Big Data Paper Summary

October 28th, 2020

Caleb Rogers

Papers and Video Titles

Hive – A Petabyte Scale Data Warehouse Using Hadoop

Choosing A Cloud DBMS: Architectures and Tradeoffs

One Size Fits All – Whose Time Has Come and Gone (2005)

References (pictures)

Facebook. (2020, October 09). Retrieved October 29, 2020, from https://simple.wikipedia.org/wiki/Facebook

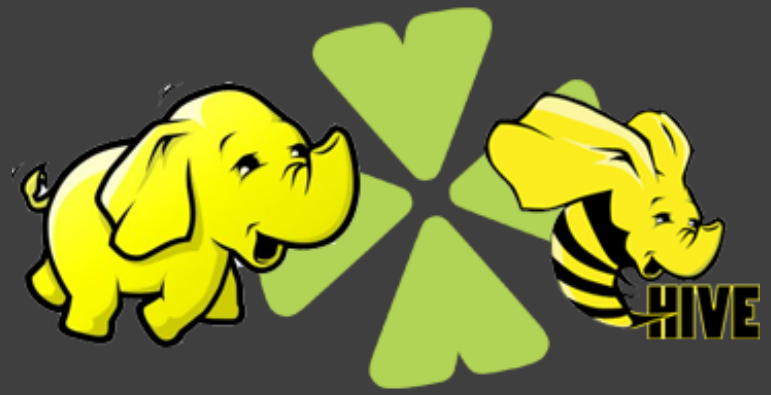
Madan, N. (2020, August 28). 3 Ways Big Data Can Influence Decision-Making for Organizations. Retrieved October 29, 2020, from https://www.simplilearn.com/how-big-data-can-influence-decision-making-article

Originally published by Cabot Technology Solution on. (2020, October 28). 5 Top Cloud Databases that Works Wonders! Retrieved October 29, 2020, from https://hackernoon.com/5-top-cloud-databases-that-works-wonders-7e628810e3ac

QuestILearn. (2018, July 23). Hive in Hadoop with Architecture. Retrieved October 28, 2020, from https://www.questilearn.org/hive-in-hadoop-with-architecture/

Hive - A Petabyte Scale Data Warehouse Using Hadoop

Hive is a query engine that was developed in response to the industries developing demands for a flexible infrastructure that supports the drastic increase of data, maintains practical costs, and addresses the progressive diversity of applications and users.



- **Problem:** traditional warehousing is emerging as prohibitively expensive due to the rapidly growing size of data sets being collected and analyzed for ad hoc analysis and business intelligence applications.
- **Solution:** Hadoop was developed as an open-source map-reduce implementation at a petabyte scale, which provided needed scalability using commodity hardware.
- **Problem:** Hadoop uses a map-reduce programming model which presents difficulty due to being low-level, requiring end-users to write custom programs at lengthy durations despite often being simple tasks.
- **Solution:** Hive was built on top of Hadoop to address the lack of expressiveness in query capabilities, similar to popular query languages like SQL. The implemented HiveQL compiled queries that allowed custom map-reduce scripts that are executed by Hadoop.

Hive Implementation

Hive and Hadoop was developed to be used extensivly by FaceBook.

- Facebook's data warehouse has 700TB of data spread across tens of thousands of tables and stores.
- Hive is used for reporting and adhoc analyses by more than 200 users per month.
- On daily average, more than 7500 adhoc jobs are submitted to the data warehouse.
- On daily average, more than 75TB of data is compressed and processed.

As FaceBook's network grows, there is a corresponding continous growth in data, and all this data presents issues for Hive.

Problem: Adhoc jobs are unpredictable with times of heavy usage and can cause significant operational challenges. Hadoop resouce scheduling is liable to degraded performance, caused by an abundence resources being shared between adhoc users and reporting users.

Solution: Reporting and adhoc queries require their maintaing their own clusters. Facebook is vastly used across the world and must process a tremendous amount of data. Faster processing and larger storage options will improve as technology progresses. Facebooks high usage also demands data discoery tools to be efficient with the addition of tens of thousands of tables. The ability of Hive and Hadoop to scale thousands of commodity nodes and process the tremendous amount data is monumental.

