WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 839: Design the Next-Generation Database

Lecture 22: Snowflake

Xiangyao Yu

4/9/2020

Announcements

Course project

- ~~Submission deadline: **Apr. 23**~~
- ~~Peer review: **Apr. 23 – Apr. 30**~~
- Presentation: **Apr. 28 & 30**
- Submission deadline: **May 4**

Will create google sheet for presentation signup

Discussion Highlights

Optimal design that combines the advantages?

- Athena with instances pre-running
- Hybrid instance store and S3; decide caching based on the workload
- High-quality code compilers
- Heterogeneous system that combines all the existing systems together

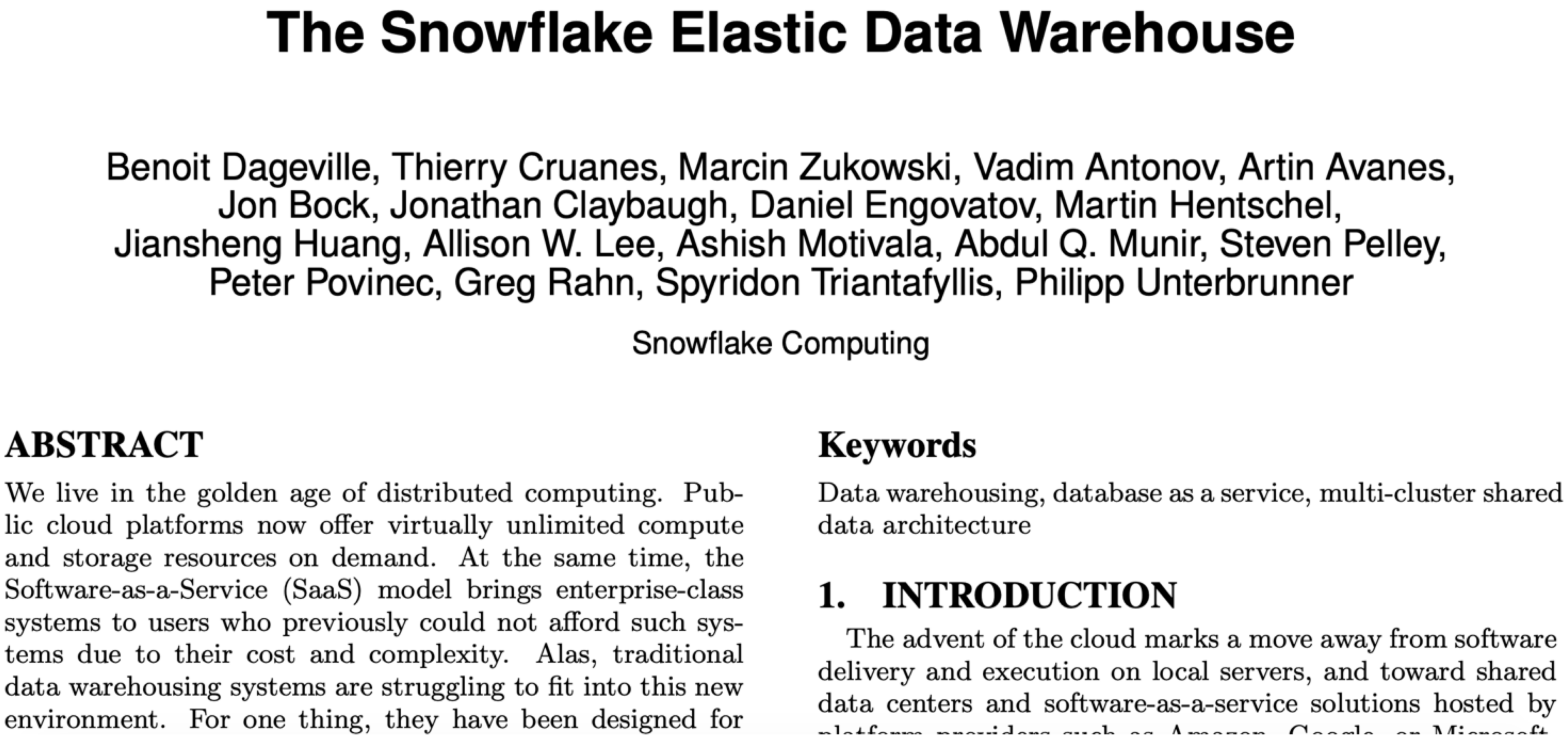
Optimization opportunities for serverless databases?

- Optimize resource sharing among users (e.g., cache, computation)
- SW/HW codesign
- Heterogeneous hardware and storage (e.g., different function on different hardware)
- Scale computation and storage on demand
- Keep instances pre-warmed to reduce cold starts

Cloud databases benefit from new hardware?

- Using GPU
- SmartSSD
- RDMA and SmartNIC (e.g., shared cache in SSD, computation offloading)
- Persistent memory to improve bandwidth and aid fast restarts

Today’s Paper



The Snowflake Elastic Data Warehouse

Benoit Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, Allison W. Lee, Ashish Motivala, Abdul Q. Munir, Steven Pelley, Peter Povinec, Greg Rahn, Spyridon Triantafyllis, Philipp Unterbrunner

Snowflake Computing

ABSTRACT

We live in the golden age of distributed computing. Public cloud platforms now offer virtually unlimited compute and storage resources on demand. At the same time, the Software-as-a-Service (SaaS) model brings enterprise-class systems to users who previously could not afford such systems due to their cost and complexity. Alas, traditional data warehousing systems are struggling to fit into this new environment. For one thing, they have been designed for

Keywords

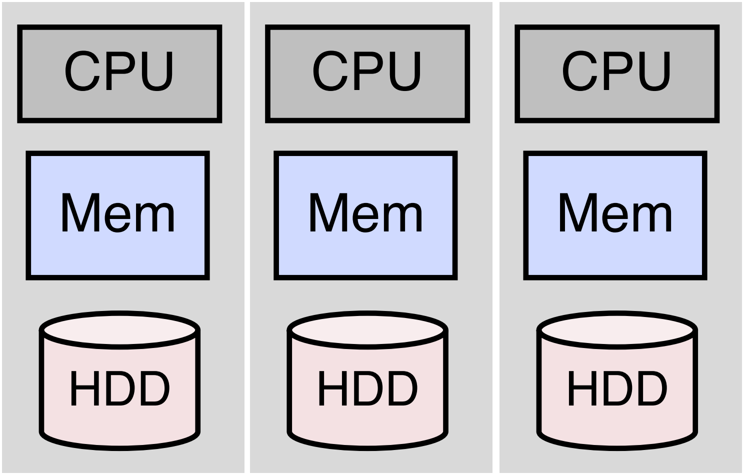
Data warehousing, database as a service, multi-cluster shared data architecture

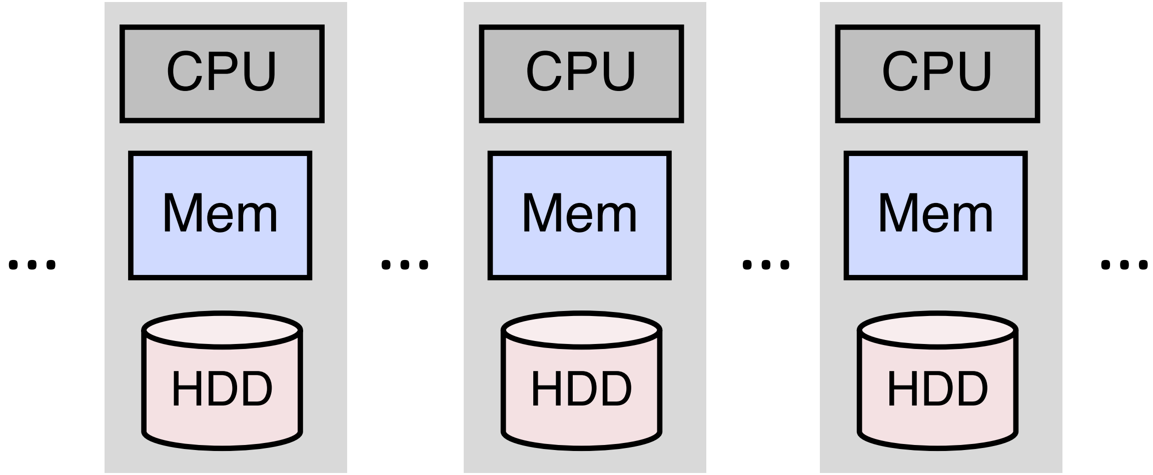
1. INTRODUCTION

The advent of the cloud marks a move away from software delivery and execution on local servers, and toward shared data centers and software-as-a-service solutions hosted by platform providers such as Amazon, Google, or Microsoft.

