Stragglers

```
select * from users
where user.name startsWith "C"
```

Read dominated

Work stealing to mitigate stragglers

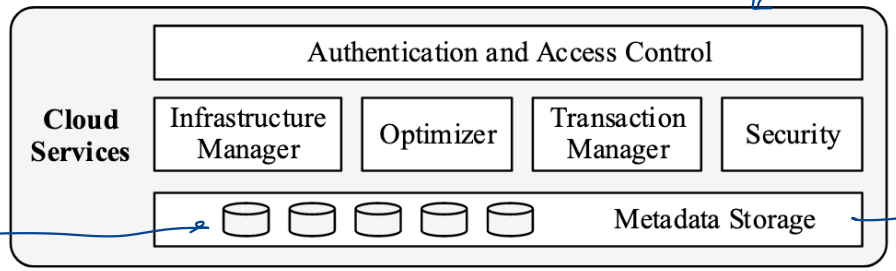


CLOUD SERVICES

query entry

Redis / MemCached.

Fast KV store



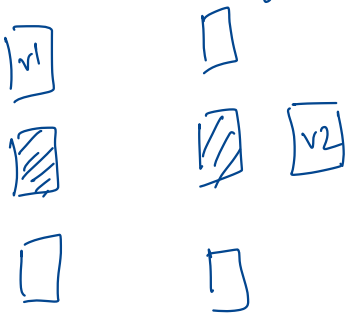
Which S3 files belong to which table

Concurrency Control → MVCC

Tables have versions

→ every insert / update creates new version

old



→ metadata update

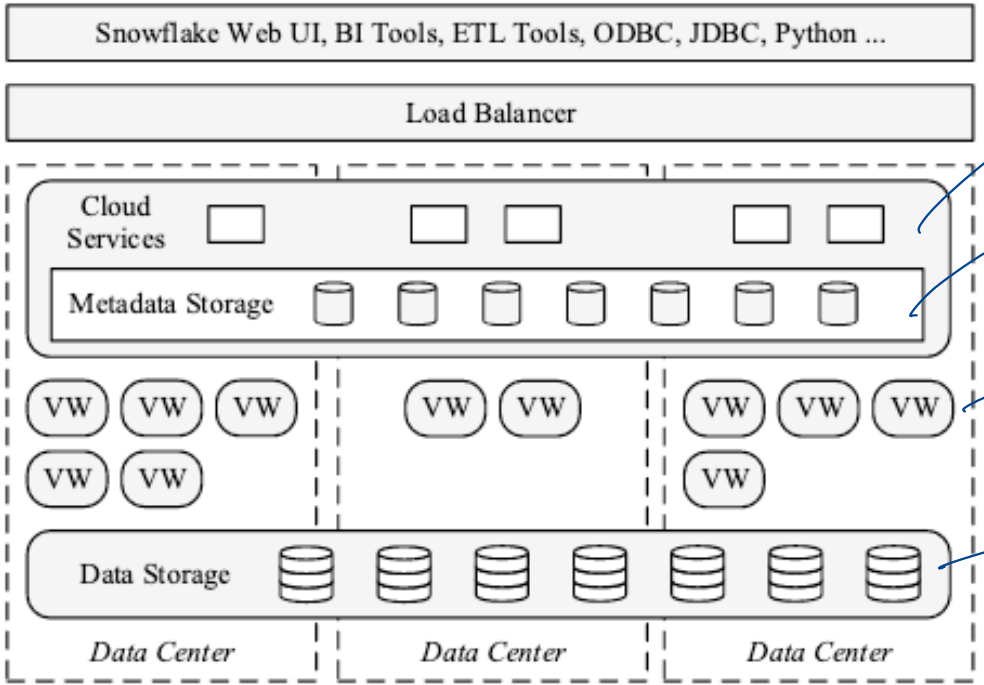
Pruning

↳ limit number of files read during query execution

→ min, max for every col. in the header of every file

FAULT TOLERANCE

general: design approach to separate stateful parts of your stack



stateless service

replicated across DCs.

on failure, retry the query.

replicated

SEMI STRUCTURED DATA

JSON data

value

```
{
    first_name: “john”,
     last_name: “doe”,
     order_id: “1234”,
}
{
   first_name: “bucky”,
   last_name: “badger”,
   order_id: “52342”,
   order_date: “3/3/2020”,
}
```

→ Integer type

→ is not present in all the records.

