



*Good morning!*

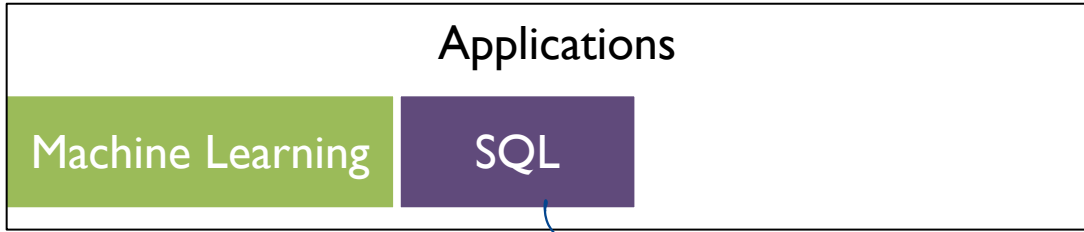
# CS 744: SNOWFLAKE

Shivaram Venkataraman

Spring 2024

# ADMINISTRIVIA

- Midterm on Thursday! Seating layout?
- Project proposal feedback  9:30 am on Thursday
-  Canvas
- Office hours moved to Wed 1-2 PM CS 7367



→ large scale data analysis

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

→ compiler  
parser  
query optimization

Snowflake: How do you build an elastic data warehouse?

