Welcome!

# CS 744: SNOWFLAKE

Shivaram Venkataraman

Fall 2020

# ADMINISTRIVIA

Prepare
→ Reading / lecture
Prepare notes
→ Try to solve previous year exams

Open ended
↳ Pros / Cons for various designs
→ Compare various systems

- Assignment 1 grades out! → Canvas. Contact Saurabh
- Assignment 2 by mid-week →
- Midterm this week! → Thursday. The exam will be posted on Canvas

- Project Proposal Peer review

↓

Assigning one other project for you to review

BBCollaborate → for clarifications

PDF upload to Canvas → Practice this !!

You can also type it out

# AEFIS FEEDBACK

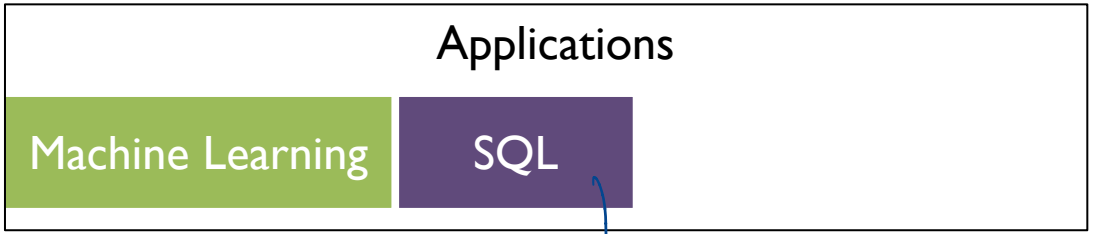
4 / 18 responses!

How has your experience been reading papers?

Are the lectures useful for learning?

How are the discussion groups? Did you get to know students in the class? Would it help to have the same group each time?

Anything else we could improve for the second half?




→ SCOPE : Relational API

Analytics → Language Integration

DAG of operators run on execution engines like Spark

Snowflake

↳ elastic or suited for the cloud

# CLOUD COMPUTING STACK

no management overhead

Software as a Service

PyTorch etc.

Machine Learning

SQL

ScoPG

AWS SageMaker etc.

MPI Spark

Computational Engines

EMR → Elastic MR

EC2 → VMs

GFS

Scalable Storage Systems

service external users

Amazon S3

storage system

enterprise Google data center

# SNOWFLAKE: GOALS

Software-as-a-Service → No need to download anything
Browser based & On-off as you need it

Elastic → Use as many as required
Not any more / change when necessary easily

Highly Available → Fault tolerance

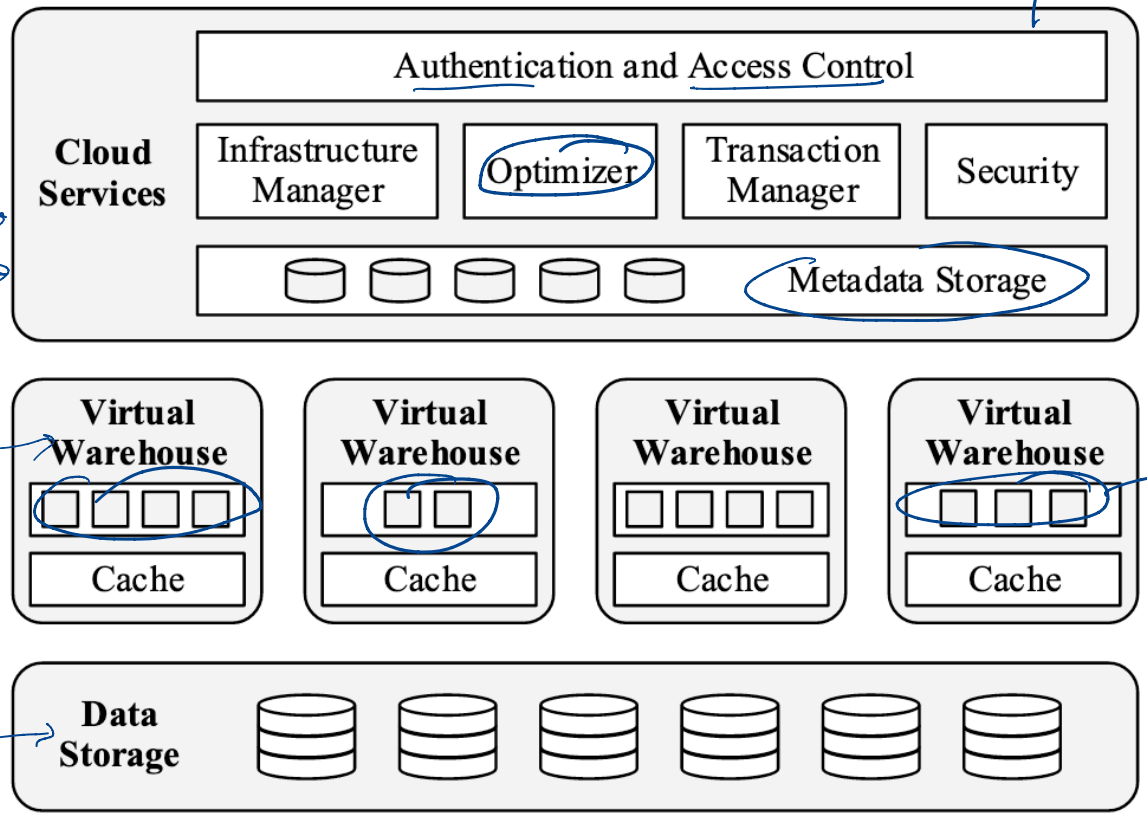
Semi-Structured Data → Data that might not fit a strict schema