SIGMOD 2016

On-Premises vs. Cloud





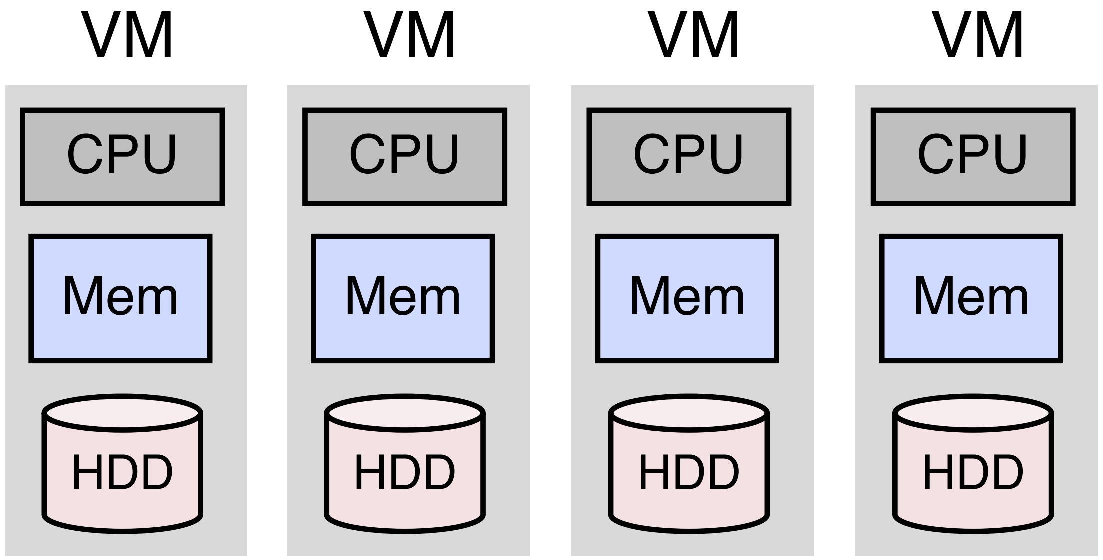
On-premises

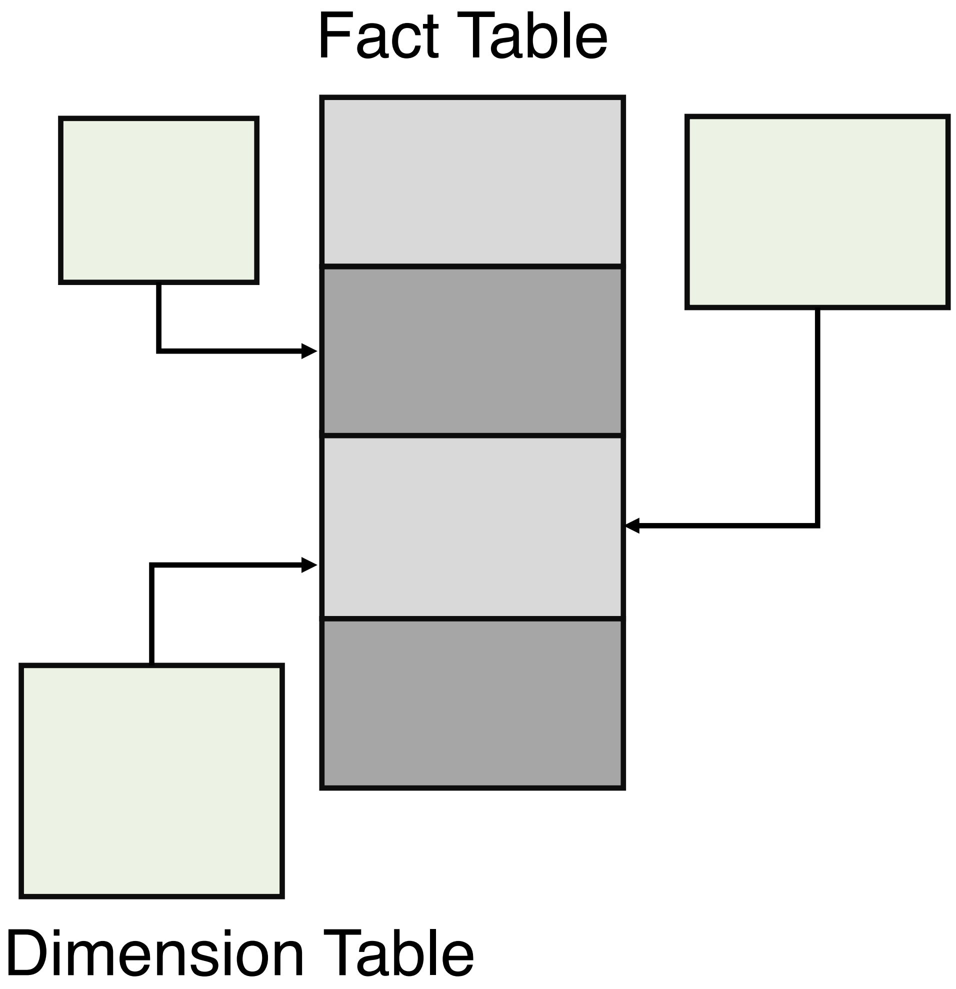
- Fixed and limited hardware resources

Cloud

- Virtually infinite computation & storage
- Pay-as-you-go

Shared Nothing – Advantages

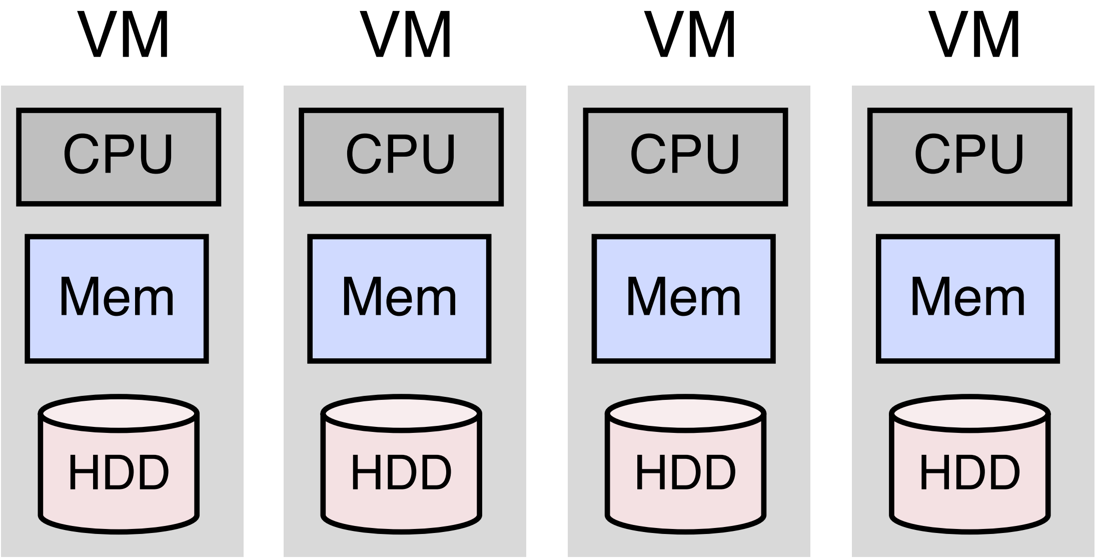


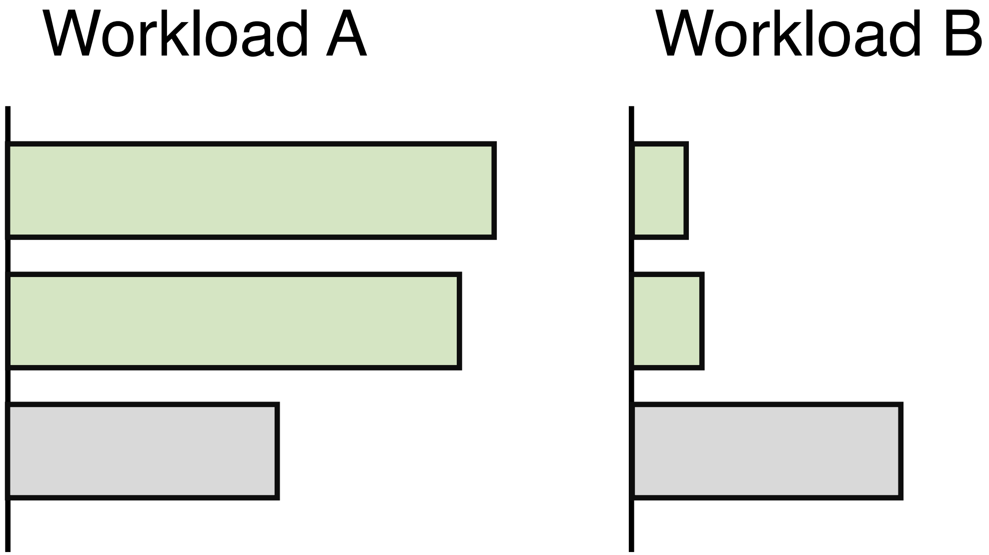


Scalability: horizontal scaling

- Scales well for star-schema queries

Shared Nothing – Disadvantages



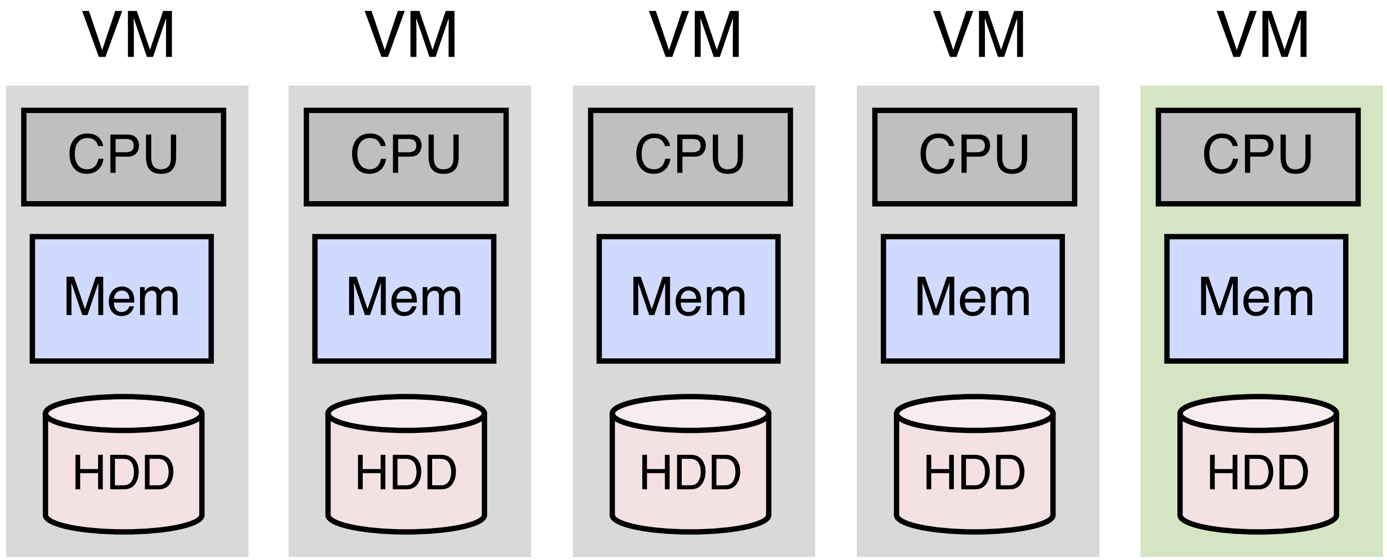


More CPU intensive

Less CPU intensive

Heterogeneous workload

Shared Nothing – Disadvantages

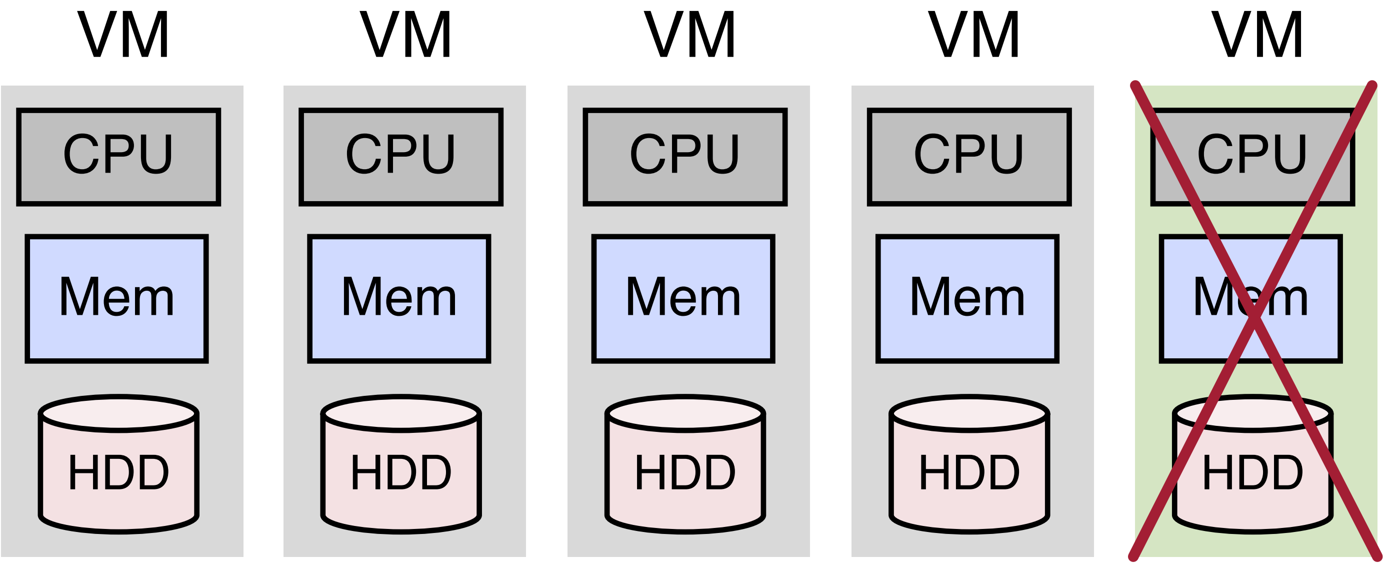


Heterogeneous workload

Membership changes

- Add a node: data redistribution