Hive Analysis

- Data Model – Hive uses traditional databases concepts to store data in tables, which consists of rows and columns.
- Type System – Each column contains a data type for its subsequent values. Hive supports primitive types and complex types.
  - Primitive Types: integers, floats, doubles, and strings
  - Complex Types: maps, lists, and structs
- HiveQL – Hive's query language that uses SQL features with extensions.
  - Uses traditional SQL features including subqueries, joins, aggregations, select-from clauses, and many more functions.
  - Limitations exist such as HiveQL does not support how traditional inserts are done. Limitations like this are not a problem due to the implementations of extensions that are used to bypass them.
- Data Storage – Hadoop Distributed File System (HDFS) is used to store data using the following primary data units and their mappings.
  - Tables: Are stored in a directory within HDFS.
  - Partitions: Are stored in a sub-directory within a directory within HDFS.
  - Buckets: If the table is partitioned, the bucket file is stored within the partitoned. If the table is non-partitioned, then the bucket file is stored within the directory.
- SerDe – The default SerDe for Hive is LazySerDe, which uses deserialization.
- File Formats – Hive is open-source and does not impose any restrictions on users to the type of their input formats.
- System Architecture & Components – Hive's main building blocks are it's metastore, driver, query compiler, execution engine, HiveServer; and can include client components like Command Line Interface, web UI, JDBC/ODBC driver; and can also include extensibility interfaces like SerDe, ObjectInspector, UDF, and UDAF.

IV. System Architecture And Components

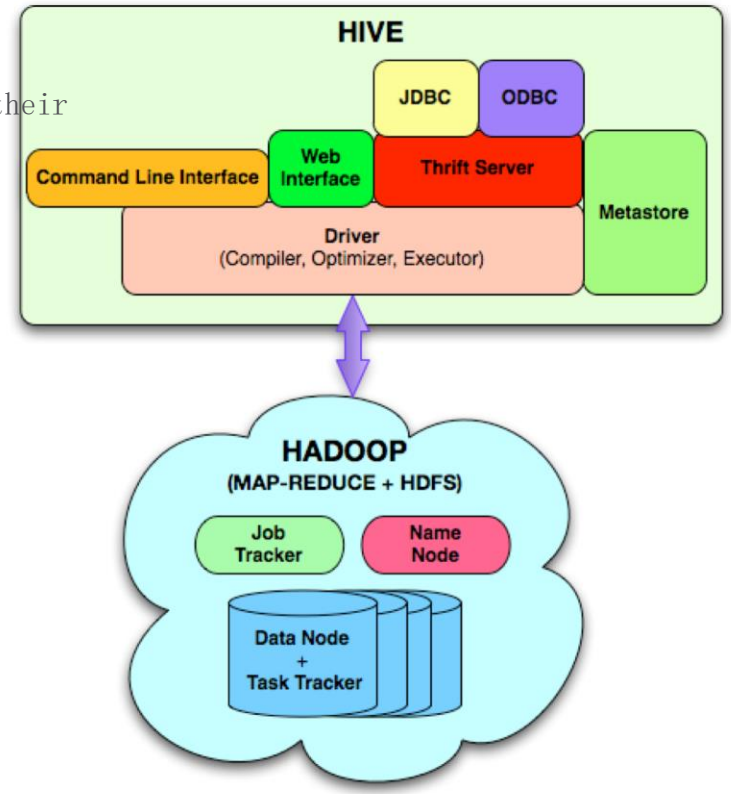


Fig. 1: Hive System Architecture

Choosing A Cloud DBMS: Architectures and Tradeoffs

This study analyzes the impending switch for DBMSs to run their analytic applications on the cloud. Different DBMSs and cloud configurations result in trade-offs considering query restrictions initialization times, query performance, costs, data compatibility with other systems, and scalability. Experimenting with external storage, query executors, and DBMS-as-a-service offerings concluded the need to prioritize shared, low-cost block storage architectures.

Cloud DBMS Study Implementation

Six popular production OLAP DBMS were implemented in this study: Athena, Hive, Presto, Redshift, Reshift Spectrum, and Vertica. These systems have similarities that group them in three areas. These areas include external storage, query executors, and DBMS-as-a-service offerings.

- External Storage – Rather than using local storage, external storage is often used with the impending move to cloud storage. The object stores these systems use are Elastic Block Storage (EBS) or Instance Store (InS), and Simple Service (S3).
- Query Executors – These are paid per use, therefore should be running as little as possible to minimize costs. This makes scalibility crucial. Horizontal scaling becomes important as more is gained from a system when using horizontal scaling compared to vertical scaling, which tends to be less beneficial.
- DBMS-as-a-service offerings – Costs are based on the amount of data scanned. Unlike query executors which offer users to enter custom programs allowing lots of flexibility, DBMS-as-a-service offering little flexibility due to being presented as a low-level service.