JSON,
XML
⋮

vs.

Relational Databases strict schema

# SNOWFLAKE DESIGN




SaaS

Elastic

HA

Semi Structured

"Brains"

↓

SaaS

↳ multiple tenants / users

replicate to HA to get

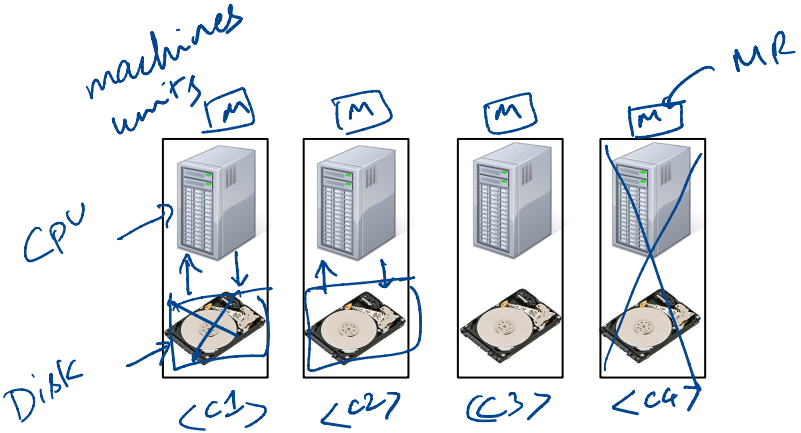
Elasticity where each VW can have diff num machines

# STORAGE VS COMPUTE

Share this data
↳ across jobs
↳ failures

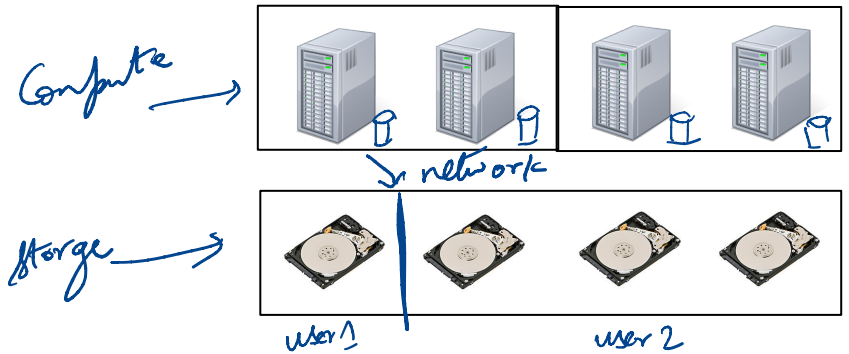
Coupling CPU, disk
↳ If I need 4 CPUs
⇒ 4 disks
↳ Utilization could be low

Separate out compute & storage
→ No locality!
→ Scale compute / storage independently




Shared Nothing

Each of these is a VM




Mega Byte level

Multi Cluster, Shared Data

Snowflake:
No database indices!

# STORAGE: HYBRID COLUMNAR

Table

| Name | Age |
|---|---|
| Alice | 32 |
| Bob | 22 |
| Eve | 24 |
| Victor | 27 |

rows

horizontally partitioned
each partition becomes a "file" in S3
↳ immutable

C1: Alice,32,Bob,22

C2: Eve,24,Victor,27

Row-oriented

Name
byte 0
byte 23

Age
byte 24
byte 32

name

Age

Alice, Bob, 32,22

Eve, Victor, 24,27

Hybrid Columnar

A lot of queries only touch a few columns
⇒ range get
Avoid reading entire file!!