Shared Nothing – Disadvantages

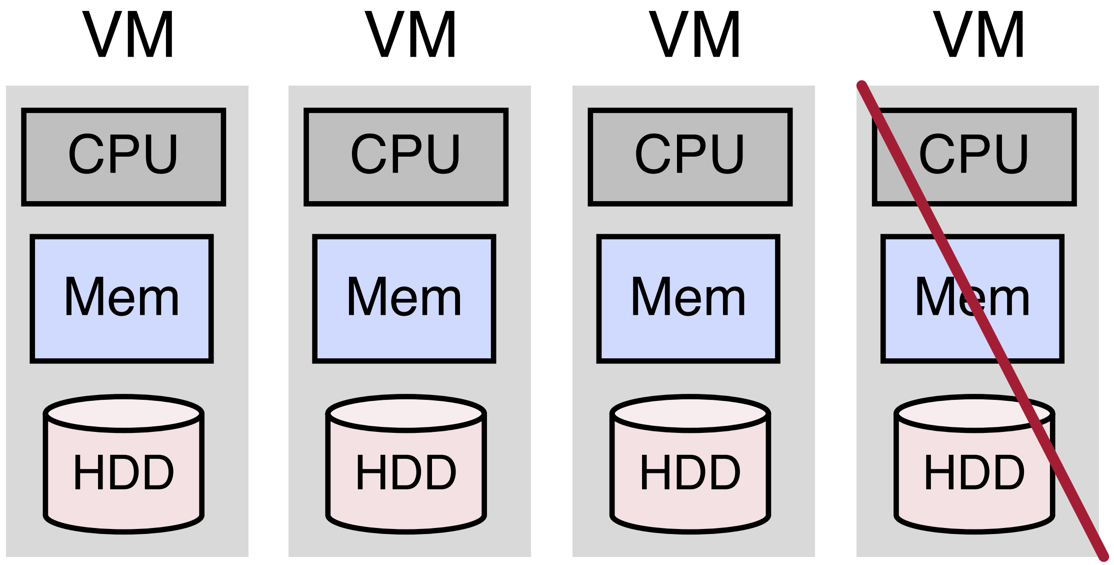


Heterogeneous workload

Membership changes

- Add a node: data redistribution
- Delete a node: fault tolerance

Shared Nothing – Disadvantages



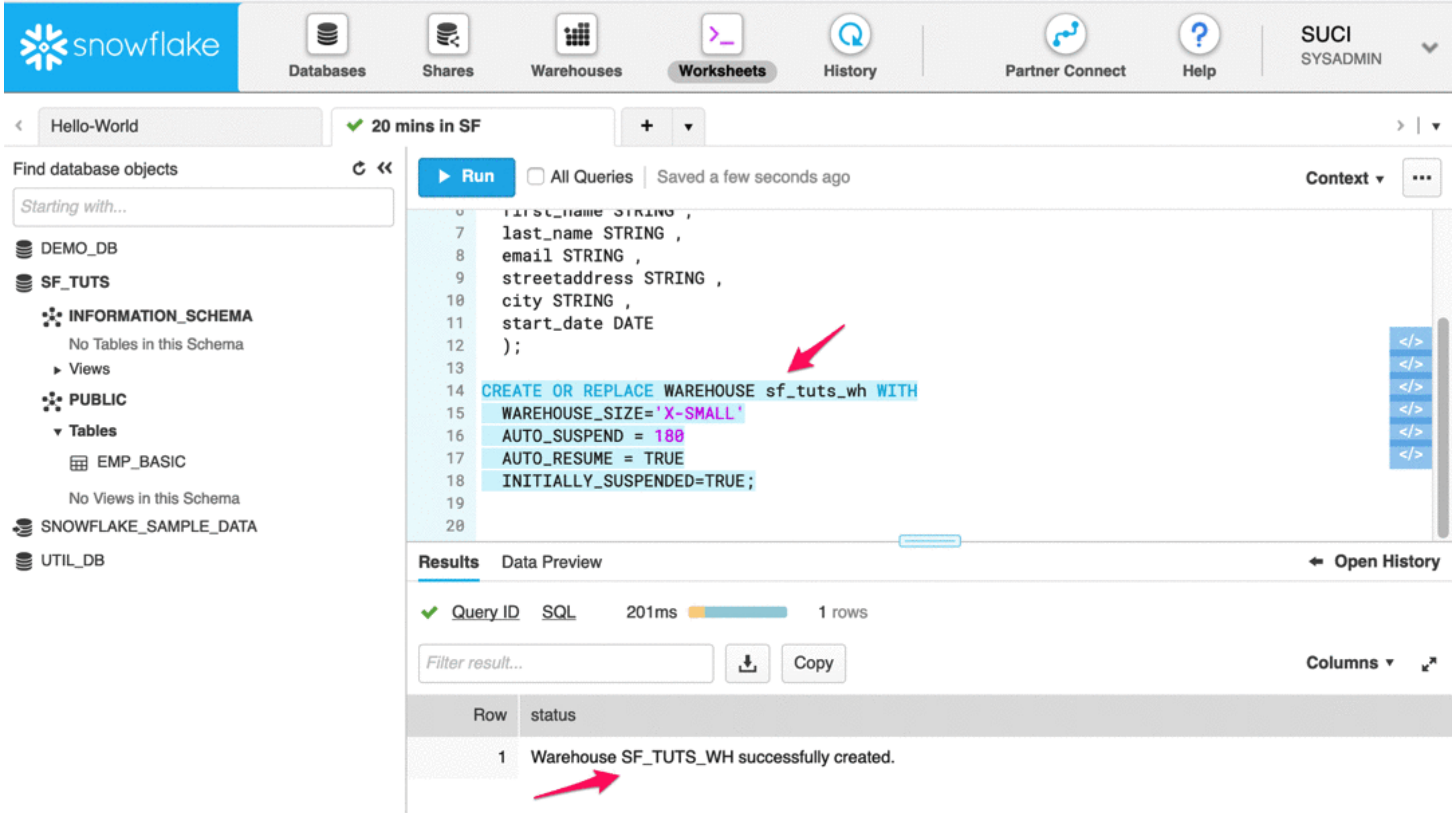
Heterogeneous workload

Membership changes

Online upgrade

- Similar to membership change

Web User Interface



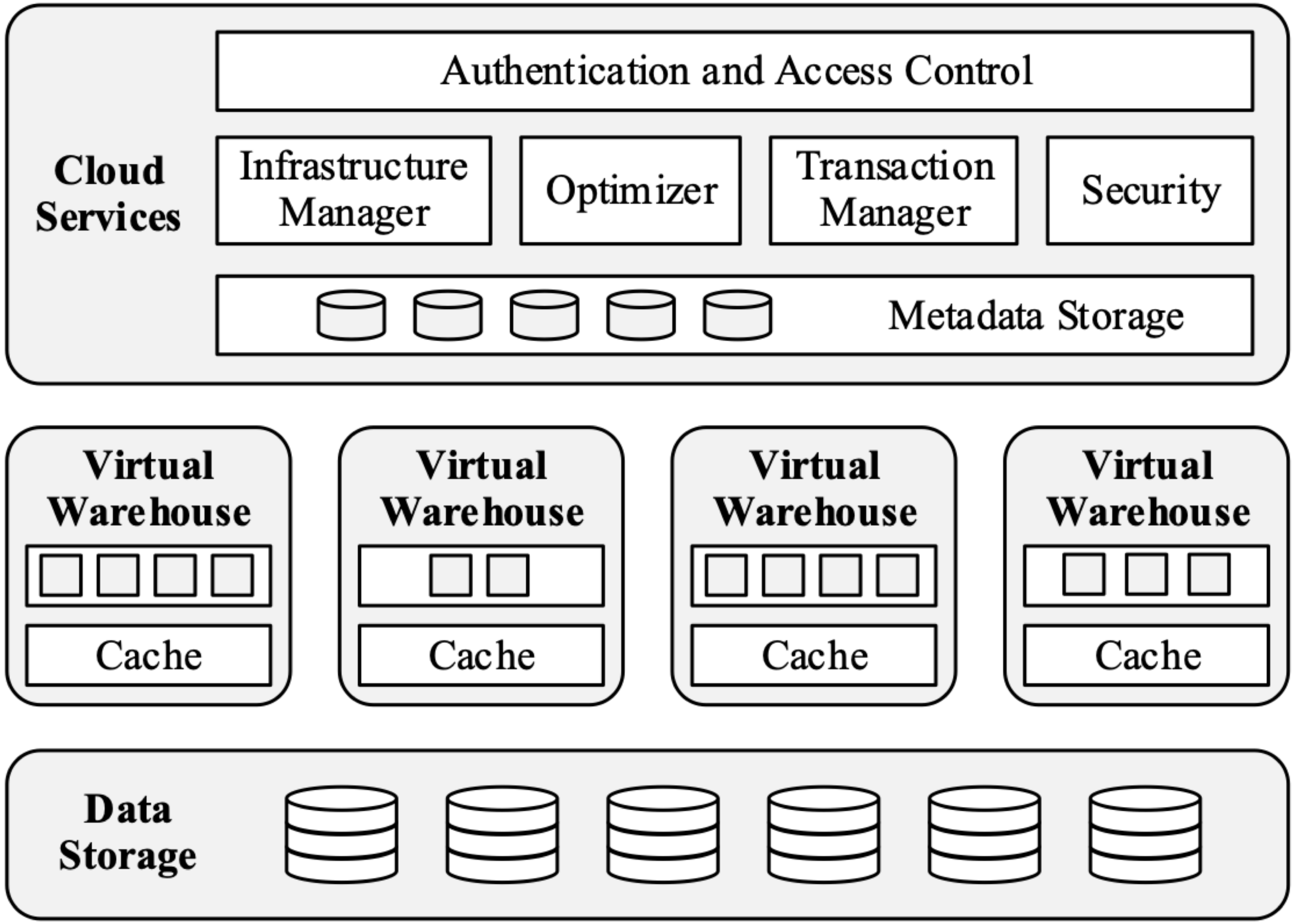
```
first_name STRING ,
last_name STRING ,
email STRING ,
streetaddress STRING ,
city STRING ,
start_date DATE
);

CREATE OR REPLACE WAREHOUSE sf_tuts_wh WITH
  WAREHOUSE_SIZE='X-SMALL'
  AUTO_SUSPEND = 180
  AUTO_RESUME = TRUE
  INITIALLY_SUSPENDED=TRUE;
```

Row	status
1	Warehouse SF_TUTS_WH successfully created.

