Hadoop vs two DBMSs, Hive to Assist Hadoop, but….Stonebraker says – “One size will not fit all” and even he doesn’t know “who will win”

Kathy Coomes – March 5, 2017

- “A Comparison of Approaches to Large-Scale Data Analysis” – Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker; SIGMOD ’09, June 29 – July 2, 2009; Copyright 2009 ACM 978-1-60558-551-2/09/06.
- “Hive – A Petabyte Scale Data Warehouse Husing Hadoop” – Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy; ICDE Conference 2010; Copyright 2010 IEEE 978-1-4244-5446-4/10/10.
- Video talk (2015) by Michael Stonebraker on their paper “One Size Fits All – An Idea Whose Time Has Come and Gone (2005)” – Michael Stonebraker and Ugur Cetintemel; video url - kdb.snu.ac.kr/data/stonebraker_talk.mp4

Comparison of Hadoop (MapReduce), Vertica and DBMS-X (both parallel SQL DBMSs),

- Goal – compare and understand the different approaches of 3 systems to large scale data analysis
- Hadoop – permits data to be in any arbitrary format
  - simple to set up;
  - programmers need to write many programs;
  - considered quite flexible if not shared; if shared another programmer has to interpret others programs first;
  - no tools, only trial and error for tuning;
  - can recover from faults in the middle of query execution;
  - to add columns to data set, have to modify existing code and retest all other programming to ensure they still worked;
  - comes with rudimentary web interface, anything additional needs to be developed in house;
- DBMS (Vertica and DBMS-X) – data is required to conform to a well-defined schema
  - easy to install;
  - difficult to configure;
  - comes with some tools to aid in tuning;
  - no failure recovery, need to start over;
  - no code modification needed when adding columns;
  - tons of existing tools, applications and third party extensions

Implementation of the Comparison Paper

- Reviewed the two alternative system types: MapReduce and Parallel DBMSs
- Discussed the architectural trade-offs of the two systems: Schema, Indexing, Programming Model, Data Distribution, Execution Strategy, Flexibility, and Fault Tolerance;
- Performed Benchmark Tests on a variety of tasks deployed on a 100-node cluster
  - Described how each system was dealt with to make the experiments uniform
    - Hadoop version 0.19.0 running on Java 1.6.0
    - DBMS-X latest release
    - Vertica (version not stated)
- Gave the times to setup and tune the software for each task and the times to load the test data for each system
- Results stated and shown in graphs
  - Hadoop was faster at tuning and loading
  - DBMSs were faster at executing and required less coding to implement each task
- Discussed their ideas why there is such a dramatic difference in performance

My analysis of the Comparison Paper and its ideas and implementation

- Hadoop is easy to startup and run, and is less expensive since it is open source, but requires quite a lot of programs to be written. Once written, they seem to be unchangeable without going back and ensuring that all previous programs will also work with the changes.

- The writers seem to be a bit more in favor of Hadoop over DBMSs. They didn't even specify the versions of the DBMSs that they were using.

- Vertica and DBMS-X may take a bit longer to start up and run, but allow for more configuration, and better optimization.

- They do test all three systems equally and give the results with each system doing better at some things and not so good at others. A decision between the three would be based on the data to be used and how it was to be used.

Hive and how it will Assist Hadoop

- Goal – show that Hive using its HiveQL (with its many SQL-like qualities) has made Hadoop easier to use.
- Hadoop is a MapReduce system already used at petabyte scale that requires developers to write custom programs that are hard to maintain and reuse.
- Hive is an open source data warehousing solution that has been built on top of Hadoop –
  - Brings tables, rows, columns, partitions, and a subset of SQL to Hadoop.
  - Supports all major primitive data types and complex data types.
  - Tables accept data from other sources and from legacy data without transforming the data saving time for large data sets by providing a jar.
  - HiveQL is a subset of SQL with subqueries, joins, Cartesian products, group by, aggregations, etc.
  - Enables anyone familiar with SQL to begin querying the system immediately.
- Limitations –
  - No insert, update, or delete.
  - Joins have to be specified using ANSI join syntax and only support equal predicates.