# VIRTUAL WAREHOUSES

one user cannot slow down another user

Elasticity, Isolation → EC2 virtual machines launched for a particular user

- Every user can be diff in size
- Number of VMs, Type of VMs → num, cores → small XXL

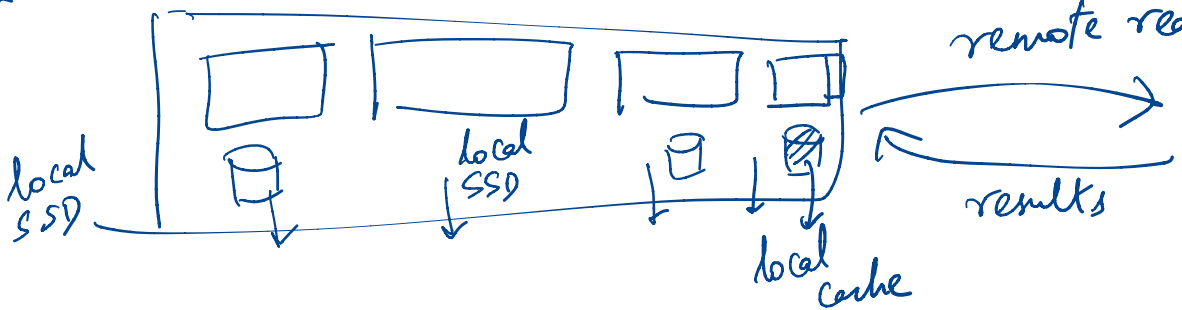
Only use them when running a query!

Local caching, Stragglers

= AFS/NFS but files are immutable

VW with 4 VMs

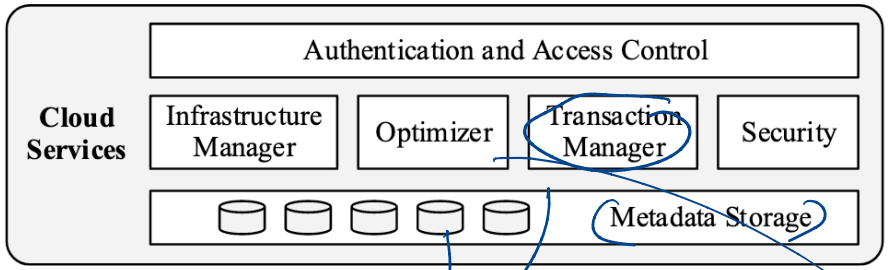
simple LRU




"File stealing"

→ if a worker is done early it steals work from slow worker

# CLOUD SERVICES




## Concurrency Control

↳ How to handle updates from many users

Snapshot Isolation ≡ Isolation level

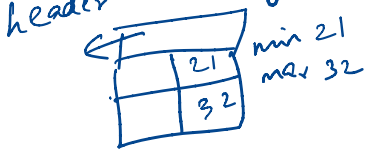
↓

all the reads come from a consistent version

## Pruning

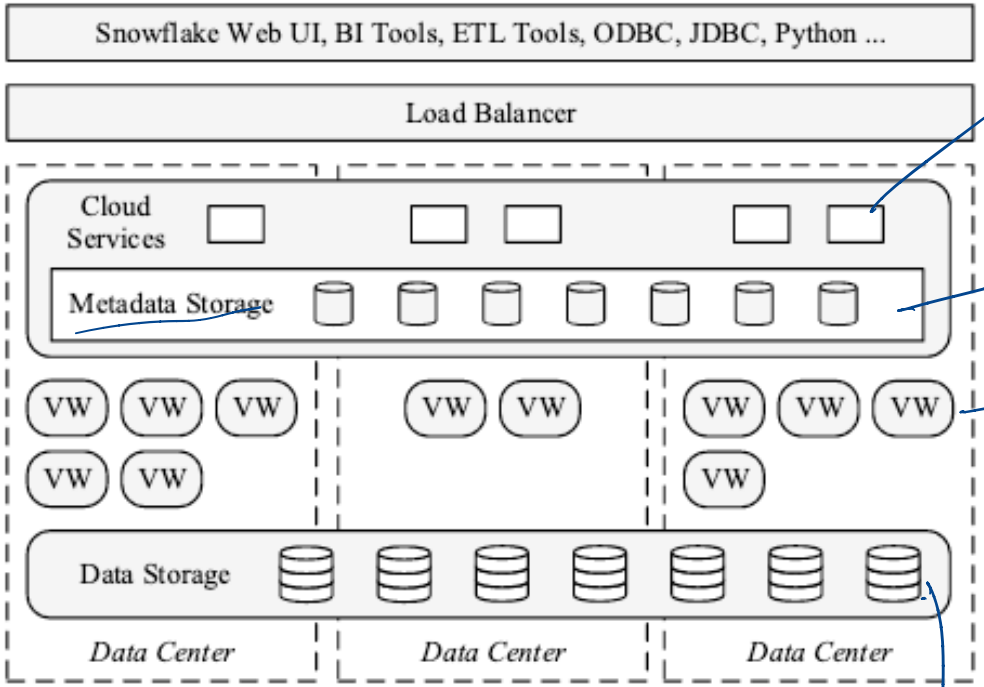
↳ Another way to improve locality

↳ It tries to skip files that don't have relevant tuples.