Serverless (similar to Athena)

Multi-Cluster Shared-Data Architecture



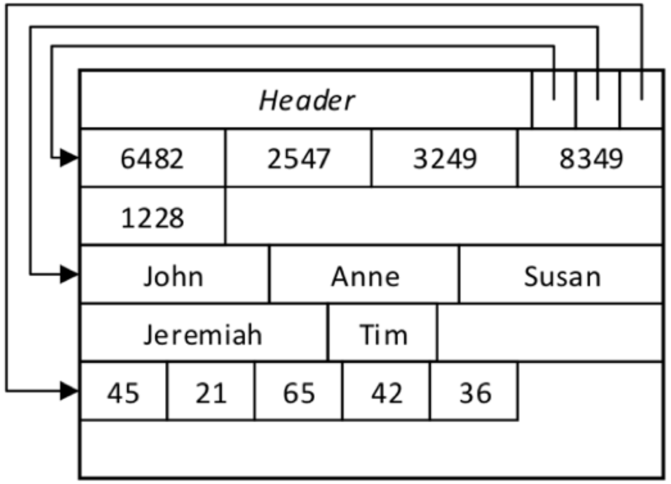
Control layer

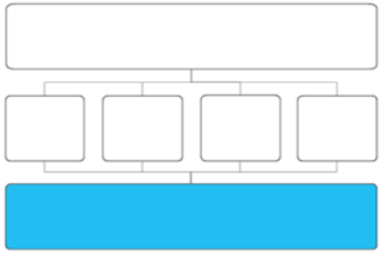
Compute layer

Storage layer

Architecture – Storage

Data format: PAX



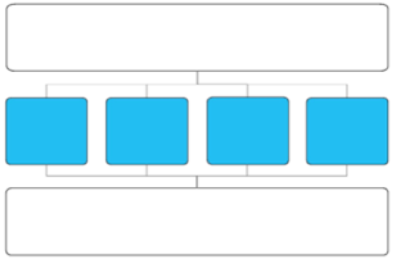


Data horizontally partitioned into immutable files (~16MB)

- An update = remove and add an entire file
- Queries download file headers and columns they are interested in

Intermediate data spilling to S3

Architecture – Virtual Warehouse

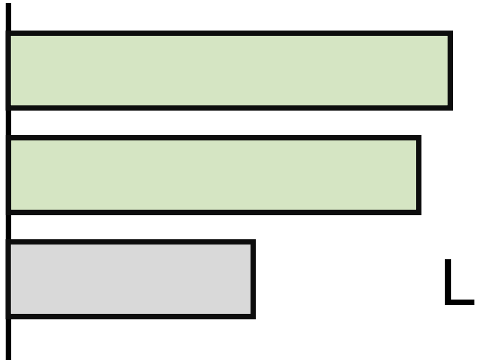


T-Shirt sizes: XS to 4XL

Elasticity and Isolation

- Created, destroyed, or resized at any point (may shutdown all VWs)
- User may create multiple VWs for multiple queries

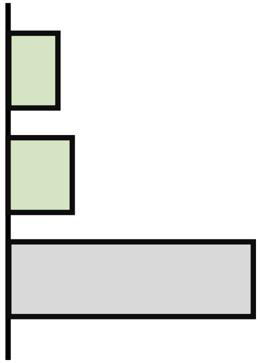
Workload A



Large VW

More CPU intensive

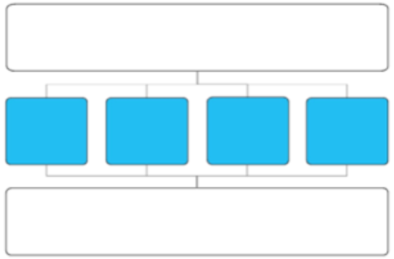
Workload B



Small VW

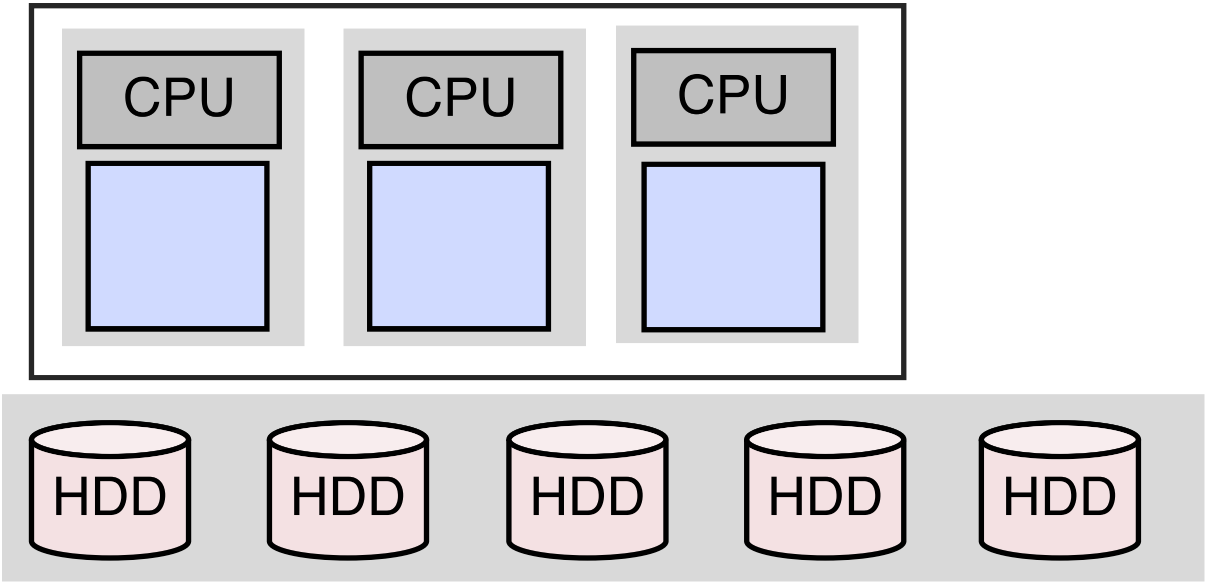
Less CPU intensive

Architecture – Virtual Warehouse

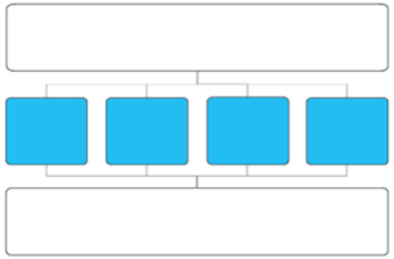


Local caching

- S3 data can be cached in local memory or disk



Architecture – Virtual Warehouse

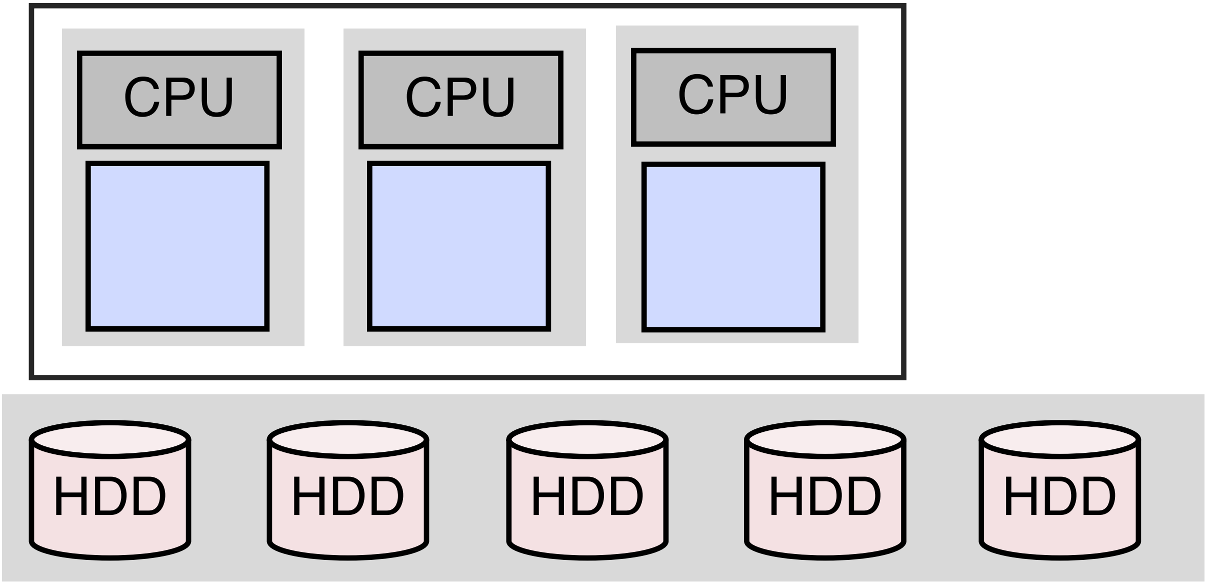


Local caching

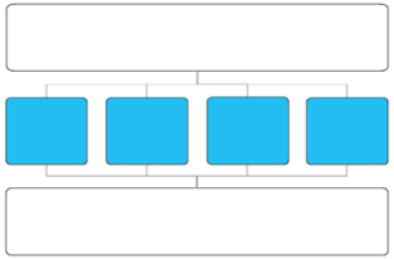
- S3 data can be cached in local memory or disk

Consistent hashing

- When the hash table (n keys and m slots) is resized, only n/m keys need to be remapped



Architecture – Virtual Warehouse

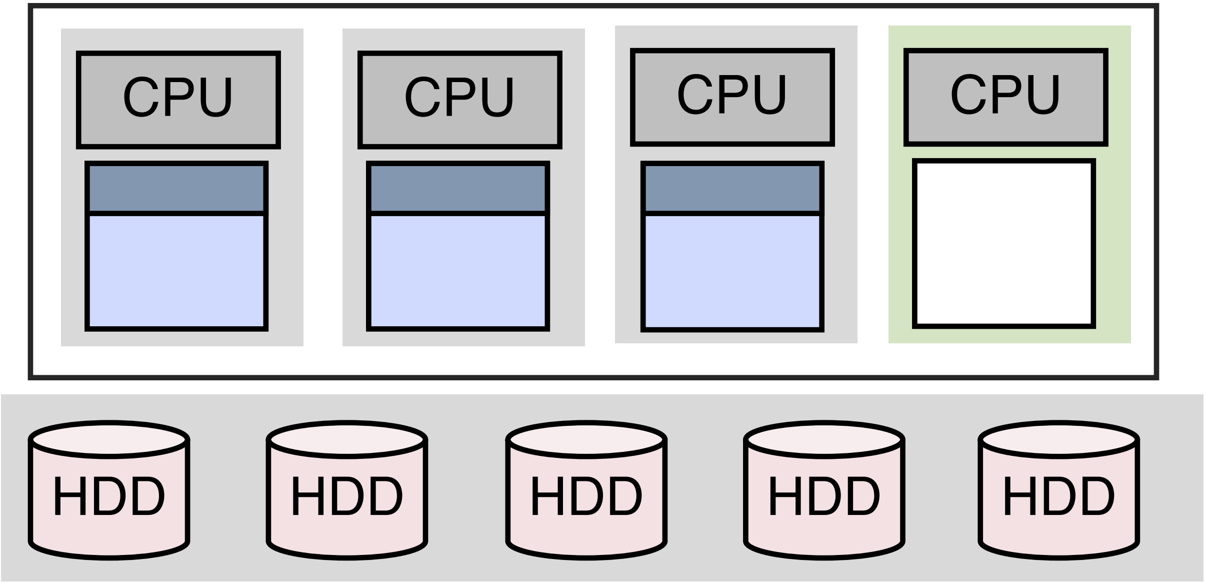


Local caching

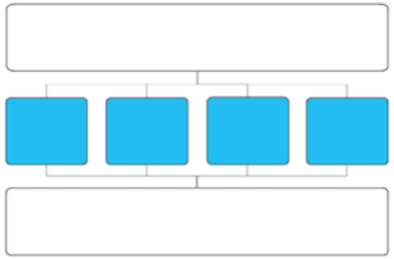
- S3 data can be cached in local memory or disk

Consistent hashing

- When the hash table (n keys and m slots) is resized, only n/m keys need to be remapped



