Data Paradigms

Vishnhav Ashok

Illinois Institute of Technology

Author Note

Vishnhav Ashok, School of Applied Technology, Illinois Institute of Technology.

Abstract

In the modern world of big data, processing and storing petabytes of data can be a challenge for even the greatest of minds. During the early 2000's processing large volumes of data, either structured or unstructured was a mighty task. DMBS and SQL have long lived where querying and fetching data from was well known while the learning curve for its implementation was relatively simple. However, there still remained large chunks of bulk data that was unprocessed, unstructured and was difficult to iterate, extract and read. MapReduce was introduced in an attempt to solve this issue. MapReduce is a library that was built under the premise that its input was raw-unstructured-schema-less data. Michael Stonebraker, a renowned Data Scientist however thought the very use of MR was a huge step back in the way such data was handled written in his blog post titled '*MapReduce: A major step backwards*'. Stonebraker also pointed out that MR did not fit the standards and protocols that RDMBS followed (Stonebraker and David, 2008). Mark C. Chu-Carroll, a Google employee, put forth his ideas and views suggesting how MapReduce is a different tool used for completely different types of datasets in his blog post titled '*Databases are hammers; MapReduce is a screwdriver.*' (Mark, 2008). Both MapReduce and RDBMS have their merits and de-merits and with the rapid growth of Cloud Computing, Big Data Analytics, increasingly sophisticated gadgets and the capability for real time sharing of data and information, humans have to look past a technology's indifference and work towards integrating them.

Keywords: MapReduce, Relational Database Management System

Discussion

David J. DeWitt and Michael Stonebraker Analysis. The primary argument according to Stonebraker was that MapReduce is a tool that sends back the database community 50 years in its past as it does not account for any of the protocols followed by a conventional RDMBS system such as Schemas, Transaction, Views, Indexing. Although the paper asserts that MapReduce advocates would implement such a system on an unstructured, schema-less data, the probability of a Map to find atleast one data field in each input record was skew at the time (Stonebraker and David, 2008).

The paper also recapitulates that Map and Reduce requires new instances to run everytime causing performance lags. Furthermore, MapReduce jobs runs in isolation making it difficult to provide any input during a run. Stonebraker also points out that MapReduce is incompatible with existing DBMS tools while it's also missing key features that DBMS already provides a solid foundation on (Stonebraker and David, 2008). Although running a MapReduce application on top of Hadoop (HBase) might improve latency and throughput across multiple machines, it was still a step backward when dealing with data as a whole.

Databases are hammers; MapReduce is a screwdriver. Mark C.'s notion is that the very use and implementation of MapReduce is different. Affording a supercomputer to analyze and extract data was expensive. MapReduce solves the capital required to run such a job by instead running on a bunch of parallel cheaper machines (Mark, 2008). The Map splits the bulk data into smaller chunks using key/value pairs across multiple machines while the Reducers combines the values to deliver the required result; as such MR programs were also easier to code (Mark, 2008). Mark never criticized the tools RDMBS provides but rather stated how MapReduce and DBMS are two different tools for two different types of datasets. To iterate and read through data in a tabular form can be done efficiently by indexing, but what if the data is just a raw text file with no structure.

Mark brought about an excellent analogy and compared RDBMS to the mightiest of mighty hammers (Mark, 2008). However great a tool is, its purpose is only good as the input you feed in. One might argue using a hammer for a screw; it would work, but removing, modifying or changing the screw would be a nightmare. Similarly, using a RDBMS system on an unstructured data might do the job intended while making it more time consuming and complex in the process, but is it the most efficient way given all the tools and technologies at our disposal?

The Data Paradigm. Both papers try to emphasize why MapReduce or DMBS is better while contradicting each paper's point of view. it's hard to see why the comparison was even being made, when you consider the role of each tool in this light. It is quite clear that the very idea of using either of the tools is for achieving different result sets (Jeffrey, 2004). While RDBMS is good for updating smaller proportions of a big database, MapReduce is good for updating larger chunks of data. RDBMS is good when data has to be updated frequently while MapReduce works better with WORM (Write once, read many times). Integrity is high in RDBMS while MapReduce is more scalable. It is even possible to use MapReduce to read unstructured data while implementing its output in a tabular structure using RDBMS (Nate, 2014). A DBMS system will have to load the data before a query can process a result set while MapReduce has a relatively short loading time. MR should complement DBMS rather than competing with it.