Age >= 35 ← skip this file

# FAULT TOLERANCE




reminiscent of Ray GCS?

restart services if they fail

Replicate metadata storage

Nothing. Ephemeral. If failure, retry the query.

Replicate data across data centers.

# SEMI STRUCTURED DATA

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

no type information

integer?

missing some fields

Extraction operation

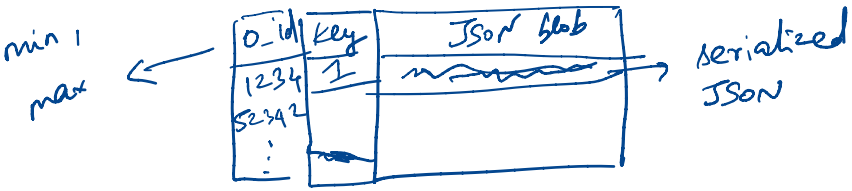
↳ to access a field inside JSON efficiently

Flattening

↳ arrays of JSON objects can create rows out of them.

Infer types, Pruning

Within a file




within a file try to infer types

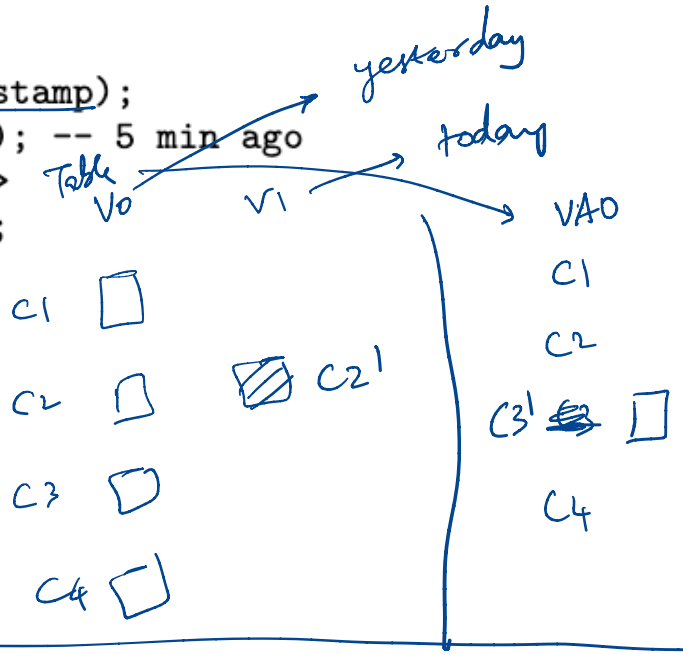
# TIME TRAVEL?

90 day retention policy

→ over table versions

you can query multiple versions of a table

```
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');
```

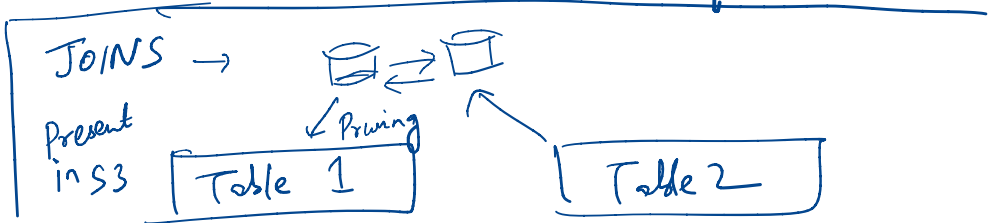



Multiple versions of table (MVCC)

