CS 839: Topics in Database Management Systems
Lecture 3: Snowflake

Xiangyao Yu
9/13/2023
Presentation Signup

Too many people signed up for
  – Auto-scaling
  – Multi-cloud

Please either consider another topic or identify a paper/article in “Paper Signup” sheet **by next Monday**

Need a few more presenters for
  – Storage disaggregation for OLAP
  – Storage disaggregation for OLTP
  – Auto-tuning
  – Memory disaggregation

Please sign up for specific papers/articles
  – If you are presenting “disaggregation for OLAP”. Pick your paper **today**!
  – For other topics, try to pick your paper asap

No need to submit review for the lecture where you will present
## Group Discussion Summary

### Replay in compute node vs. storage node?

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Aurora</th>
<th>Log replay as a Service</th>
</tr>
</thead>
</table>
| **Advantages**   | Simpler design                     | • Less network traffic  
                    | • Higher availability (DB server as single point of failure)? | Replay traffic spread across storage nodes                  |
| **Disadvantages**| Replay traffic through a single primary instance | • Storage needs a CPU and is more expensive;  
                                            | • Log replay is a potential bottleneck  
                                            | • Consumes more resources overall | Log service can be a bottleneck |
Group Discussion Summary

Multi-master
- Must deal with conflicts across master nodes -> complex concurrency control (potentially happen inside the storage)
- Increased network traffic
- Deals with hotspot better
- More available since some of the masters can fail
- Potentially lower latency (e.g., skewed accesses)

Data partitioning
- Distributed transactions have higher latency due to coordination
- Hard to balance hotspot
- More scalable
The Snowflake Elastic Data Warehouse

Benoit Dageville, Thierry Crouze, Marcin Zukowki, Vadim Antonov, Artin Avanes, Jon Book, Jonathan Dlaybaugh, Daniel Engexotov, Martin Hentschel, Jiansheng Huang, Allison W. Lee, Ashish Medvada, Abdul Q. Munir, Steven Pelley, Peter Poveitch, Greg Rahn, Spyridon Tramafylakis, Philipp Unterbruner

Snowflake Computing

ABSTRACT

We live in the golden age of distributed computing. Public cloud platforms now offer virtually unlimited scale and storage resources on demand. At the same time, the Software-as-a-Service (Saas) model for enterprise class systems to users who previously could not afford such resources due to their cost and complexity. Also, traditional data warehousing systems are struggling to fit into this new environment. For one thing, they depend on complex ETL pipelines and physical tuning to be viable with the flexibility and business requirements of the cloud's low total cost of ownership and rapid provisioning model. For another thing, the common approach for traditional systems was to build an enterprise-class data warehousing solution for the cloud. The result is the Snowflake Elastic Data Warehouse, or "Snowflake" for short. Snowflake is a multi-tenant, transactional service, highly scalable andShot for systems with full SQL support and built-in connectors for semi-structured and relationless data. The system is offered as a pay-per-use service in the AWS cloud. Users upload their data to the cloud and can immediately manage and query it using familiar tools and interfaces. Implementation begins in early 2017 and Snowflake has been generally available since December 2014. Snowflake is powering a growing number of small and large organizations alike. The system runs several million queries per day over multiple petabytes of data.

In this paper, we describe the design of Snowflake and its novel multi-tenant, shared-data architecture. The paper highlights areas of the key features of Snowflake: extreme elasticity and scalability, schema-less and schema-aware data, fine-grained, and column-level security. It concludes with lessons learned and an outlook on ongoing work.

Categories and Subject Descriptors

Information systems [Data management systems]: Database management systems

Keywords

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

1. INTRODUCTION

The advent of the cloud enables a move away from software delivery and operation on local servers, and toward shared computing resources in the cloud. A range of traditional software platforms and cloud services has been created for this new model. For example, Amazon Elastic Compute Cloud (EC2) provides a virtual machine to which users can upload software and data that can be accessed via a web interface. Cloud services like Amazon Simple Storage Service (S3) enable users to store and retrieve vast amounts of binary data. Snowflake is the latest venture in this class of cloud services and aims to complement existing cloud platforms with data warehousing capabilities.

Snowflake is a pay-per-use data warehousing service. It allows users to efficiently store and query voluminous datasets, and it is designed to be an easy-to-use platform for running interactive queries against these datasets. Snowflake is available as a pay-per-use service hosted in the Amazon Web Services environment. The architecture of Snowflake is designed with unique characteristics that differentiate it from traditional data warehousing systems. These characteristics include the use of a columnar storage format, a highly parallel execution model, and the ability to scale out horizontally to meet the needs of larger datasets.

In this paper, we describe the design of Snowflake and its novel multi-tenant, shared-data architecture. The paper highlights areas of the key features of Snowflake: extreme elasticity and scalability, schema-less and schema-aware data, fine-grained, and column-level security. It concludes with lessons learned and an outlook on ongoing work.
Shared-nothing vs. Storage-disaggregation

Shared-nothing
- Fixed and limited hardware resources

Storage disaggregation
- Virtually infinite computation & storage, Pay-as-you-go price model
Shared Nothing – Advantages

Scalability: horizontal scaling
• Scales well for star-schema queries
Heterogeneous workload

- Static resource provisioning cannot adjust to heterogeneous workloads
- Must pay for entire cluster even when no queries exist
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: similar to the fault tolerance problem
Shared Nothing – Disadvantages

Heterogeneous workload
Membership changes
Online upgrade
  • Similar to membership change but affect all nodes
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
– Determine the VW size based on performance and cost requirements
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
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
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 as some other data warehousing systems)
– Min-max based pruning: store min and max values for a data block
High Availability and Fault Tolerance

Snowflake Web UI, BI Tools, ETL Tools, ODBC, JDBC, Python ...

Load Balancer

Cloud Services

Metadata Storage

Data Storage

Data Center

Data Center

Data Center

Stateless services
High Availability and Fault Tolerance

- Replicated metadata
  (FoundationDB)

- Cloud Services

- Load Balancer

- Metadata Storage

- Data Storage

- Data Center

- Snowflake Web UI, BI Tools, ETL Tools, ODBC, JDBC, Python...
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
S3 is highly available and durable
Online Upgrade

Deploy new versions of services and VWs

Previous version terminates after active queries finish
Web User Interface (Serverless)
Extract-Transform-Load (ETL)

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

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)
Snowflake – Q/A

No indexing used? min-max filtering

Node failure causes the entire query to fail? More graceful failures like in Spark?

Push-based execution?

Transaction support in Snowflake?

How to deal with large intermediate data?

Performance of Snowflake vs. a shared-nothing deployment?
Discussion Question

For a multi-cloud analytical database where the data is stored across AWS, Azure, and GCP, is Snowflake architecture a good fit? What architecture would you choose in this scenario?

Snowflake is promoting the idea of data marketplace, where Snowflake users can share/trade their data and queries (think of App Store). What new applications can this enable in your opinion?

Please submit your discussion to hotcrp as a new submission by the end of Thursday (9/14)
- Title starts with “[Discussion L3]”
- Set authors properly
Before Next Lecture

Review one of “disaggregation for analytical processing” papers