Hive

A Petabyte Scale Data Warehouse Using Hadoop (Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy)

A Comparison of Approaches to Large-Scale Data Analysis (Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebraker)

Michael Stonebraker on his 10-Year most Influential paper Award at ICDE 2015

By: Piradon (Tien) Liengtiraphan

2016-03-12

Hive - The Interesting Parts

- Hive is a scalable open source warehousing solution built on top of Hadoop (a popular open-source map-reduce implementation) made to address the growing size of datasets being collected and analyzed in industry and to take advantage of and SQL-like declarative language called HiveQL.

- Throughout the paper the authors write about the usages of Hadoop and their shortcomings: being too low level at times, requiring a lot of maintenance, etc.

- They then go on to address how Hive addresses these issues: HiveQL, underlying IO libraries, having a built in system catalog “*Metastore*”, etc.

- Hive essentially is a specialized Hadoop designed specifically with the functionality of Business Intelligence and Analytics in mind. The rest of the paper goes deeper into the details of the functionality and features of Hive: Introduction, Data Model, Type System and Query Language, Usage Example, and Conclusions + Future Works.

Hive - How the Bees at Facebook did it

- Basics: Hive was built off a foundation of Hadoop. Hadoop already optimized the analysis of large bodies of data; however, lacked the “expressiveness of popular query languages like SQL. The goal behind Hive was to bring the data analyzed by Hadoop and present it to the user in an intuitive or more simplistic way than it was already being done.
- *HiveQL*: Bringing the concept of tables, rows, partitions, and columns into Hadoop, to be accessed by what was originally a subset of SQL. Supports all major primitive types - doubles, strings, floats, and integers; with the possibility of extending support with external libraries.
- Example Scripts:
  - CREATE TABLE t1 (st string, fl float, li list<map<string, struct<p1:int, p2:int>>);
  - SELECT t1.a1 as c1, t2.b1 as c2
- Data Storage: Tables are still logical data units in Hive, the metadata associated with those tables are now stored in a table to hdfs directory.
  - Tables - A table is stored in a directory in hdfs
  - Partitions - A partition of the table is stored in a subdirectory within a table’s directory.

Hive - My Opinions

- Positives: The addition of an SQL-like declarative language was an inspired move to make the system more accessible to the wider IT community. Another positive thing that was done was also the addition or rather expedited process in which other libraries can be incorporated into the Hive system.
- Possible Improvement: To say that Hive was a complete improvement over Hadoop would be a slight overstatement, given that Hadoop was designed to be used with a wider breadth of things. More SQL statements should be implemented into HiveQL so help ease the process of moving over to Hive from an SQL database, and to eliminate the need for external libraries in basic query functions. Replacing the naive rule-based optimizer with one that prioritizes cost and adaptivity would improve the functionality of the system.
- Currently Hive runs at about the same speed as Hadoop with about a 20% chance of being slightly slower. However Hive is still a work in progress and therefore still has a lot of room for improvement. Currently the team is looking into multi-query optimization techniques and n-way joins in a single map-reduce job.

A Comparison of Approaches to Large-Scale Data Analysis - Main Points

- The general purpose of the second paper, was to illustrate the differences of the Map-Reduce approach to Business Intelligence when compared to other parallel systems designed for similar purposes.
- Map-Reduce (MR): One of the most commonly used methods for managing Big Data. MR is considered the most simplistic of the two, ajd allows input data sets to be stored in a collection of partitions, which are then deployed to each node in a cluster via a distributed file system.
- DBMS: Have existed since the late 1980s, meaning there is a lot of support for it avaliable. The benefit of DBMS over MR is that it supports standard relational tables and structured query language; making the data stored on multiple machines transparent to the end-user. However it should be noted that DBMS is generally considered more expensive to maintain.
- General Conclusion: As determined by the tests laid out in the paper, it was concluded that on a general principle DBMS performs better than MR, with DBMS-X and Vertica performing over 2 times faster than Hadoop.

