Good morning!

CS 744: SNOWFLAKE

Shivaram Venkataraman
Fall 2021
- Assignment 1 grades out! → Yien on Piazza/email
- Assignment 2 by mid-week
- Midterm on Thursday! Seating layout?
  - Randomize seating
SparkSQL/Scope: Given a query how do you run it efficiently?

Snowflake: How do you build an elastic data warehouse?

Able to increase or decrease resource use as workload changes.

OLAP queries: A collection of data and you do analytics queries on it.
CLOUD COMPUTING STACK

Scalable Storage Systems

Computational Engines

Machine Learning

SQL

Pytorch

MapReduce / Spark

Amazon EC2

SQL as a software service

Elastic MapReduce

Amazon EC2 - VMs

Amazon S3

GFS
SNOWFLAKE: GOALS

Software-as-a-Service → Users don't need to install/manage this

Elastic → Use resources as required

Highly Available → Fault tolerance

Semi-Structured Data → Traditional warehouses only support structured data

JSON queries on specific fields
We are isolated from each other.

VWs are isolated from each other.

Data storage moved across VWs.

Admin & Optimizer in Cloud

Cloud Services

Data storage shared across VWs.

VWs are isolated from each other.

Data Storage

Authentication and Access Control

Infrastructure Manager

Optimizer

Transaction Manager

Security

Metadata Storage

Virtual Warehouse

Cache

Virtual Warehouse

Cache

Virtual Warehouse

Cache

Virtual Warehouse

Cache
STORAGE VS COMPUTE

- Shared Nothing
  - CPU, disk resources tied together
  - Some queries need a lot of disk, less CPU, can lead to under-utilization

- Multi Cluster, Shared Data
  - Separate out storage & compute
  - Scale storage independently
  - Handle irregular arrival of queries
  - Downside: No locality of access

CPU → local disk

EC2 instances

Amazon S3

Shared Nothing

Multi Cluster, Shared Data
**STORAGE: HYBRID COLUMNAR**

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>32</td>
</tr>
<tr>
<td>Bob</td>
<td>22</td>
</tr>
<tr>
<td>Eve</td>
<td>24</td>
</tr>
<tr>
<td>Victor</td>
<td>27</td>
</tr>
</tbody>
</table>

Entire table = 4 rows

Row-oriented:
- Alice, 32, Bob, 22
- Eve, 24, Victor, 27

Hybrid Columnar:
- Alice, Bob, 32, 22
- Eve, Victor, 24, 27

Each partition or file table has 2 rows in every range query. Start offset & length & ID compression saves space & also network traffic.
VIRTUAL WAREHOUSES

Elasticity, Isolation

- For each user, spin up some EC2 instances = VW
  -> Isolation as instances separate for each user
  -> Can turn off these machines once you are done

Local caching, Stragglers

Local SSD

LRU caching policy

fetch data from S3 (Partition)

P2P communication

Stragglers happen due to data skew

one machine has lot more work to do
Data/Partitions are versioned: v1, v2

ConcURRENCY Control

A query will see all the files at a specific version. If a query is associated with v1, all reads come from a consistent version. "Snapshot Isolation"

MVCC

PRUNING

The range of various columns in each block: min 22, max 34

Files: limit or exclude files which are not going to be used in the query

Track version numbers of blocks/files

Cloud Services

Authentication and Access Control

Infrastructure Manager

Optimizer

Transaction Manager

Security

Metadata Storage
Fault Tolerance

These are again stateless replicated across AZs. Just spin up new ones. "Retry queries, replicate across data centers." Availability Zone.
**SEMI-STRUCTURED DATA**

Extraction operation
- access individual fields with JSON as part of query

Flattening
- flatten nested JSON objects

Infer types, Pruning
- not present in all records

Inside each Partition

\[ \text{JSON} \rightarrow \text{order-id} \rightarrow \text{integer} \]
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) → Run queries which say
processes data till yesterday

Undo accidental deletes

Cheap to clone / snapshot a table
Hierarchical key management

Key rotation, re-keying
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/buUDM9nRs6Gg9tURA
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?

Data Parallel → row-based partitioning

hybrid columnar?

Compute

Storage

Parameter Servers

Model, train data

Test data

(read) updated every iteration ⇒ overhead?
NEXT STEPS

Next class: Midterm!