→ cloud

as-a-Service



Snowflake

Machine Learning

SQL

Enterprise /  
Fixed cluster

Scope

# CLOUD COMPUTING STACK

EMR  
EC2

Computational Engines

MR /  
Spark

virtual  
machine

Amazon  
S3

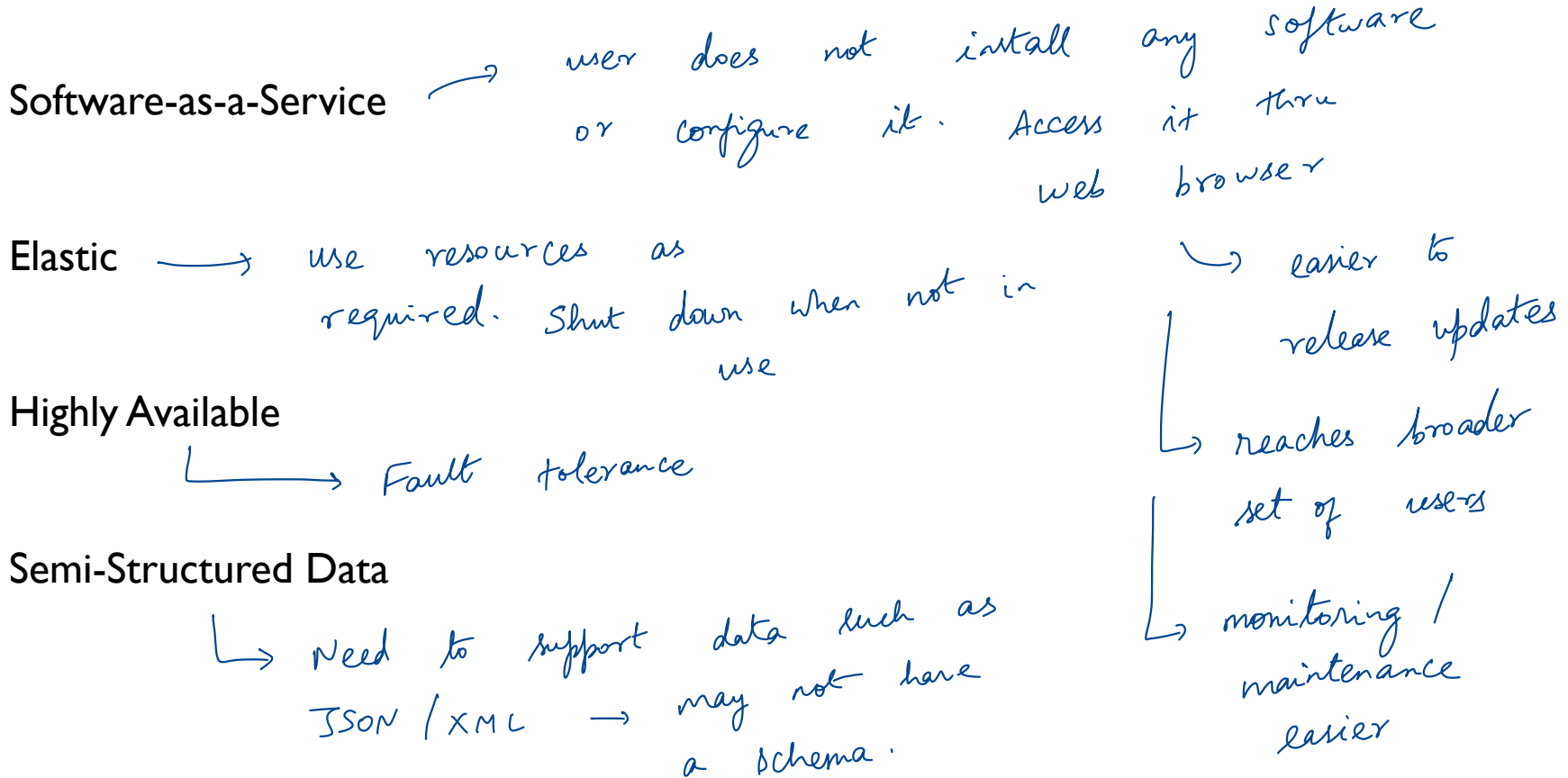
Scalable Storage Systems

GFS

---

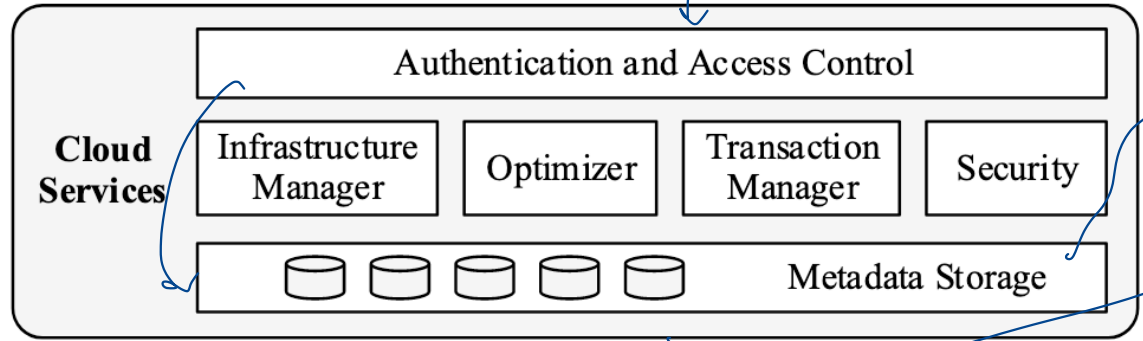
Amazon AWS  
GCP

# SNOWFLAKE: GOALS



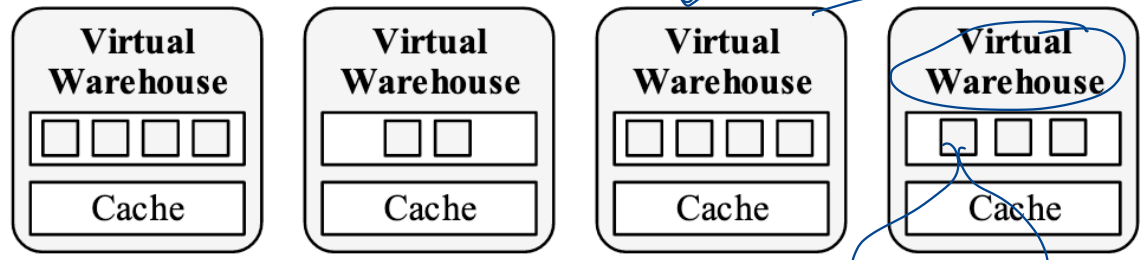
# SNOWFLAKE DESIGN

Control plane  
→ Coordinator  
or  
Master



metadata about  
schema  
tables & S3  
files etc.

