CS 764: Topics in Database Management Systems
Lecture 26: Snowflake

Xiangyao Yu
12/6/2021
Announcements

DAWN workshop schedule

– Online workshop using the lecture zoom link
– Reserve a presentation slot using the following google sheet
  https://docs.google.com/spreadsheets/d/1BkO3ZqxNXxHRkl-XTnHmvQ1z66sS4LUVvIjiHS6HIJI/edit?usp=sharing
– Each group has a 10 min slot: **8 min presentation + 2 min Q/A**
– Live presentation preferred, but recording is also ok

Project report (DDL: Dec. 18)

– Sample reports available from the course website
– 5–7 pages sufficient. Content is more important than length.
– **Submit to the hotcrp website** (like the proposal)
ABSTRACT

The SaaS is 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. Also, traditional data warehousing systems are struggling to keep up with the growth of data and new requirements stemming from the cloud’s new types of data-intensive workloads and speed-driven workloads.

The challenge that we set out to address was to build an enterprise-ready data warehousing solution for the cloud. The result is the Snowflake Elastic Data Warehouse, or “Snowflake” for short. Snowflake is a multi-tenant, transactional, shareable scalable cloud data system with full SQL support and built-in horizontal scale for semi-structured and columnar data. The system is offered as a pay-as-you-go service in the Amazon cloud. Users upload their data to the cloud and can immediately manage and query it using familiar tools and interfaces. The implementation began in late 2012 and Snowflake has been publicly available since late 2013.

Categories and Subject Descriptors

Information systems [Data management systems]: Database management system engine

Keywords

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

1. INTRODUCTION

The advent of the cloud made it possible for large amounts of data to be easily and cost-effectively stored and processed. This has given rise to a new generation of data warehousing systems that are designed to leverage the cloud’s elasticity. For these systems, the primary focus has been on complexity and scalability, with the goal of providing a true enterprise-class data warehousing solution.

Today’s Paper

The Snowflake Elastic Data Warehouse


Snowflake Computing

Editor’s Note: This is part of the SIGMOD 2016 proceedings. SIGMOD is the premier international conference in the field of database systems. The conference brings together researchers, practitioners, and students to discuss the latest developments in database technology, methodology, and applications. SIGMOD 2016 was held in San Francisco, CA, USA, from May 24-26, 2016. Full details on the conference can be found at http://sigmod2016.acm.org.
On-Premises vs. Cloud

**On-premises**
- Fixed and limited hardware resources
- **Shared-nothing** architecture

**Cloud**
- Virtually infinite computation & storage, Pay-as-you-go price model
- **Disaggregation** architecture
Shared Nothing – Advantages

Scalability: horizontal scaling
• Scales well for star-schema queries
Shared Nothing – Disadvantages

Heterogeneous workload

- Static resource provisioning cannot adjust to heterogeneous workloads

Workload A

- More CPU intensive

Workload B

- Less CPU intensive
Shared Nothing – Disadvantages

Heterogeneous workload
Membership changes
  - Add a node: data redistribution
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
  - Authentication and Access Control
  - Cloud Services
    - Infrastructure Manager
    - Optimizer
    - Transaction Manager
    - Security
    - Metadata Storage
  - Virtual Warehouse
    - Cache
  - Data Storage
- 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
- More CPU intensive
- Large VW

Workload B
- Less CPU intensive
- Small VW
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

Stateless services

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

Load Balancer

Cloud Services

Metadata Storage

Data Storage

Data Center  Data Center  Data Center
High Availability and Fault Tolerance

Replicated metadata
(FoundationDB)
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

Previous version terminates after active queries finish
Semi-Structured Data

**Extensible Markup Language (XML)**

```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)**

```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"
        }
      ]
    }
  ]
}```
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)
Q: What are the limitations of Snowflake’s design?
A Follow-up Paper

Limitations of current Snowflake design and potential research directions

– Decoupling of compute and ephemeral storage
– Deep storage hierarchy
– Pricing at sub-second timescales
Distributed Ephemeral Storage

Intermediate data is short-lived
- Need low-latency and high throughput
- Strong durability not needed
- Caching of intermediate data vs. persistent data
- Query scheduling: locality-aware task + work stealing
Lakehouse Architecture

Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics

Michael Armbrust1, Ali Ghodsi2,3, Reynold Xin1, Matei Zaharia1,3

1Databricks, 2UC Berkeley, 3Stanford University

Abstract
This paper argues that the data warehouse architecture as we know it today will wither in the coming years and be replaced by a new architectural pattern, the Lakehouse, which will (i) be based on open access data formats, such as Apache Parquet, (ii) have first-class support for machine learning and data science, and (iii) offer state-of-the-art performance. Lakehouses can help address several major challenges with data warehouses, including data staleness, scalability, total cost of ownership, data lock-in, and limited use-case support. We discuss how the industry is already moving toward Lakehouses and how this shift may affect work in data management. We also report results from a Lakehouse system using Parquet that is competitive with popular cloud data warehouses on TPC-DS.