Undo accidental deletes → new command UNDROP

Cheap to clone / snapshot a table

↳ Copy on Write



# SECURITY

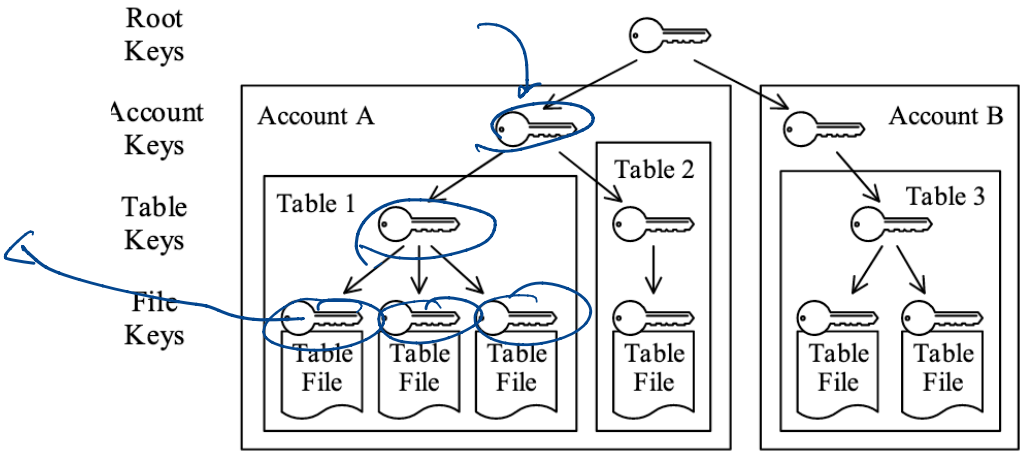
Hierarchical key management

↳ You encrypt a child key using parent key

→ limits what gets accessed when a key is compromised

Key rotation, re-keying

~~user~~ refresh keys being used



# SUMMARY, TAKEAWAYS

Snowflake

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

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

↓ limit number of files read

# AEFIS FEEDBACK

# DISCUSSION

https://forms.gle/ZFosdUnizXYABAE86

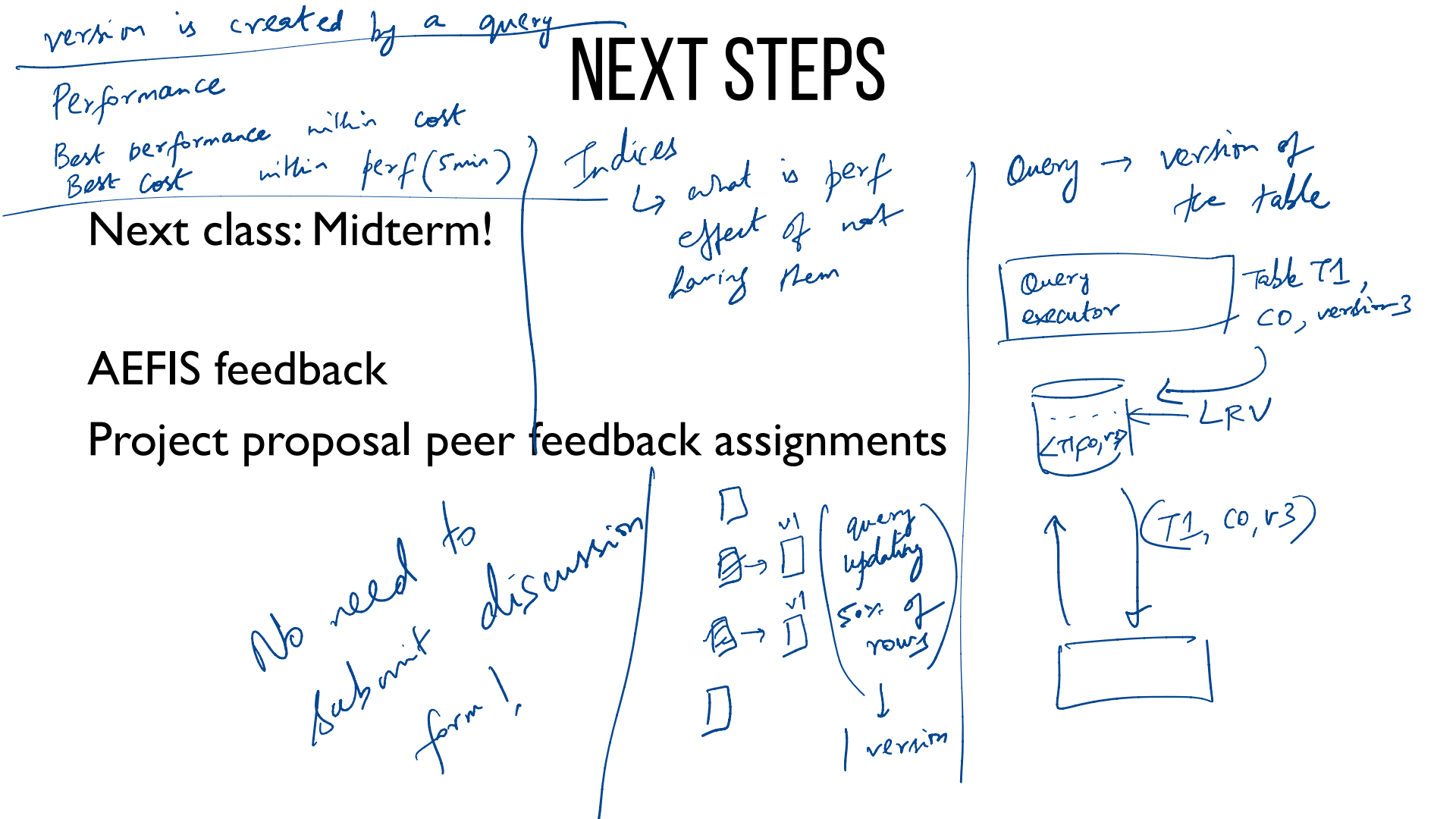
We see how Snowflake leads to the design of an elastic data warehouse. If we were to similarly design an Elastic PyTorch for training how would the design look? What are some design trade-offs compared to existing PyTorch?