Implementation of Showing How Hive Assists Hadoop

- Written by a Facebook Data Infrastructure Team more as an explanation why they switched from a RDBMS to Hadoop and then built Hive on top of it.
- Hadoop was already open source, already used at petabyte scale, and could complete jobs that took more than a day within a few hours. So they built Hive to use with it in January 2007. It is also available as open source.
- They gave a great description of Hive:
  - Data model and type systems,
  - Query language – HiveQL and how it is like SQL,
  - How data is stored in the underlying distributed file system (HDFS - Hadoop file system),
  - It's default SerDe implementation – LazySerDe and it's serialization/deserialization of delimited data,
  - How Hadoop files can be stored in different file formats,
  - How the Metastore acts as a system catalog and is used by the query compiler to generate the execution plan much like a SQL database.
- They described how Hadoop and Hive are widely used at Facebook, how others are working on systems such as this, and how Hive is still a work in progress.

My analysis of the Hive Paper and it's implementation

- It seems that Hive has brought forward the use of Hadoop for those who are used to using a SQL based database.
- Hive has made it much easier to user Hadoop, in that there doesn't seem to be as much programming needed for each task.
- Hive is still missing quite a bit of SQL, but they do state that they are working toward making HiveQL subsume SQL syntax.
- Hadoop is great for handling large amounts of data and Hive has made it easier to manipulate that data with the SQL-like HiveQL.
- Many will read this paper and see that they are using Hadoop and Hive on Facebook. They will think if it works for them with all the people using it and all the data that comes across Facebook, maybe we should give it try.

Comparison of the ideas and implementations of the two papers

- It is always a good idea to compare systems. The choice of what data set to use in a comparison such as this can be slanted toward the system that you want to "win". As each system had it's winning and losing qualities, I don't think that this was done here even though I previously stated that I thought they were leaning toward Hadoop. I was interested to see that Michael Stonebraker contributed to this comparison in 2009 after seeing his video about his paper in 2005 (which already stated that RDBMS and row-store weren't the answer).
- Hive and HiveQL were described and shown to have made Hadoop much easier to use by the Facebook team who built it.
- I believe that when comparing the three systems: Hadoop, Vertica and DBMS-X, if Hive had been used with Hadoop, there might not have been as many differences. Of course without running the tests and comparing results, there is really no way of knowing.

“One Size Fits None” ideas

- 1970 to 2000 was about trying to make a universal RDBMS the answer to any question.
  - Added abstract data types, referential integrity, triggers
  - SQL specs grew from 100 to 1500 pages in the attempt.
- 2005 – They realized it would never work.
  - Streaming apps couldn’t be wedged into traditional row-store RDBMS implementation.
  - Column-store was being worked on.
- 2015 – He lists the different data markets and what might work for them:
  - Data Warehouse Market - all have or will have c-store; 2 times faster then row-store;
  - OLTP Market – moving toward main memory deployment with lightweight transaction techniques;
  - NoSQL Market – use a mix of data models and architecture, none standard;
  - Complex Analytics Market – use data warehouses that run business intelligence products; in future will be run by data scientists with regression, eigenvectors, data clustering, all defined on arrays;
  - Streaming Market – OLTP and streaming engines are dominant;
  - Graph Analytics Market – c-store, array engine, graph engine – no standard;
- Great Opportunity for New Ideas and new ways to adapt older system so not to lose market share

Finally

- According to Stonebraker's video, there really aren't any advantages/disadvantages to the main ideas of the Hive paper or the Comparison paper because they are obsolete.
- The Comparison paper compares two DBMSs and Hadoop. The DBMSs are SQL based and row-stored, both stated as obsolete in his video. Hadoop is unstructured and therefore may not be obsolete.
- However, Hive and HiveQL make Hadoop more SQL-type based, so that may be something that should not be built on to Hadoop for the future.
- Stonebraker does say in the video that the different types of engines will be using a different type of SQL implementation depending on the different type of storage used. Hadoop may still be workable.
- Per Stonebraker "the jury is out on who will win."