Architecture – Virtual Warehouse

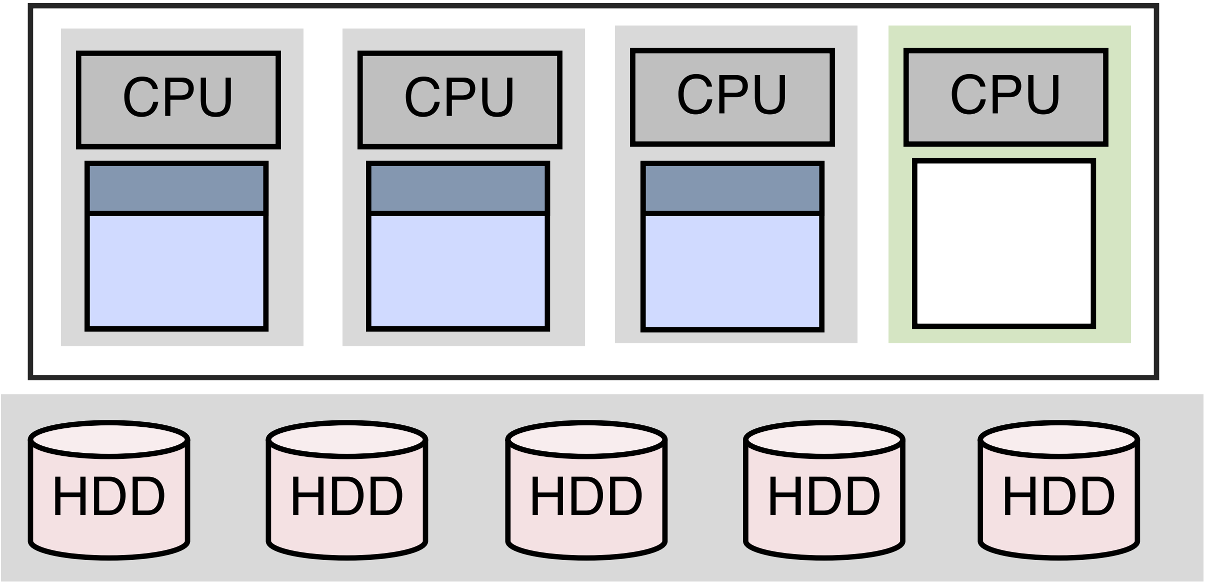


Local caching

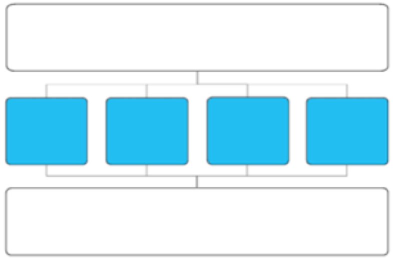
- S3 data can be cached in local memory or disk

Consistent hashing

- When the hash table (n keys and m slots) is resized, only n/m keys need to be remapped
- When a VW is resized, no data shuffle required; rely on LRU to replace cache content

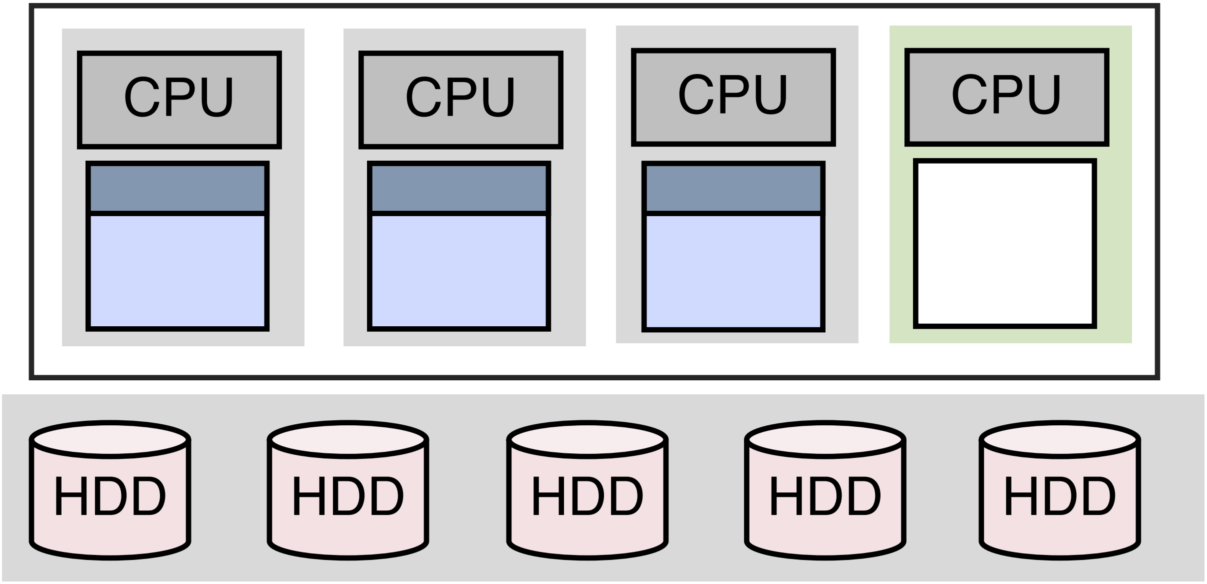


Architecture – Virtual Warehouse



Local caching

- S3 data can be cached in local memory or disk

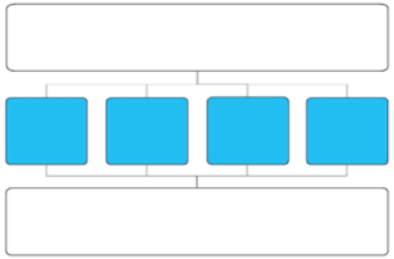


Consistent hashing

- When the hash table (n keys and m slots) is resized, only n/m keys need to be remapped
- When a VW is resized, no data shuffle required; rely on LRU to replace cache content

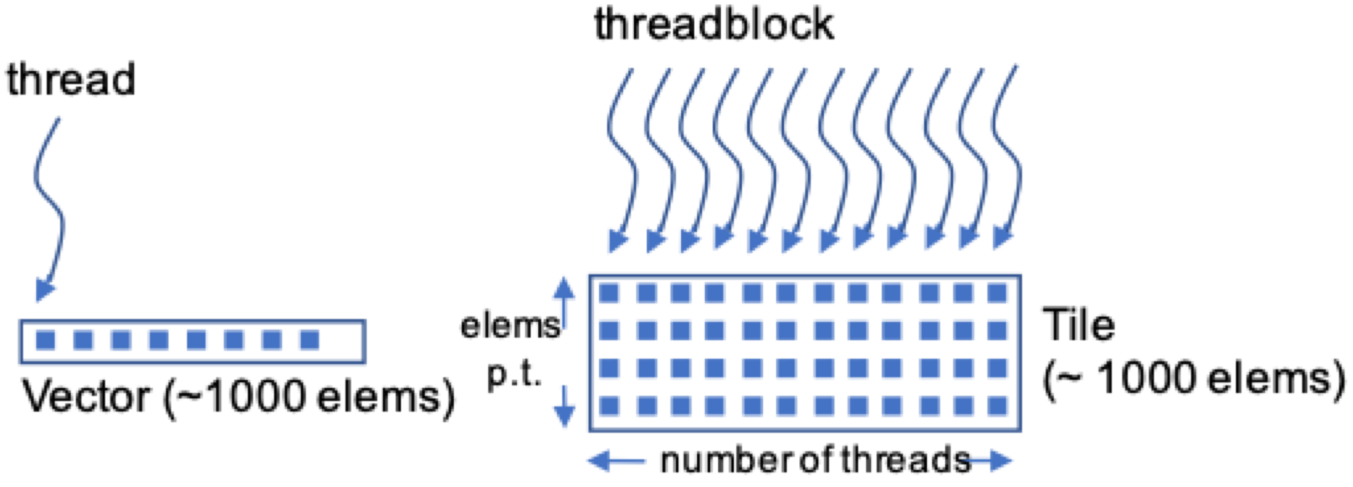
File stealing to tolerate skew

Architecture – Virtual Warehouse



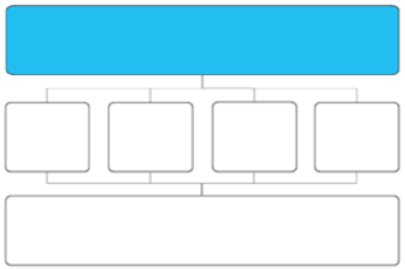
Execution engine

- Columnar: SIMD, compression
- Vectorized: process a group of elements at a time
- Push-based



(a) On CPU (b) On GPU

Architecture – Cloud Services



Multi-tenant layer shared across multiple users

Query optimization

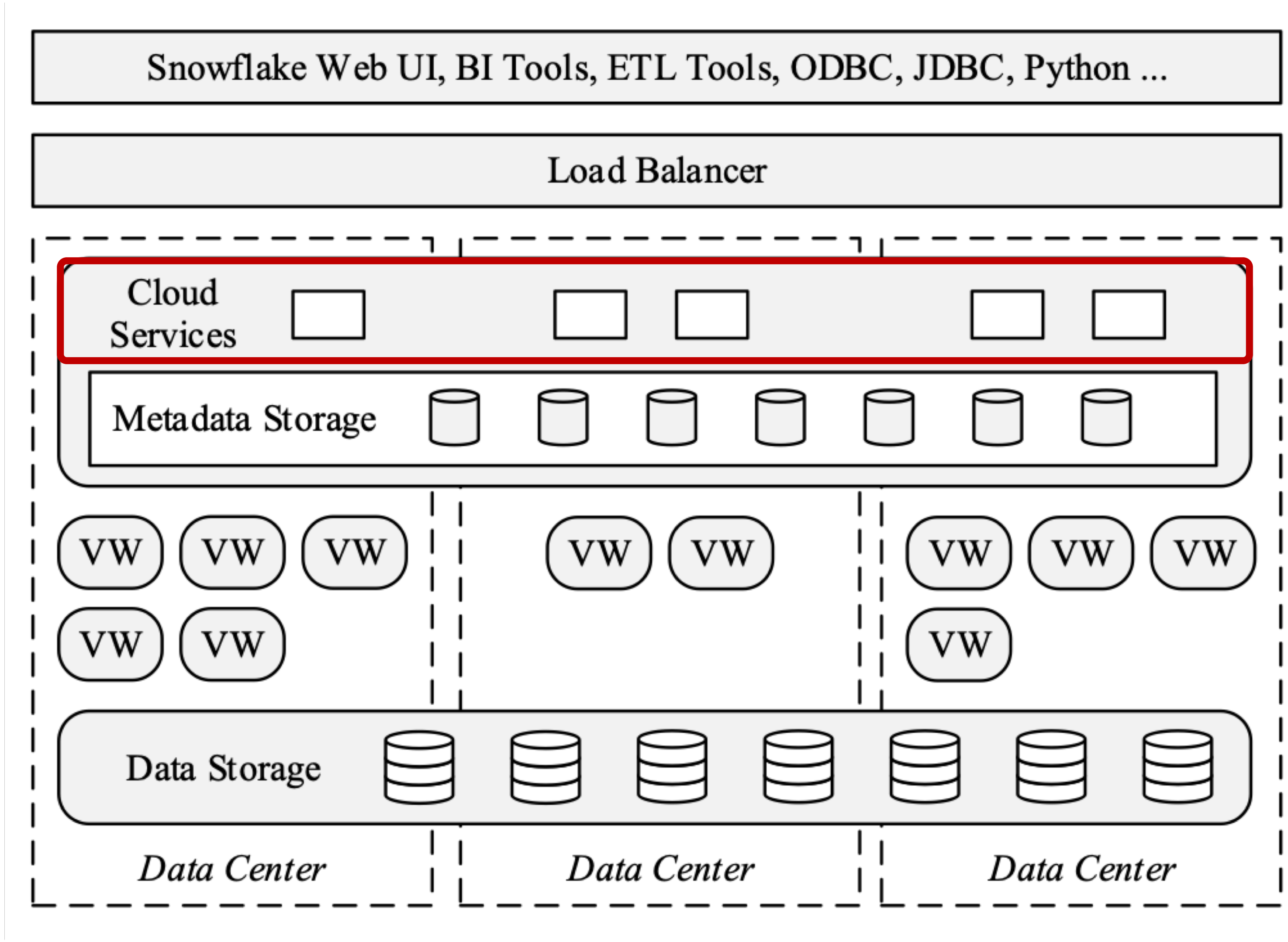
Concurrency control

- Isolation: snapshot isolation (SI)
- S3 data is immutable, update entire files with MVCC
- Versioned snapshots used for time traveling