# NEXT STEPS

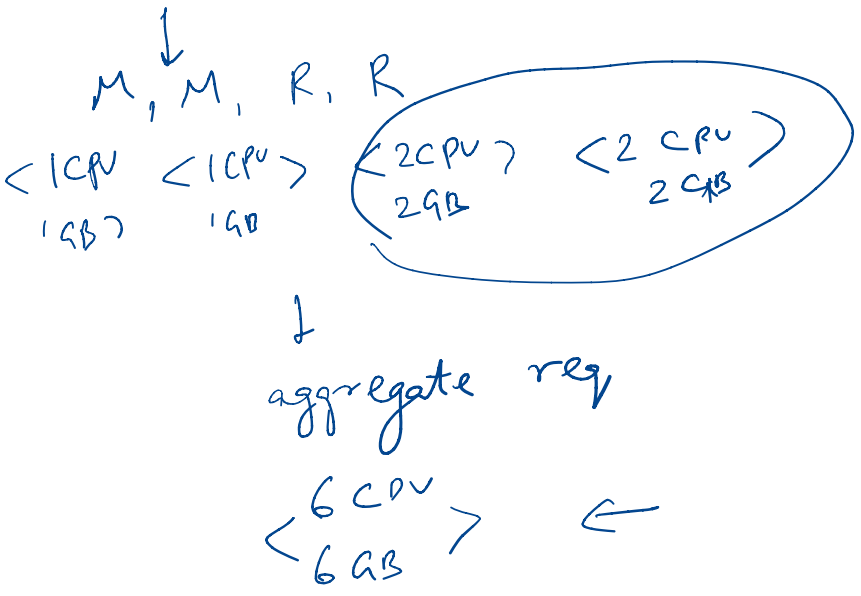
Next class: Midterm!

AEFIS feedback

Project proposal peer feedback assignments



DRF → task dependencies




Doesn't have a time dimension

Does this work?