Data plane  
→ data  
storage /  
query  
processing



elastic  
each client  
launch a  
set of VMs



shared S3  
storage  
global accessible

query

Cloud Services

Infrastructure Manager

Optimizer

Transaction Manager

Security

Metadata Storage

Virtual Warehouse

Virtual Warehouse

Virtual Warehouse

Virtual Warehouse

Cache

Cache

Cache

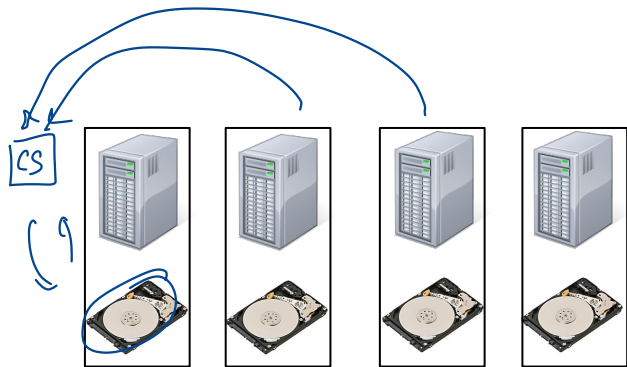
Cache

Data Storage

# STORAGE VS COMPUTE

- Shared data
  - consistency?
- locality → lose this with shared data → Networks lot faster

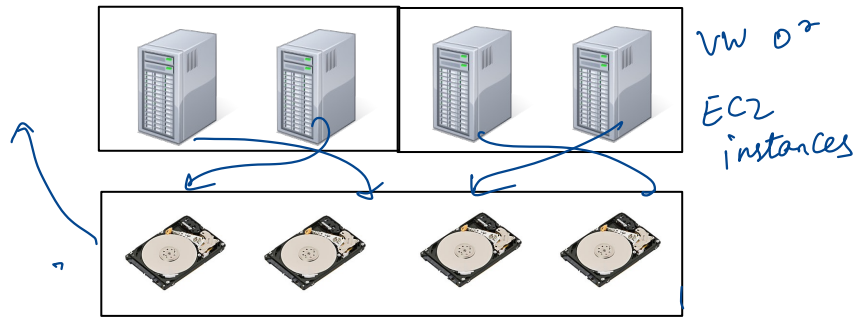
- Independence between compute and storage
  - Compute machine goes down then it doesn't affect storage
- scale independently