Pruning

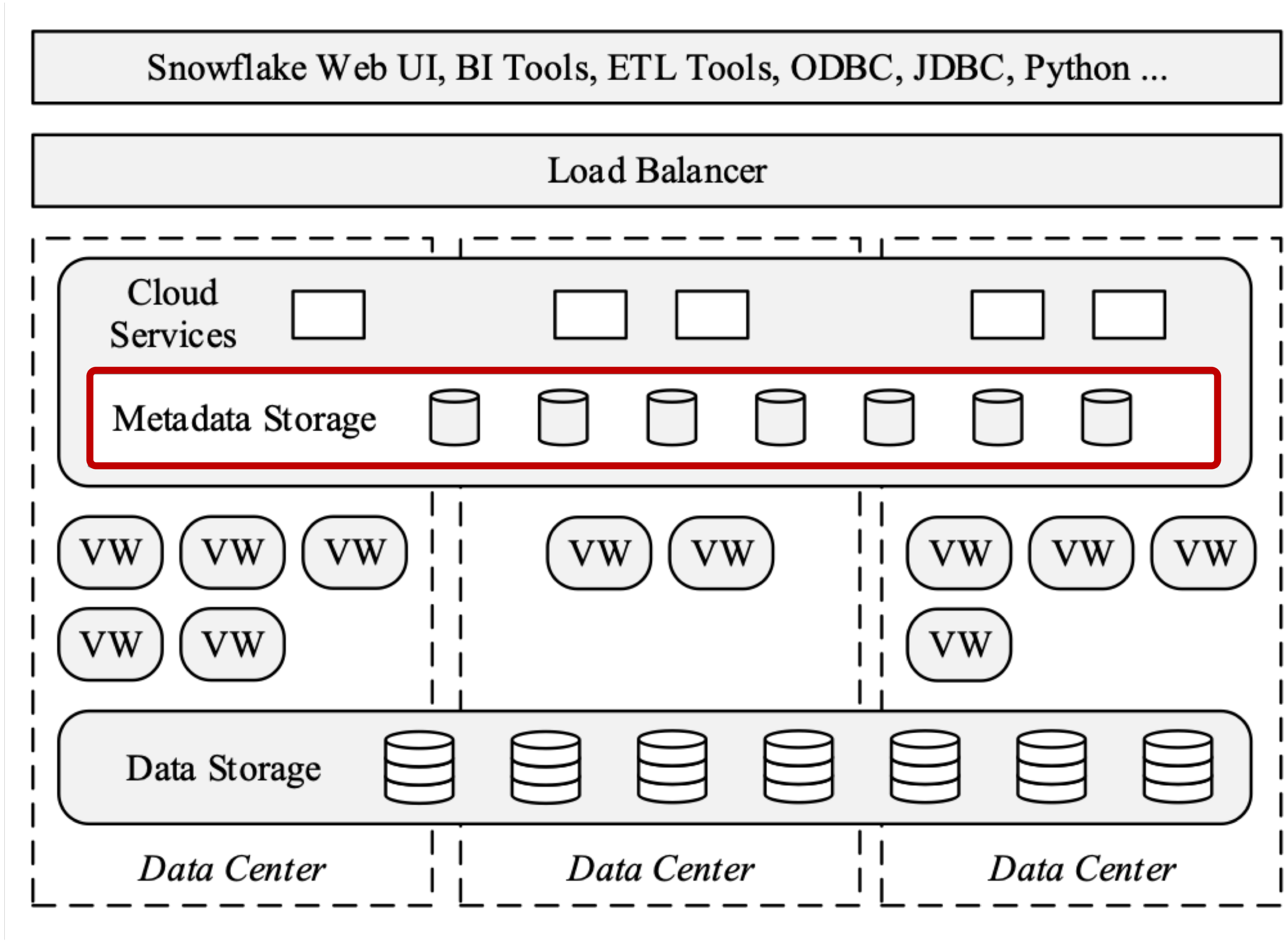
- Snowflake has no index (same in Athena, Presto, Hive, etc)
- Min-max based pruning: store min and max values for a data block

High Availability and Fault Tolerance



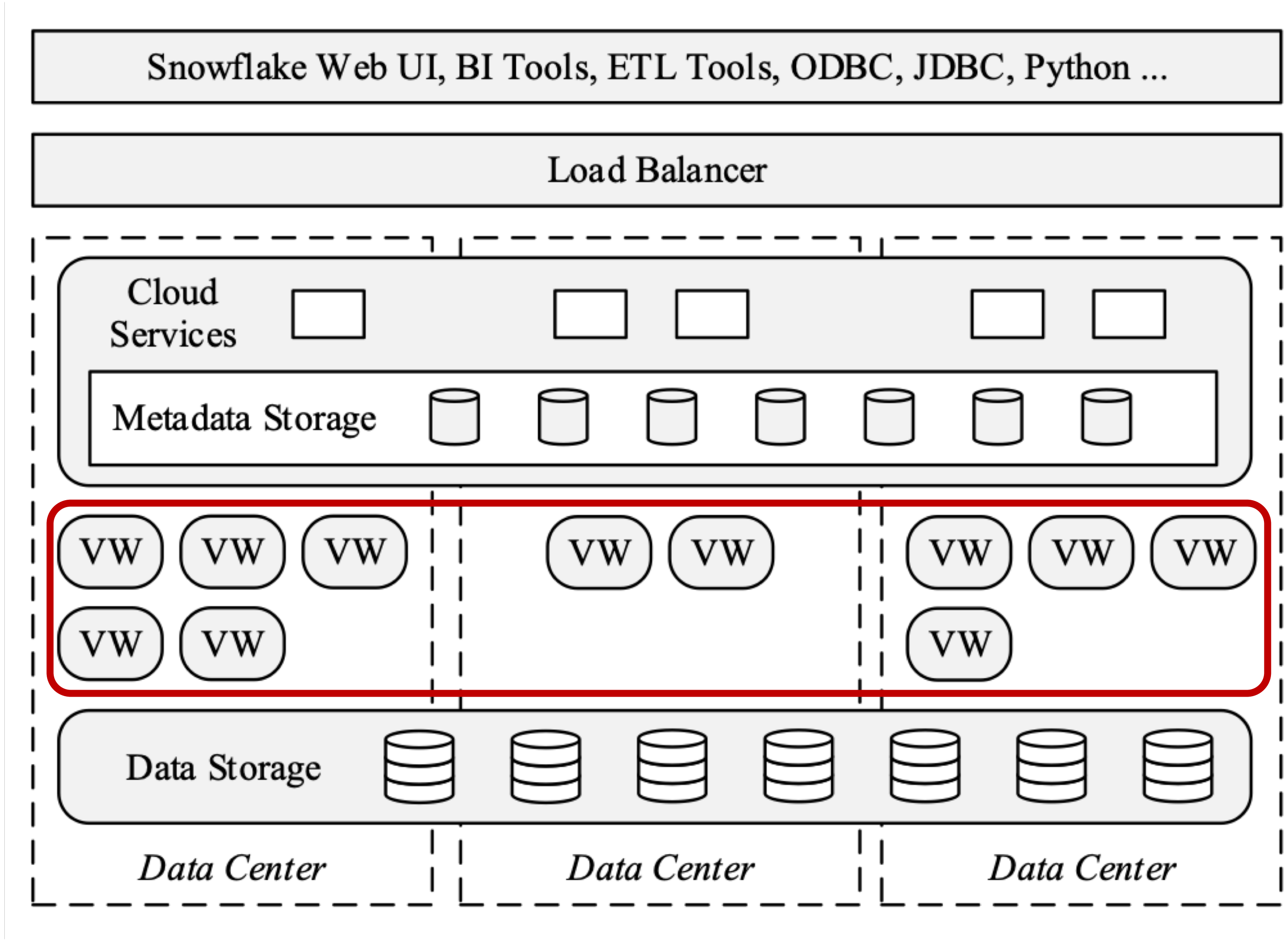
Stateless services

High Availability and Fault Tolerance



Replicated metadata

High Availability and Fault Tolerance



One node failure in VW

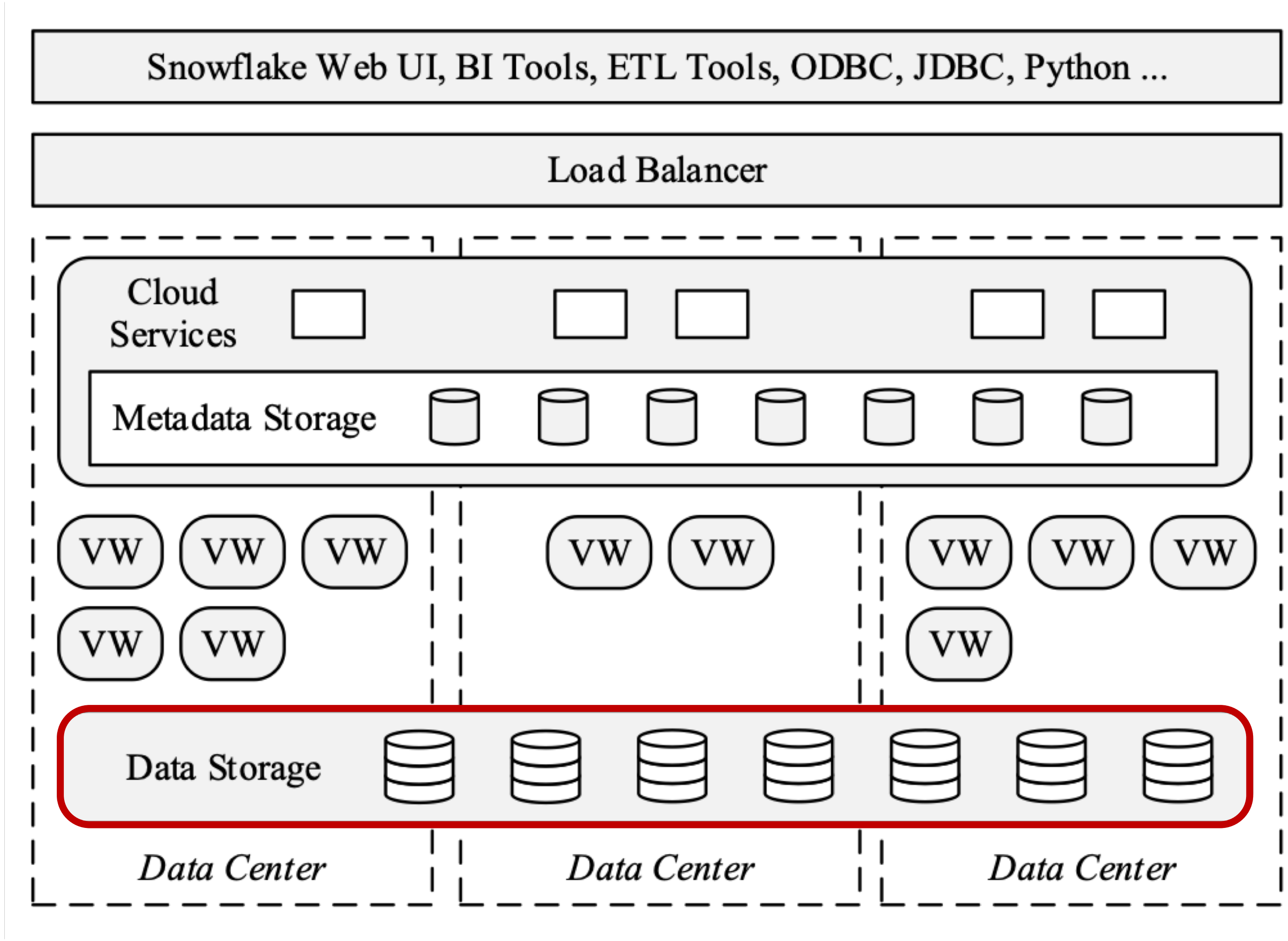
- Re-execute with failed node immediately replaced
- Re-execute with reduced number of nodes