DRF ⟶ Instantaneous fair scheme

→ Upstream tasks

Down stream tasks ⇒ the downstream tasks don't inherit shares

RDDs

↳ Immutable vs. materialized

---

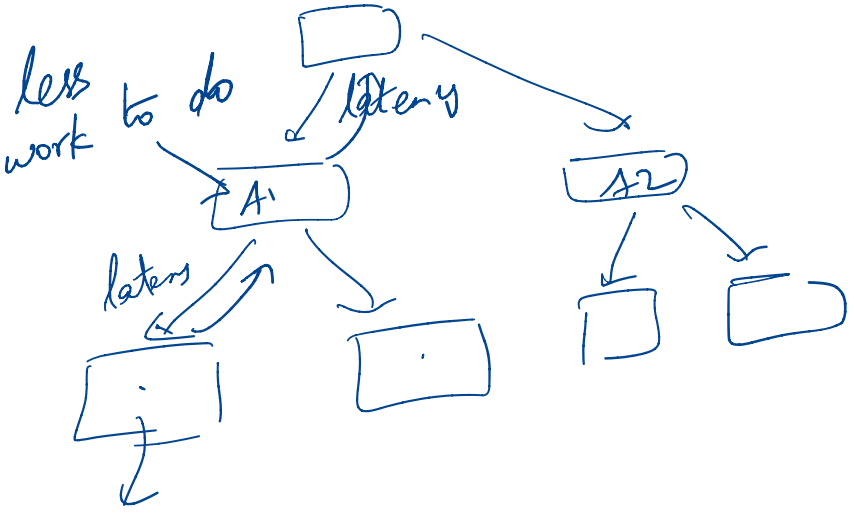
(b) Improving FT

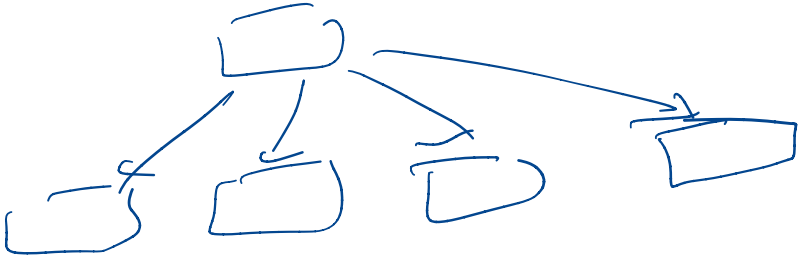
↳ just lineage is default

↳ see 5.4 in paper

checkpoints can shorten the lineage

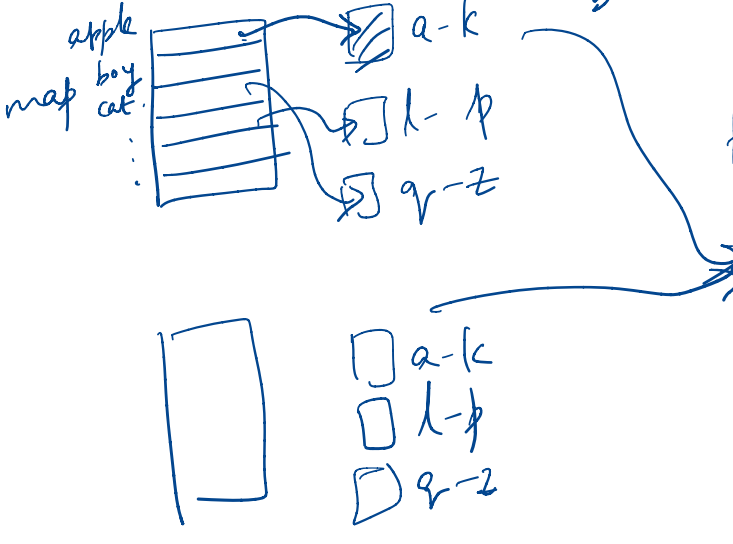
mid level aggregator



# Sorting in MapReduce


sort this bucket




Sample the data

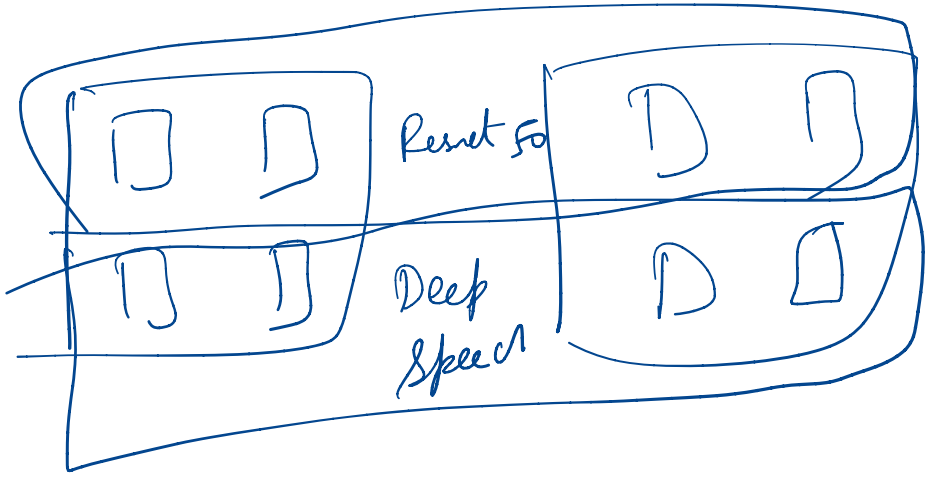
random 1% of data

histogram of words

→ fetch ~~its~~ to 1 machine

compute buckets

Gandiva, DRF




Sharing incentive if

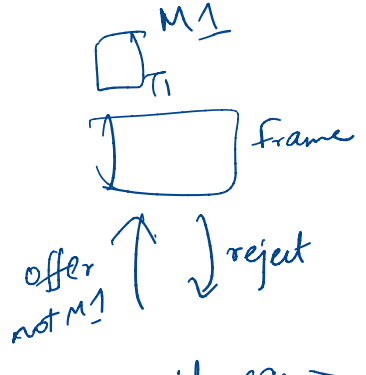
shared allocation is as good as having small exclusive cluster