Extraction, Flattening operations

select value . order_id from table

Infer types, Pruning

- Type inference on every file
- min/max values for pruning

TIME TRAVEL?

```
SELECT * FROM my_table AT(TIMESTAMP =>
  'Mon, 01 May 2015 16:20:00 -0700'::timestamp);
SELECT * FROM my_table AT(OFFSET => -60*5); -- 5 min ago
SELECT * FROM my_table BEFORE(STATEMENT =>
  '8e5d0ca9-005e-44e6-b858-a8f5b37c5726');
```

Multiple versions of table (MVCC)

Undo accidental deletes

Cheap to clone / snapshot a table

SUMMARY, TAKEAWAYS

Snowflake

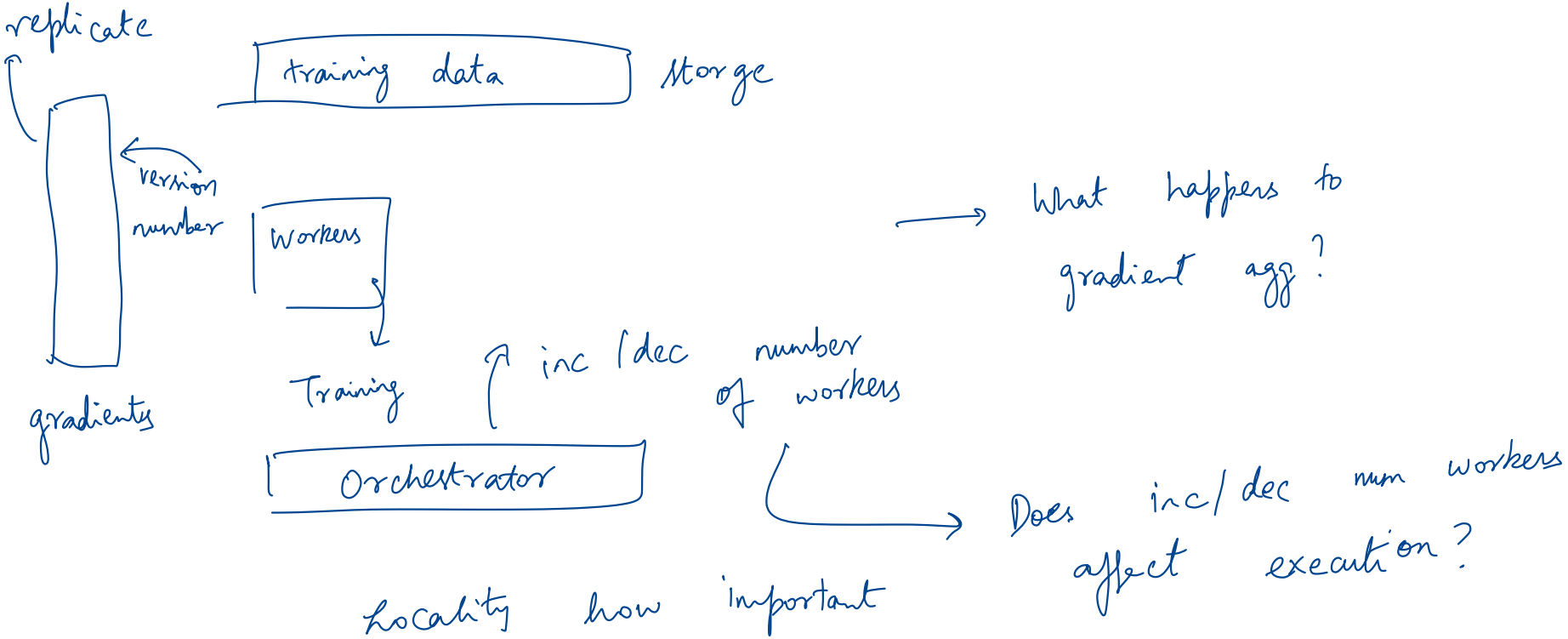
- Cloud computing → Elastic data warehouse
- Key idea: Separation of compute and storage!