Whole AZ failure

- Re-execute by re-provisioning a new VW

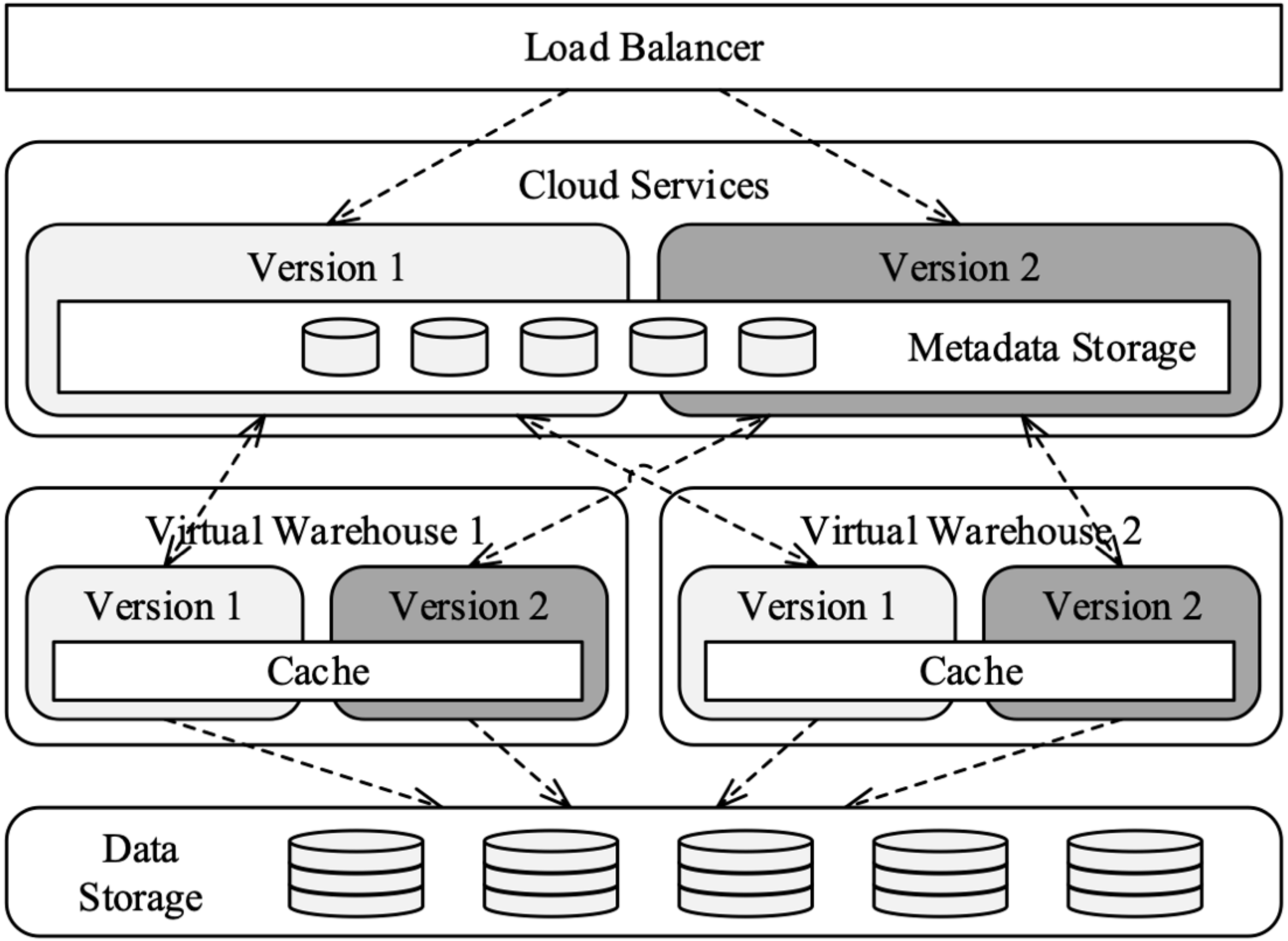
Hot-standby nodes

High Availability and Fault Tolerance



S3 is highly available and durable

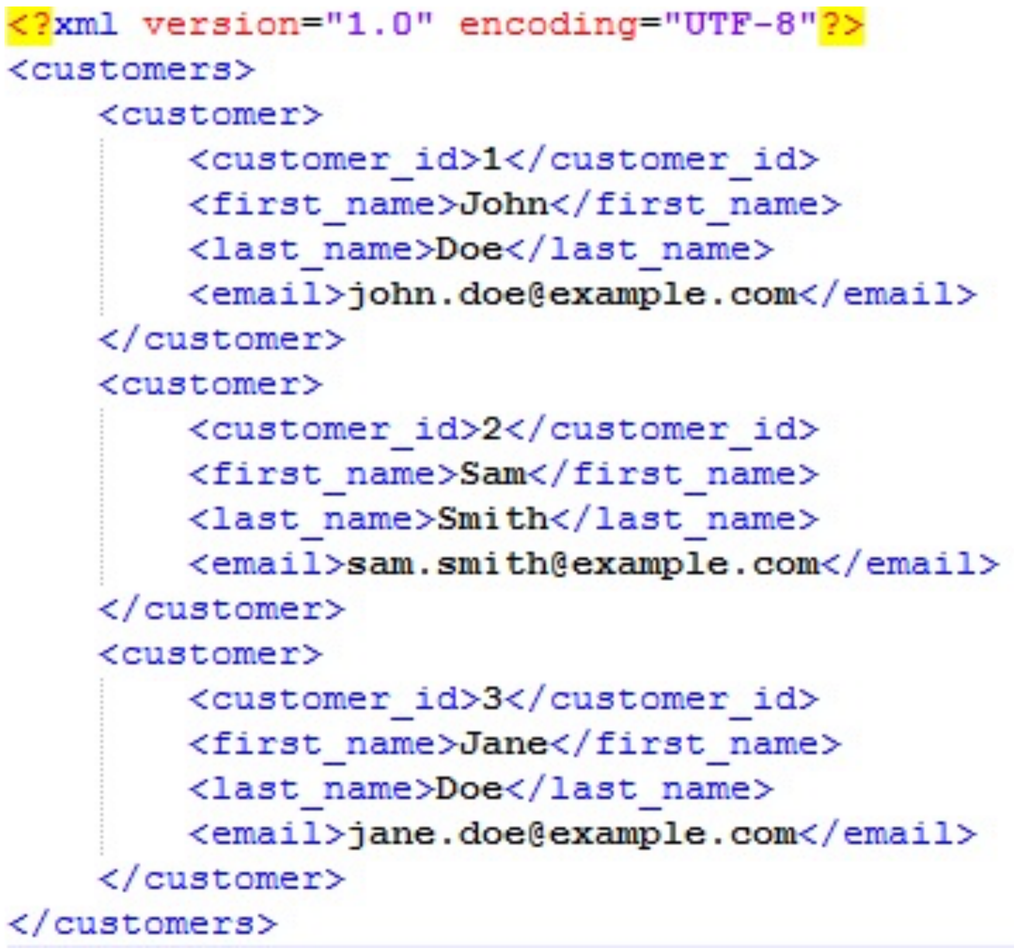
Online Upgrade



Deploy new versions of services and VWs

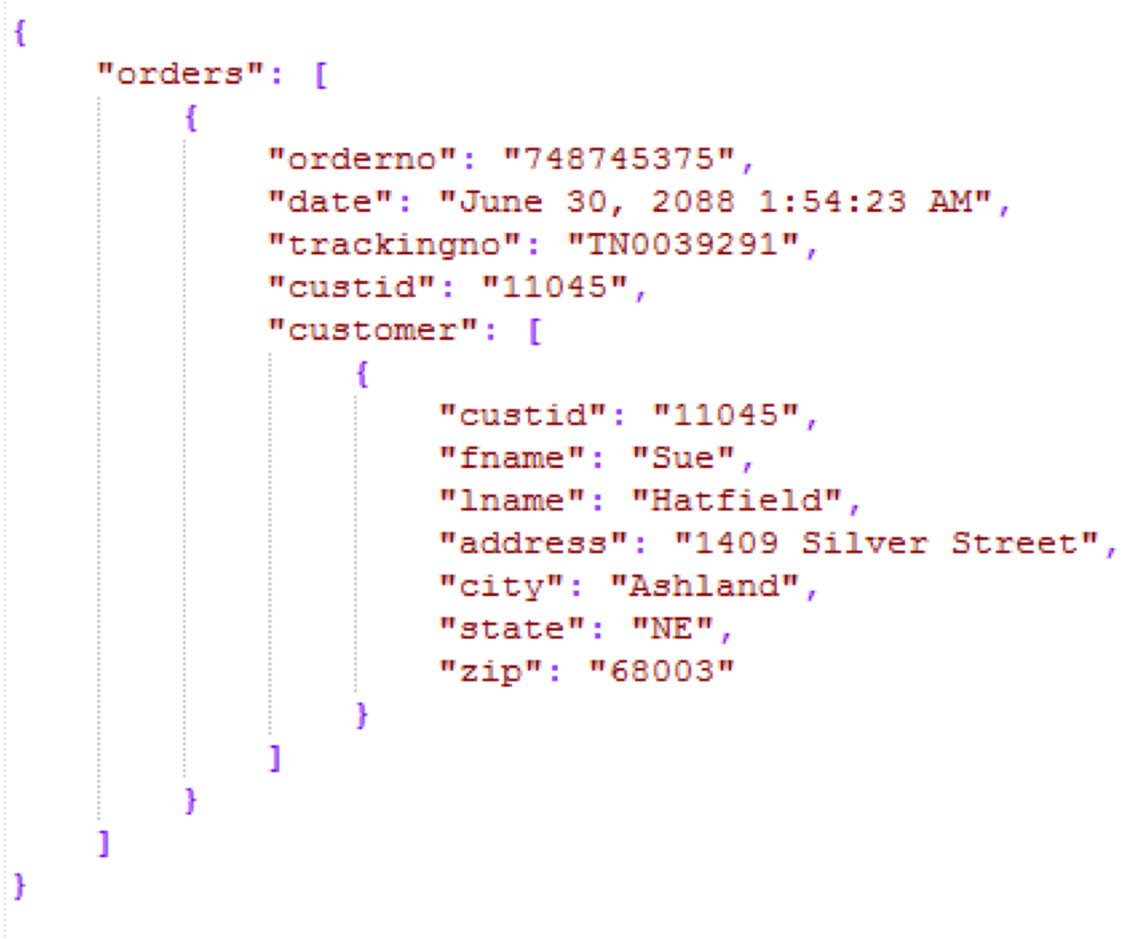
Semi-Structured Data

Extensible Markup Language (**XML**)