---

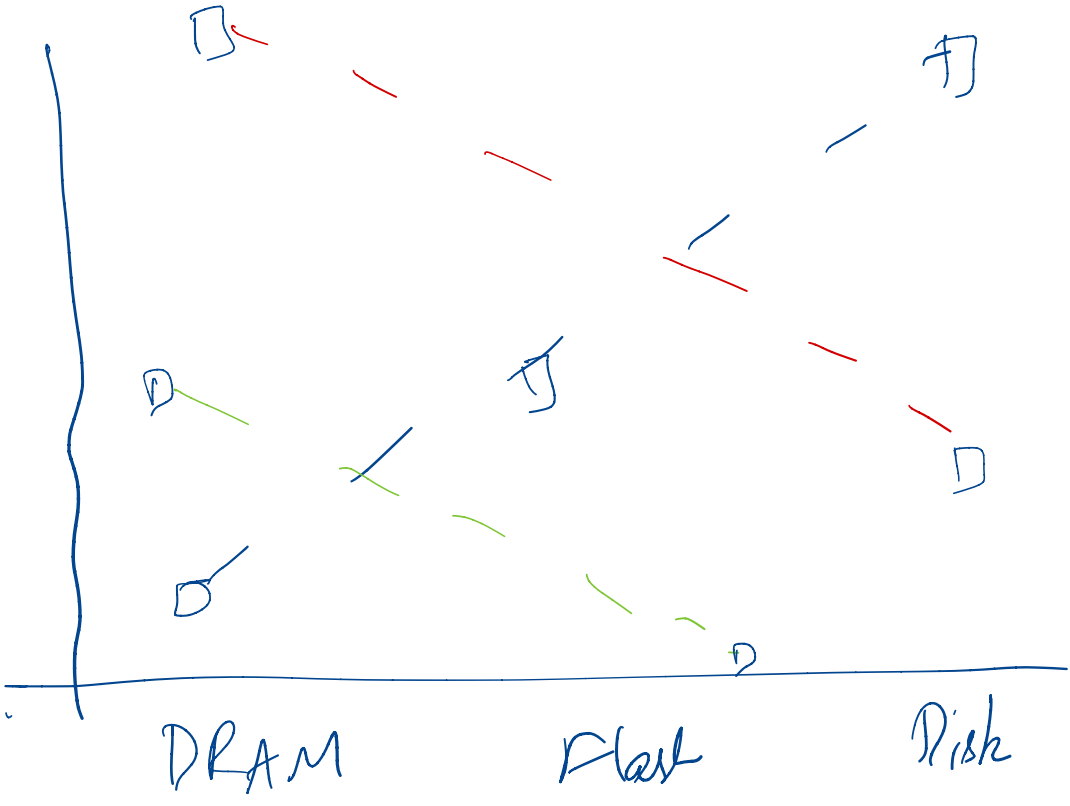
DRF → task preferences

↓

co-locate some tasks?



Workload




Blue: Capacity, latency

Red: Bandwidth, → Power usage?

Green: Price / GB

Assumption:
Process failure

MR failures

(i) Map Complete failure

(ii) In-progress map

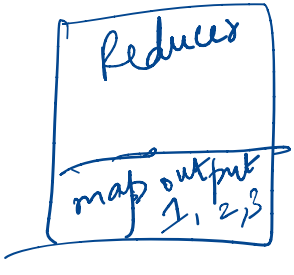
(iii) In-progress reduce

(a) Map output is already on disk, nothing to be done

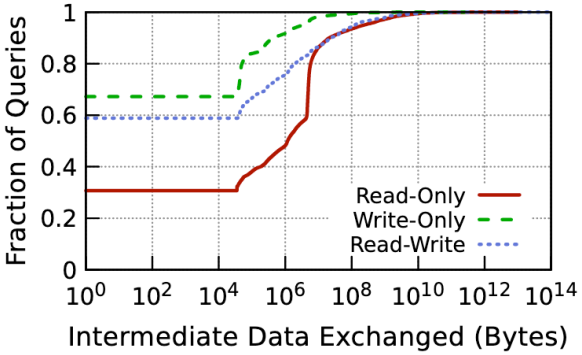
(b) restart map task

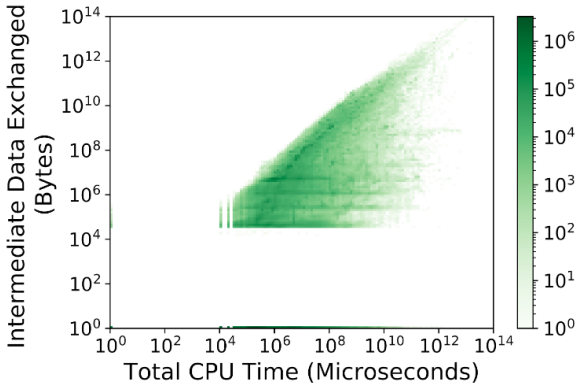
(b) restart reduce task

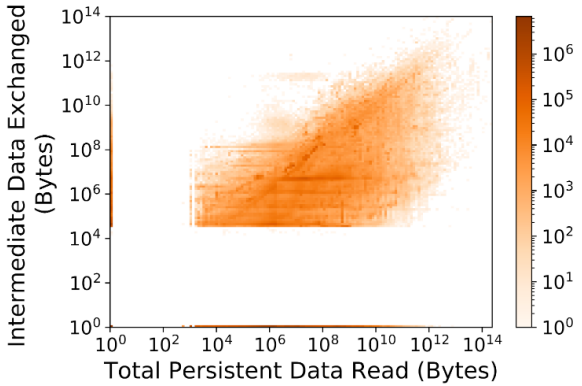
all map outputs still available (only process failure)




→ prev map tasks


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)
$10^6$ $10^5$ $10^4$ $10^3$ $10^2$ $10^1$ $10^0$