Chunk server

Shared Nothing

ensure access control



Multi Cluster, Shared Data

# STORAGE: HYBRID COLUMNAR

Compression  
and query performance

Columns

Alice	32
Bob	22
Eve	24
Victor	27

Alice, Bob, Eve, Victor ←

32, 22, 24, 27 ←

partition the table  
into files/chunks

first row / second row

Blobs

Alice, 32, Bob, 22

Eve, 24, Victor, 27

Row-oriented

minimize  
partitions/  
data read to  
answer a  
query

Alice, Bob, 32, 22

Eve, Victor, 24, 27

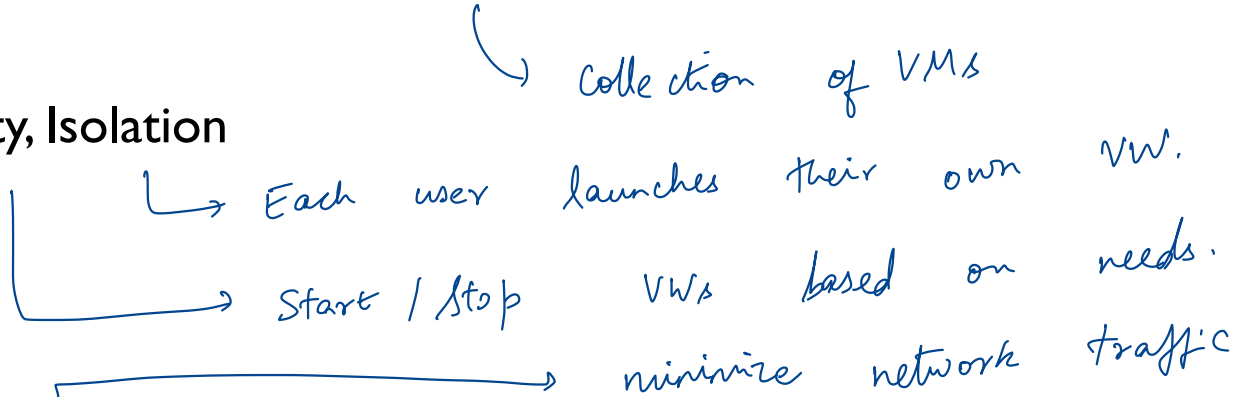
Hybrid Columnar

columnar  
storage within  
a partitions



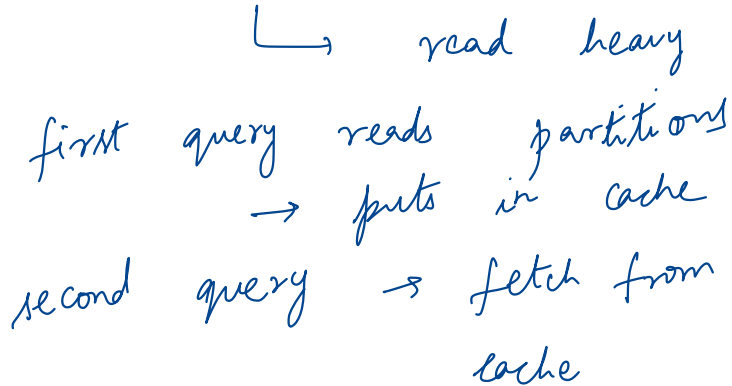
# VIRTUAL WAREHOUSES

## Elasticity, Isolation

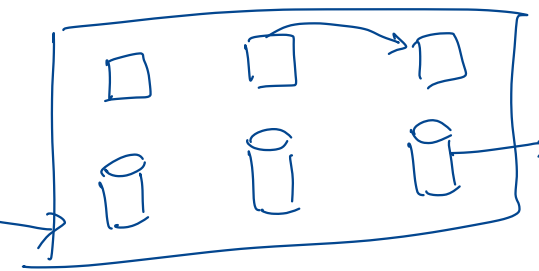


Work stealing  
↳ steal some tasks from other VMs

## Local caching, Stragglers



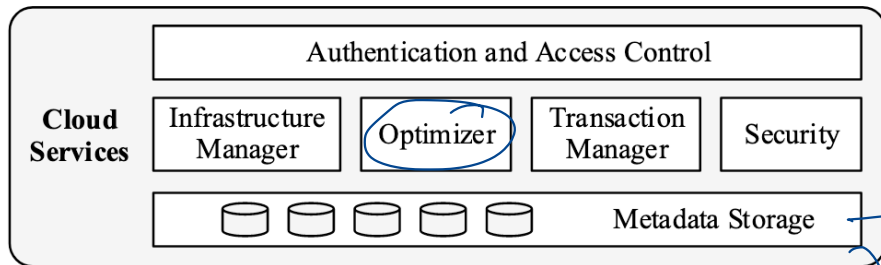
local disk as a cache



intermediate data

# CLOUD SERVICES

externalize  
state into  
a storage  
system



query

```
select *  
from users  
where age < 18
```

Redis /  