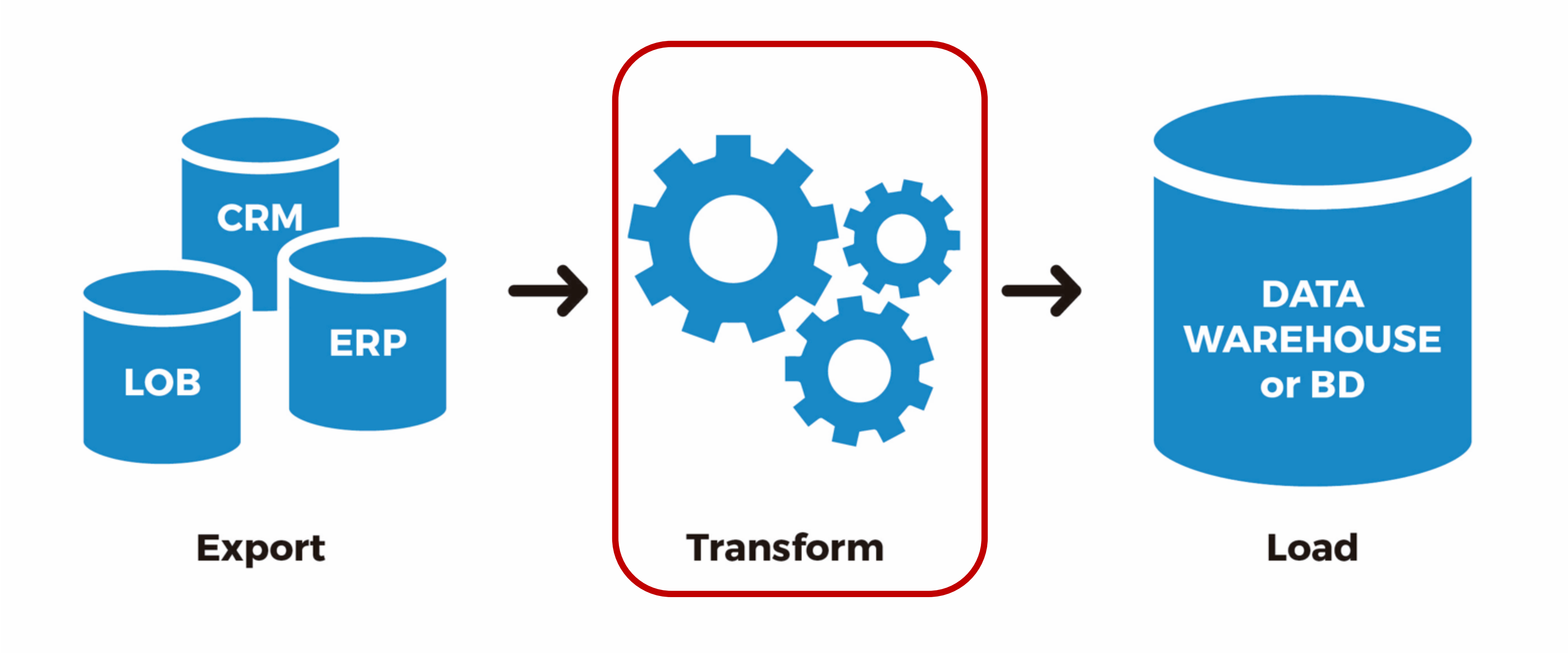
```
<?xml version="1.0" encoding="UTF-8"?>
<customers>
    <customer>
        <customer_id>1</customer_id>
        <first_name>John</first_name>
        <last_name>Doe</last_name>
        <email>john.doe@example.com</email>
    </customer>
    <customer>
        <customer_id>2</customer_id>
        <first_name>Sam</first_name>
        <last_name>Smith</last_name>
        <email>sam.smith@example.com</email>
    </customer>
    <customer>
        <customer_id>3</customer_id>
        <first_name>Jane</first_name>
        <last_name>Doe</last_name>
        <email>jane.doe@example.com</email>
    </customer>
</customers>
```

JavaScript Object Notation(**JSON**)



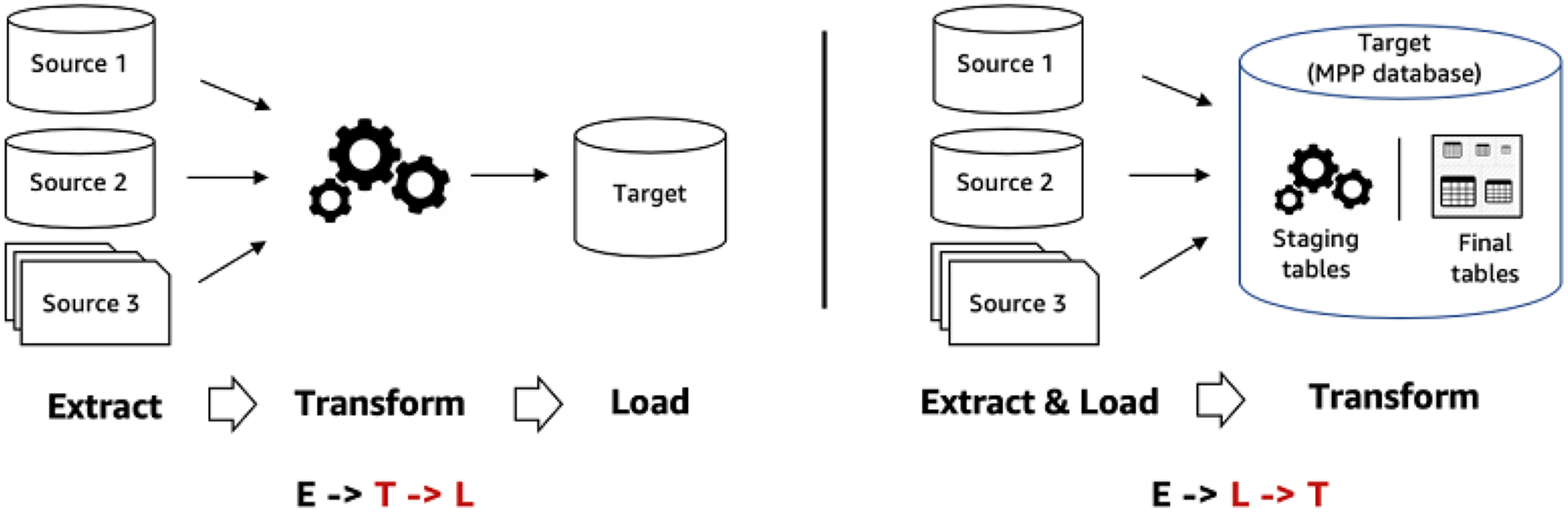
```
{
    "orders": [
        {
            "orderno": "748745375",
            "date": "June 30, 2088 1:54:23 AM",
            "trackingno": "TN0039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Sue",
                    "lname": "Hatfield",
                    "address": "1409 Silver Street",
                    "city": "Ashland",
                    "state": "NE",
                    "zip": "68003"
                }
            ]
        }
    ]
}
```

Extract-Transform-Load (ETL)



Transform (e.g., converting to column format) adds latency to the system

ETL vs. ELT



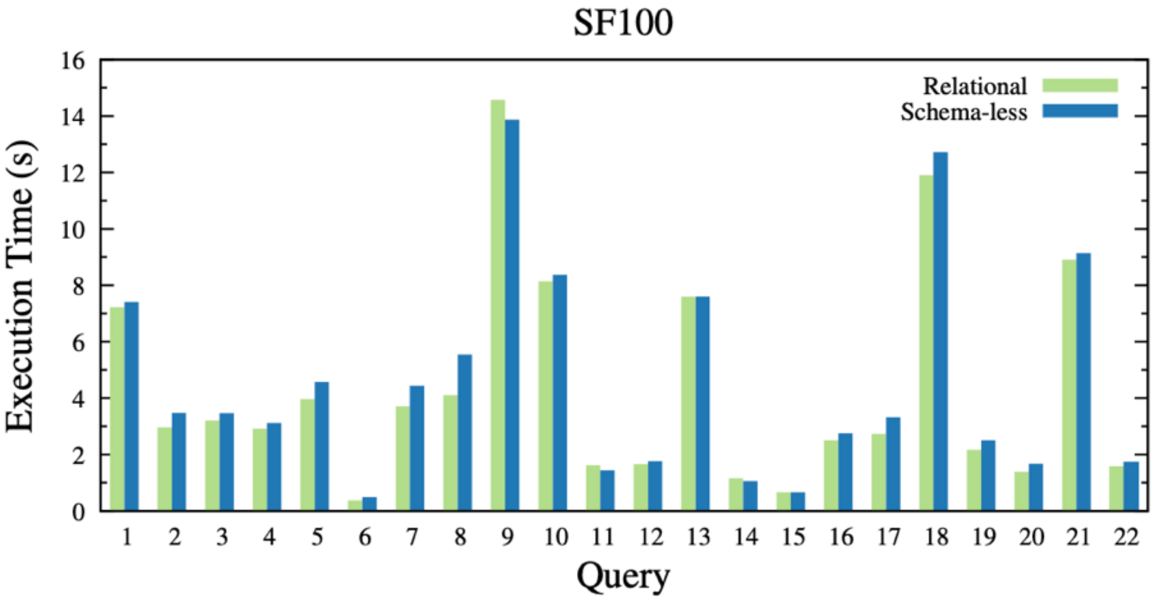
Picture from https://aws.amazon.com/blogs/big-data/etl-and-elt-design-patterns-for-lake-house-architecture-using-amazon-redshift-part-1/

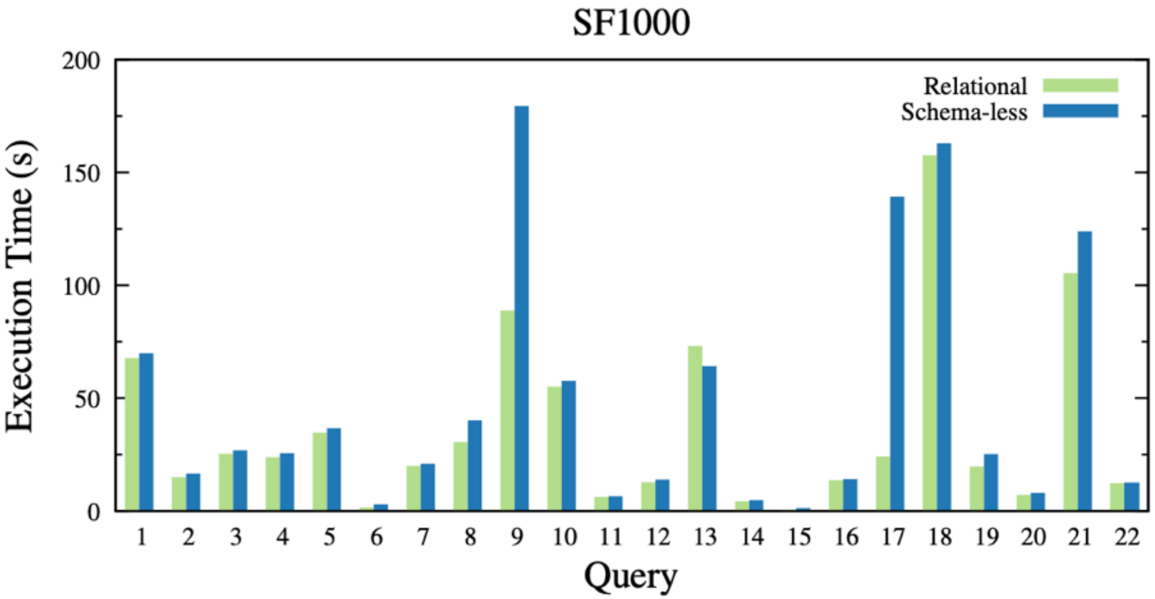
Optimization for Semi-Structured Data

Automatic type inference

Hybrid columnar format

- Frequently paths are detected, projected out, and stored in separate columns in table file (typed and compressed)
- Collect metadata on these columns for optimization (e.g., pruning)





Summary

Snowflake vs shared nothing

- Heterogeneous workload
- Membership changes

Snowflake vs. Redshift (Spectrum)

Snowflake vs. Athena

Snowflake vs. Presto/Hive/Vertica

Snowflake – Q/A

Storage system better than S3 (e.g., allow updates)

Row store for transaction processing?

Server-side cursor?

Min-max based pruning replacing indices?

Other systems similar to Snowflake?

Pay-as-you-go?

Push vs. pull?

Pruning requires sorting?

Snowflake autoscaling compute based on demand?

Group Discussion

How far away is Snowflake from the “optimal design” that you discussed last time?

- High-quality code compilers
- Athena with instances pre-running
- Hybrid instance store and S3; decide caching based on the workload
- Heterogeneous system that combines all the existing systems together

Can you come up with a nice way of combining cloud data warehousing (e.g., Snowflake) with cloud transaction processing (e.g., Aurora)?