1 Introduction
This paper argues that the data warehouse architecture as we know it today will wither in the coming years and be replaced by a new architectural pattern, which we refer to as the Lakehouse, characterized by (i) open access data formats, such as Apache Parquet and ORC, (ii) first-class support for machine learning and data science workloads, and (iii) state-of-the-art performance.

The history of data warehousing started with helping business leaders get analytical insights by collecting data from operational databases into centralized warehouses, which could be used for decision support and business intelligence (BI). Data in those warehouses would be written with schema-on-write, which ensured that the data model was optimized for downstream BI consumption. We refer to this as the first-generation data analytics platform.

A decade ago, the first-generation systems started to face several challenges. First, they typically coupled compute and storage into an on-premises appliance. This forced enterprises to provision and pay for the peak of user load and data volume under management, which became very costly as datasets grew. Second, not only were datasets growing rapidly, but more and more datasets were completely sanitized, e.g., video, audio, and text documents, which data warehouses could not store and query at all.

To solve these problems, the second-generation data analytics platforms started offloading all the raw data into data lakes at low-cost storage systems with a file API that hold data in generic and usually open file formats, such as Apache Parquet and ORC [8, 9]. This approach started with the Apache Hadoop movement [5], using the Hadoop File System (HDFS) for cheap storage. The data lake was a schema-on-read architecture that enabled the agility of storing any data at low cost, but on the other hand, posed the problem of data quality and governance downstream. In this architecture, a small subset of data in the lake would later be loaded into a decision support and BI applications. The use of open formats also made data lake data directly accessible to a wide range of other analytics engines, such as machine-learning systems [27, 42].

From 2015 onwards, cloud data lakes, such as S3, ADLS, and GCS, started replacing HDFS. They have superior durability (often >9999), geo-replication, and, most importantly, extremely low cost with the possibility of automatic, even cheaper,archival storage, e.g., 48% cheaper. The rise of the architecture is largely the same in the cloud as in the second-generation systems, with a downstream data warehouse such as Redshift or Snowflake. This two-tier data lake + warehouse architecture is now dominant in the industry in our experience (used at virtually all Fortune 100 enterprises).

This brings us to the challenges with current data architectures. While the cloud data lake and warehouse architecture is assembly cheap due to separate storage (e.g., S3) and compute (e.g., Redshift), a two-tier architecture is highly complex for users. In the first-generation platform, all data is ETL-ed from operational data systems directly into a warehouse. In today’s architecture, data is first ETL-ed into data lakes, and then again ETL-ed into warehouses, creating complexity, delays, and new failure modes. Moreover, enterprise use cases now include advanced analytics such as machine learning, for which neither data lakes nor warehouses are ideal. Specifically, today’s data architectures commonly suffer from four problems:

Reliability: Keeping the data lake and warehouse consistent is difficult and costly. Continuous engineering is required to S3 data between the two systems and making it available to high-performance decision support and BI. Each ETL step also risks sourcing failures or introducing bugs that reduce data quality, e.g., due to subtle differences between the data lake and warehouse engines.

Data staleness: The data in the warehouse is stale compared to that of the data lake, with new data frequently taking days to load. This is a step back compared to the first generation of analytics systems, where new operational data was immediately available for queries. According to a survey by Dimensional Research and BizTrends, 88% of analysts use out-of-date data and 62% report waiting on engineering resources numerous times per month [47].

Limited support for advanced analytics: Businesses want to ask predictive questions using their warehousing data, e.g., “Which customers should I offer discount to?” Despite much research on the frontiers of ML and data management, none of the leading machine-learning systems, such as TensorFlow, PyTorch, and XGBoost, work well on top of warehouses. Unlike ML systems, which extract a small amount of data, these systems need to process large datasets using complex in-memory code. Reading this data via GCS/ADLS is inefficient, and there is no way to directly access the internal...
Data Warehouse vs. Data Lake

Lakehouse = Data warehouse + data lake
Snowflake – Q/A

How does Snowflake support ACID transactions?

Snowflake vs. Databricks, which is more promising?

Performance of Snowflake vs. AWS, Azure, and Cloudera?

How to guarantee security? Previous papers didn’t discuss it.

Is metadata management a bottleneck?

What is the future trend? Shared-nothing?

Snowflake on storage service other than S3?
Before Next Lecture

Submit review for

– Yifei Yang, et al., FlexPushdownDB: Hybrid Pushdown and Caching in a Cloud DBMS. VLDB, 2021