Memcached  
KV store

## Concurrency Control

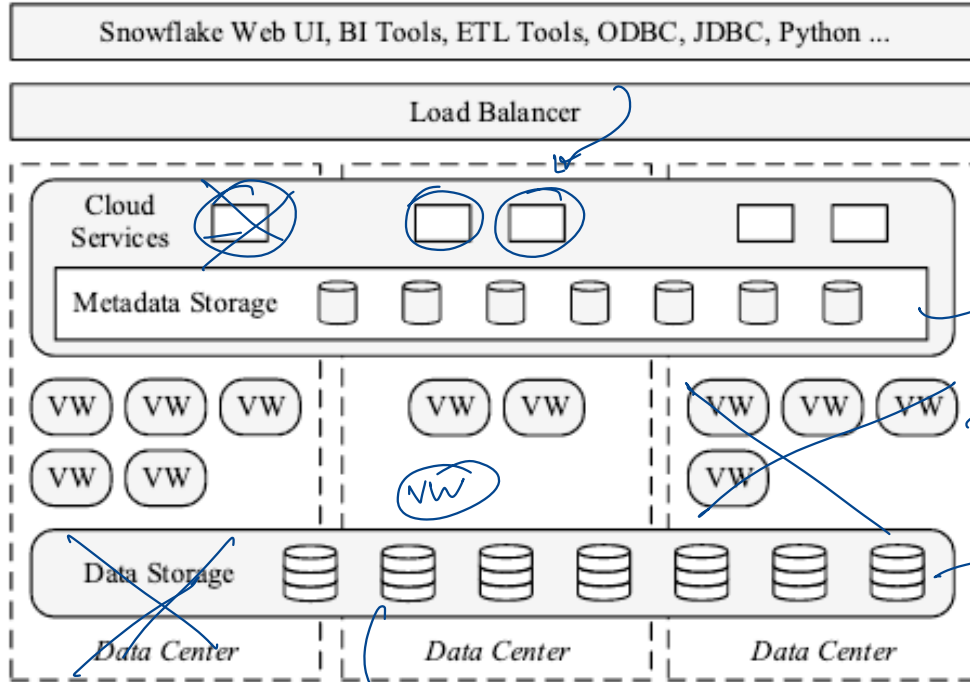
- multiple versions of a partition
- query tied to particular version

## Pruning

- skip some partitions not relevant to query
- min/max per column metadata

which tables  
are in  
which S3  
files

# FAULT TOLERANCE



separate stateful and stateless

replicated across DCs

Always On

stateless

On Demand

Infinite

stateful replicate across DCs

overhead

# SEMI STRUCTURED DATA

JSON → tables

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

String

integer

not present in prev entry

Extraction, Flattening operations

*select value.order\_id from  
table*

Infer types, 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



*delete a partition?  
prev version is available*

Cheap to clone / snapshot a table

# SUMMARY, TAKEAWAYS

## Snowflake

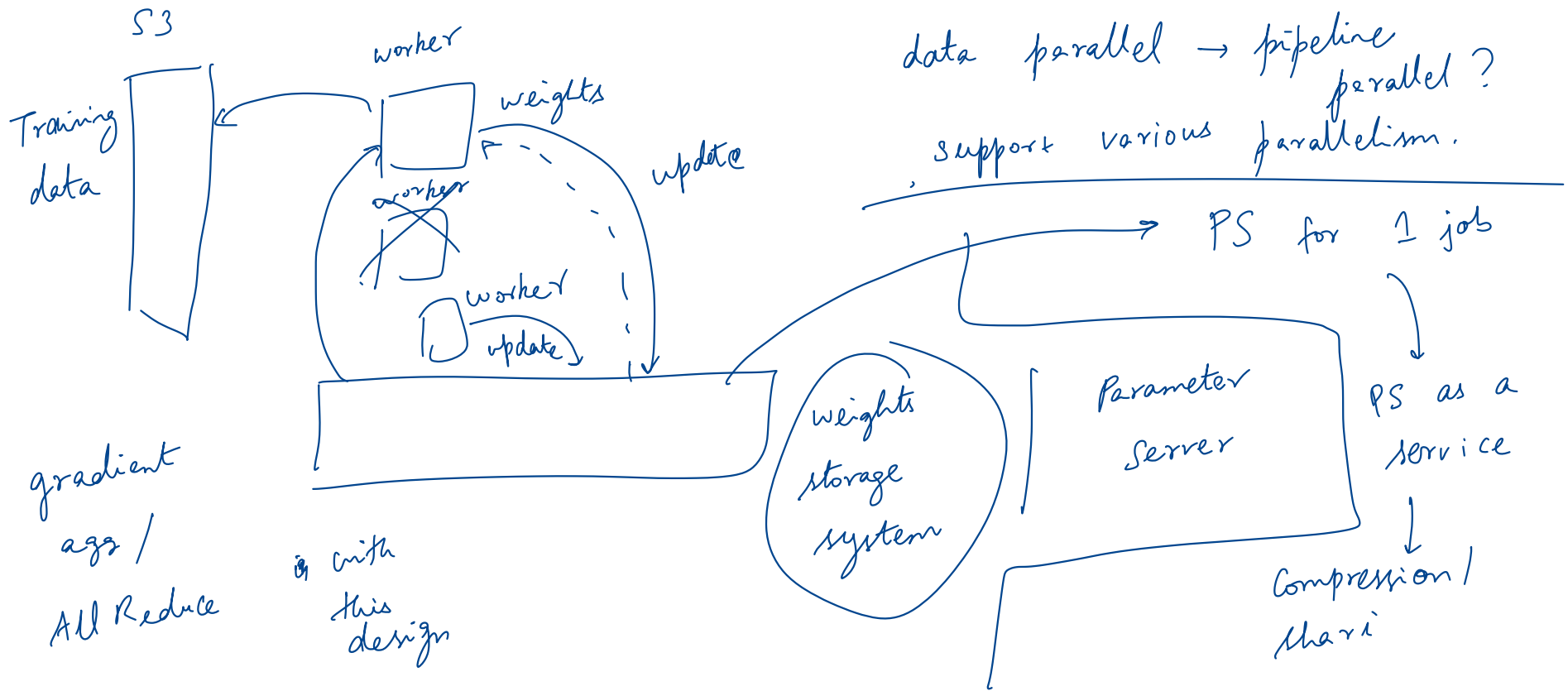
- Cloud computing → Elastic data warehouse
- Key idea: Separation of compute and storage!
  