A Comparison of Approaches to Large-Scale Data Analysis - Ideas and Implementation

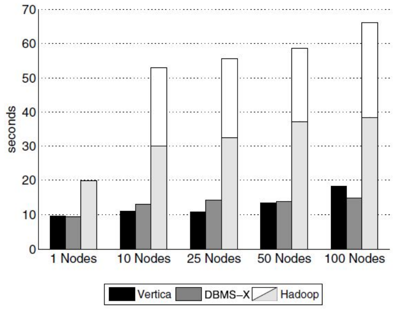


Figure 4: Grep Task Results – 535MB/node Data Set

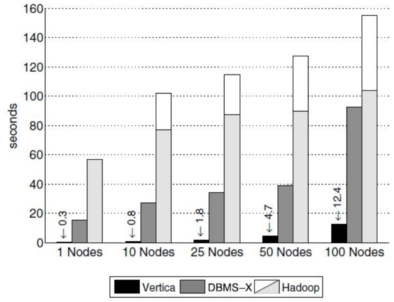


Figure 6: Selection Task Results

- The authors of the paper go on to illustrate the differences between the two systems, through both a theoretical analysis of their background and backbone processes and a plethora of test designed to compare the system's performance, fault tolerance, and flexibility. The Tests Include: Grep Task, Task Execution, Data Loading, Selection Task, Aggregation Task, Join Task, and UDF Aggregation Task.
- MR Candidates: Hadoop - The most popular open-source implementation of MR
- DBMS Candidates: DBMS-X and Vertica
- As demonstrated by combining the results of all those test, DBMS outclasses MR in most regards, beating out Hadoop in all but a few isolated Grep Tasks.
- General Conclusion: DBMS is much more flexible when compared to MR, due to the fact that it relies on an indexing system, without the need to separately perform two tasks like the Map and Reduce function that make up MR.

A Comparison of Approaches to Large-Scale Data Analysis - Analysis of Ideas and Implementation

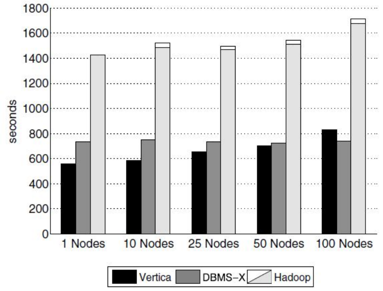


Figure 7: Aggregation Task Results (2.5 million Groups)

- Throughout the paper we were given the impression that while both systems do what they were intended to do, analyse and index large repositories of data, DBMS seems to outperform MR in most categories.
- To illustrate, the aggregation task test performed: required the system to calculate the total adRevenue generated for each sourceIP in the UserVisits Tables (20gb/node), and group them by the sourceIP column.
  - `SELECT sourceIP, SUM(adRevenue) From UsersVisits GROUP BY sourceIP`
- The reason these test were chosen was because for MR to perform this specific task it requires both the Map and Reduce functions, showing the performance of the system when both main elements are being used.
- The results of the table show that MR was outdone by DMS in every case, due to the fact that DBMS executes these queries by having each node scan the local table and extract the necessary fields. These are then locally merged, without the need for as much overhead as MR.

Comparison of the 2 Papers (Ideas and Implementations)

Hive - A Petabyte Scale Data Warehouse Using Hadoop

Compared Hadoop and Hive, the latter being a specialized form of the former and being built off of the former.

Discussed the structure of the new system when compared to the old, and emphasized the improvements over the older system.

Emphasized the improvement of the older system with the additions in the newer. Focused mainly on describing the usage and syntax of the new system.

Opinions

The two papers were quite different from one another. One focused on the improvement of an existing system to make it more accessible and intuitive to the wider community, the other focused on the comparison of two existing system.

The goal of the two were different. Hive being an improvement over Hadoop in terms of functionality and usage. Large-Scale Data Analysis serves more as a benchmark and perhaps a pitch toward the usage of DBMS over MR.

A Comparison of Approaches to Large-Scale Data Analysis

Compared 2 similar system for Big Data/Business Intelligence analysis.

Discussed their advantages using both the theory behind them and the empirical evidence gathered from the tests there were run to compare the systems.

Emphasized the difference between the two systems and the performance of each when put to specific tasks.