This paradigm was also during a time when Hadoop and its many frameworks hadn't been developed yet. Since then, Hadoop has been extended with many different frameworks that sits on top of an HDFS system, even including SQL(HIVE) (Jeffrey, 2004). Zookeeper provides centralized services for Hadoop cluster configuration management. Ambari administers a Hadoop cluster (Nate, 2014). Apache Storm can be used for streaming support and Spark, a real-time general engine is utilized for data processing among dozen more

frameworks. Hadoop has been the frontrunner in Big Data in the recent past. Due to the exponential growth of unstructured data such as emails, text documents, videos, photos, audio files, and social media posts, Hadoop's (MapReduce, Hive, Spark, Etc.) ability to join, aggregate, and analyze vast stores of multi-source data without having to structure it allows companies to gain deeper insights much more efficiently and quickly than a traditional database. However, Hadoop's biggest shortcoming is its inability to provide security across large clusters while it's also missing any form of encryption (Nate, 2014). The environment Hadoop is running on should have its own security measures in place. As of our current state Stonebraker's criticism can no longer be applicable due to the massive improvements to MapReduce and its large spectrum of available integrated frameworks. Apache Hadoop is an easy open source framework implementation of MR. In retrospect, MapReduce is ultimately a data analytical tool in its very core.

Conclusion

Needless to say, MapReduce and RDMBS can co-exist in the same cluster of data. What each of these platforms have in common is the ability to improve the efficiency and reliability of data collection, aggregation, and integration largely working with different datasets. It is interesting to note how companies jumped on the band wagon, that is, MapReduce since its release by Google, while Google themselves have since moved onto BigTable, a more interactive storage system, at that time. In 2017, MapReduce alone is inefficient to process data in bulk and does require integration with other platforms (Hadoop, HBase, Hive, etc.) to be able to retrieve more efficient results (Stonebraker, 2014). Companies should focus more on integration and innovation of such tools to deal with the many challenges humanity is yet to face with BigData, or could wait for Google to release its next big platform while they are already a country mile ahead of its competitors.

References

Jeffrey, D., (2004). MapReduce: Simplified Data Processing on Large Clusters. *Google, Inc.*, 1-13. Retrieved February 07, 2017, from https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

Mark, C., (2008). Databases are hammers; MapReduce is a screwdriver [Web log post]. Retrieved February 07, 2017, from http://scienceblogs.com/goodmath/2008/01/22/databases-are-hammers-mapreduc/

Nate, P., (2014). Hadoop vs. Traditional Database [Web log post]. Retrieved February 10, 2017, from https://www.qubole.com/blog/big-data/hadoop-vs-traditional/

Stonebraker, M., (2014). Hadoop at a Crossroads? [Web log post]. Retrieved February 09, 2017, from http://cacm.acm.org/blogs/blog-cacm/177467-hadoop-at-a-crossroads/fulltext

Stonebraker, M., David, J., (2008). MapReduce: A major step backwards [Web log post]. Retrieved February 07, 2017, from http://homes.cs.washington.edu/~billhowe/mapreduce_a_major_step_backwards.html

Appendix A

Michael Stonebraker is a pivotal figure in the database industry. Currently an Adjunct professor at MIT, Stonebraker has dedicated the better part of his life towards reinforcing the advancements in a ’Conventional’ DMBS system. Winner of ACM Turing Award in 2014, he has been the cofounder of a number of database companies such as Illustra, Cohera, StreamBase Systems, Vertica, VoltDB among many others. Ingres, an efficient implementation of the relational model kickstarted his career in this field while key elements of this project is still widely used today especially B-trees.

Mark C Chu Carroll, maybe not as well established like Stonebraker yet played a pivotal part working with Google’s implementation of MapReduce while his main specialization was in software configuration management and distributed systems. Famous for his website ‘Good Math Bad Math’ Mark has played an integral role in a vast array of companies he has worked in such as Google, ForeSquare Labs, Twitter, while he’s currently a Software Engineer at Dropbox.