- Hybrid columnar storage format
- Elastic compute with virtual warehouses
- Pruning, semi-structured optimizations, fault tolerant



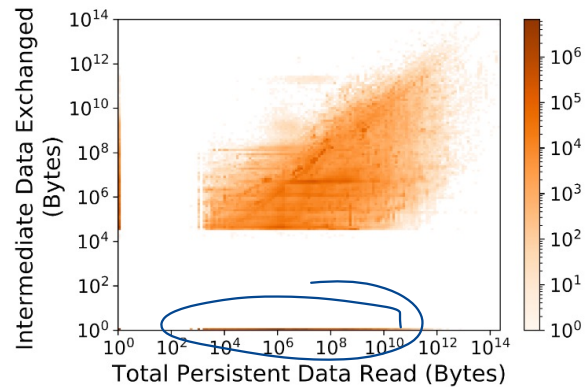
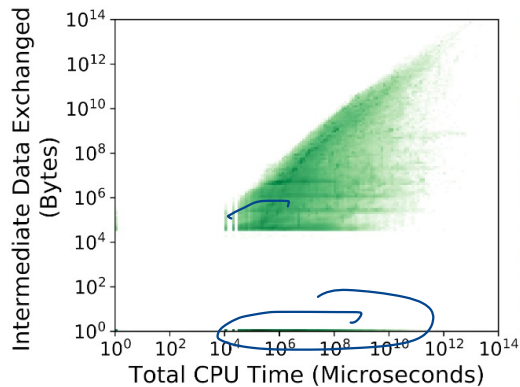
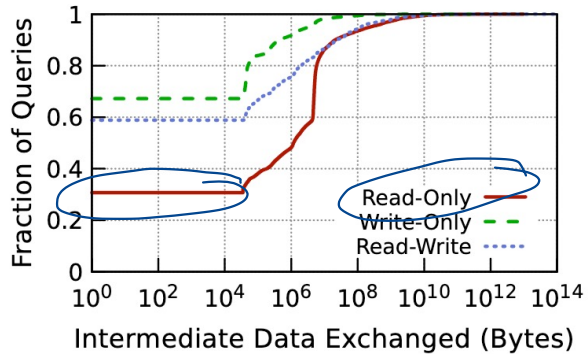
# DISCUSSION

<https://forms.gle/Not7Pz4t9LwntSct7>

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?







jobs which use CPU / persistent data  
 scan table & filter?

# NEXT STEPS

Next class: Midterm!