Stonebreaker Talk - Main Ideas

- The idea behind relational databases was to be the “One size fits all” which Stonebreaker claims to have been an idea whose time has “come and gone”
- It was originally claimed that RDBMS provided
  - Abstract Data Types, Referential Integrity, and Triggers
  - And therefore could be the “universal” solution to database models.
- There was no place for streaming applications, within the traditional row-store of RDBMS implementation
- It was determined that it was impossible for “one size to fit all” (based on a sample of 3) in Stonebreaker’s 2005 Paper.
- One Size Fits None: “DB2, Oracle, Sequel Server are good for nothing”
  - Data Warehouse Market: Moving towards Column Store, evidence points to it being 2 orders of magnitude faster.
  - OLTP Market: Main Memory Deployments, row stores are resource consuming and heavy weight. Lightweight transactions are proven to be more effective for OLTP markets. (TimeStamp Techniques, MVCC, OCC, Cmd Logging, and Active-active replication)
  - NoSQL Market: 100+ vendors. Compared to “Potpouri of data models and architectures”. No Standards.
  - Complex Analytics: Business Intelligence using data warehouses. Congnos and Business Objects. Capable of performing basic/standard data analytics. Regressions, Data Clustering, Predictive Models, the Business Intelligence of the future are defined on Arrays and will rarely include tables at all. While SQL is capable of performing these operations they are tedious to write and take an overly long time to accomplish. Tables can be cast to arrays with table add ons but functionality has not be thoroughly tested.
  - Steaming Market: Not based on traditional row stores. OLTP engines seem to have greater market share here. Kindred to a message coming in to an application, which then runs a stored procedure which is then run again the main memory database system.
  - Graph Analytics Market: Simulating a column store architecture or and Array Matrix. (E.G Graphlab). Tradditional Row stores are not ideal for this kind of emulation.
- There is a huge diversity of engines, all of which are specialized in their own way. Traditional Row Stores do not meet any of these criteria.
- NVRAM, Big Main Memory, Processor Diversity, Higher Network Speeds, LLVM, Vectorization. All these fields are advancing these days allowing for new architecture to emerge that will take advantage of these new technologies.

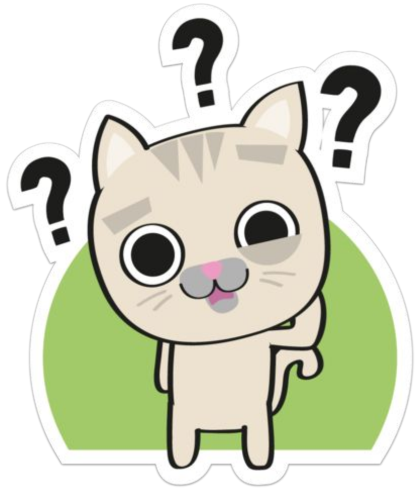
Advantages and Disadvantages of the main idea of Hive in context of Hadoop Paper and Stonebreaker Talk

Advantages

The Hive paper is extremely detailed in its analysis of its parent system when in comparison to itself. A majority of the paper consists of the authors explaining the differences through comparisons of what was lacking within Hadoop and the improvements made to address this within Hive.

Another advantage is the length at which the authors went to to illustrate the different usages of Hive (giving examples of SQL statements which can be run, and explaining at length the theories behind them).

The conclusion also addresses the fact that Hive is still a work in progress and that it is continually being worked on, making it a constantly evolving technology.



Disadvantages

For all the advantages that the Hive paper presents, its sole purpose was to introduce the readers to an improvement over the old Hadoop system. The problem lies in that it covers one side of a multi sided argument.

While hive might be the ideal solution in a particular situation it could possibly me slow and unresponsive for another. This is not given consideration within the Hive paper itself.

Another disadvantage that the idea behind the Hive paper presents is that it is based on Hadoop, which has been proven in the comparison paper to generally be the slower of the two compare system. It would see that the authors of Hive chose Hadoop to be the basis of their project, without consideration for other possible systems that might have worked out better. This however could be stated as the author’s attempt to make Hive more accessible, since Hadoop, while slower, is still the most popular implementation of the MR concept.