Table 1: Tested Systems and Supported Storage Architectures

Category	DBMS	Database Storage System	Temp Node Storage System and Usage
Proprietary database-as-a-service offerings	Redshift	Local storage (snapshotted to S3)	Local storage for spill- to- disk
	Spectrum (Redshift feature)	Remote object store (S3)	Local storage for spill to disk and possibly remote data
	Athena	Remote object store (S3)	Unknown, but no cache effects observed
Query engines	Presto	Remote object store (S3 or HDFS)	Node mounted storage volumes (EBS or local) for spill-to-disk
	Hive	Remote object store (S3 or HDFS)	Node mounted storage volumes (EBS or local) for spill-to-disk
Cloud provider agnostic OLAP DBMS	Vertica	Node mounted storage volumes (EBS or local)	Node mounted storage volumes (EBS or local) for spill-to-disk
	Eon (Vertica mode)	Remote object store (S3 or HDFS)	Node mounted storage volumes (EBS or local) for spill-to-disk and caching S3 data

Cloud DBMS Study Analysis

The following are the resulting informing gained from this study. Analyzing these fields help DBMS users determine the best system required for their database needs.

- Query Restrictions – The TPC-H benchmark was used to consistently test the OLAP DBMSs. Spectrum nor Athena could run full TPC-H query suites; thus they were selectively plotted during the study.
- Initialization Time - Systems using local storage like Redshift look the experienced the longest initialization times, while serverless offerings like Athena experienced the quickest initialization times.
- Query Performance - There is little difference between most of the systems considering query performance, and because of the little resulting diversity, this test enlightens us that substantial cost advantages exceed performance disadvantages.
- Cost - When considering cost of shared storage, S3 greatly outperforms EBS, with proportional performance insignificant

  in proportion to cost. Local storage cost models focus more on data scan and node uptime, rather than cost vs performance.
- Data Compatibility - Without compatibility, ETL (extract, transform, and load) costs become extensive, and AWS

  proprietary and shared-nothing systems suffer from this fault. The most compatible systems experienced used general

  data formats, systems like Hive, Presto, and Vertica.
- Scalability - horizontal scaling tends to be advantageous resulting in performance benefits, while opposed to vertical

Comparison of Hive and Cloud DBMS

Pros of Hive:

- Based on Hadoop, Hive is on a petabyte scale, allowing processing more data to keep up with the ever-growing demand.
- Hive offers HiveQL, which allows end-users to enter custom map-reduce scripts.
  This greatly decreases programming times.

Cons of Hive:

- Update and delete statements are not in HiveQL, resulting in extra effort put towards Hive's extensions to do these processes.

Pros of Cloud DBMS:

- Cloud is necessary for the increasing demand of storage of data. Cloud is also more cost-effective and has more flexibility to the diversity of applications and users.

Cons of Cloud DBMS:

- Security becomes the biggest issue due to the lack of control users have when their data is processed in the cloud.

One Size Fits All - An Idea Whose Time Has Come And Gone (2005)

Context: The Relational Database Management System (RDBMS), introduced in the 1970s, became popularized in the '80s and was made to be very abstract so that it could answer any question a user had. RDBMS introduced many features still used today, like referential integrity and triggers, which laid the groundwork for the DBMSs used today.

2005 Paper: In their paper, they discussed that RDBMS still worked but it could no longer "fit all", as problems were arising which included the inability to stream applications. Stonebraker and Cetintemel, along with collogues he acknowledged, started research into C-Store (column store), as well as talked about text.

A Decade Later: By 2015, the introduction of Data Warehouse markets, OLTP markets, NoSQL markets, Complex Analytics, Streaming markets, and Graph analytics market has outdated RDBMS. There is a large diversity of engines that all orient towards their own specializations and traditional row stores are now obsolete. Rather than being the universal solution it was, RDBMS have become the "One Size Fits None".

Moving Forward: Stonebraker talks about the excitement of being a database researcher now, as databases are no longer stuck to a singular relational database. We are experiencing many new implementations including nonvolitale random-access memory, transactional processing, complex analytics with arrays, and more! Stonebraker talks of "the Innovators Dilema" which expects the loss of market share trying to adapt. Other than the conflict between SAP and Oracle, Stonebreaker hopes these databses can keep their user interface, switch out engines as needed, and databases will just keep getting better.

Advantages and Disadvantages of Hive in the Context of Cloud DBMSs and the Stonebraker Talk

Advantages: Hive encomposes the improvements of databases made over the years. Learning from traditional data warehousing and implementing open-source map-reduce implementation, Hive uses the resources it has including benefitting from a petabyte scale. Hive uses the advantages of cloud to be a diverse, accessible query executor. RDBMS was made used to try to solve all database needs, while Hive understands the need for specialization and the use of many different systems to accomplish different tasks based on user's needs. Hive also uses the the cloud's avalible storage and cost-effective options.

Disadvantages: While Hive is diverse, it also is specific and does not capitalize on the overall effectiveness RDBMS are able to offer. Considering cloud, Hive suffered in regard to removing caches. Hive was insufficient due to "cold start" configuration, causing cache removal for the DBMS and the OS after each run.