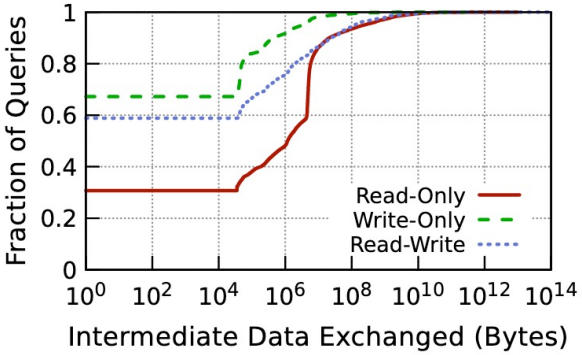
- Hybrid columnar storage format
- Elastic compute with virtual warehouses
- Pruning, semi-structured optimizations, fault tolerant

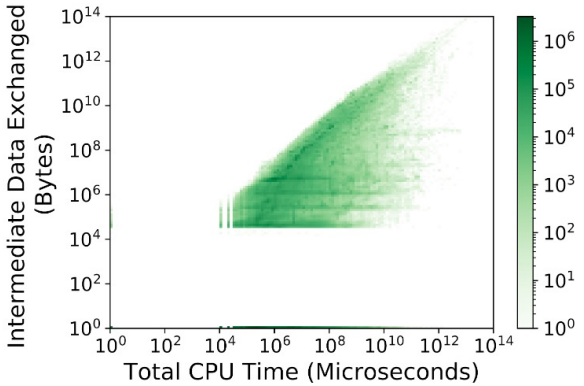
DISCUSSION

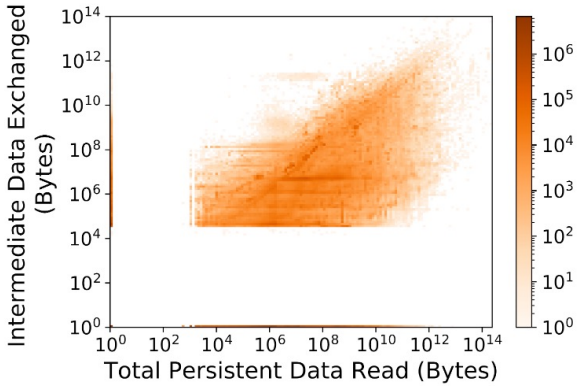
https://forms.gle/Trfe62jEp1ZUocJk6

We see how Snowflake leads to the design of an elastic data warehouse. If we were to similarly design an Elastic PyTorch for training how would the design look? What are some design trade-offs compared to existing PyTorch?




Fraction of Queries
1
0.8
0.6
0.4
0.2
0
10^0 10^2 10^4 10^6 10^8 10^{10} 10^{12} 10^{14}
Intermediate Data Exchanged (Bytes)
Read-Only
Write-Only
Read-Write


Intermediate Data Exchanged (Bytes)
10^{14} 10^{12} 10^{10} 10^8 10^6 10^4 10^2 10^0
10^0 10^2 10^4 10^6 10^8 10^{10} 10^{12} 10^{14}
Total CPU Time (Microseconds)
10^6 10^5 10^4 10^3 10^2 10^1 10^0


Intermediate Data Exchanged (Bytes)
10^{14} 10^{12} 10^{10} 10^8 10^6 10^4 10^2 10^0
10^0 10^2 10^4 10^6 10^8 10^{10} 10^{12} 10^{14}
Total Persistent Data Read (Bytes)
10^6 10^5 10^4 10^3 10^2 10^1 10^0

NEXT STEPS

Next class: Midterm!