Data Paradigms Comparison Paper

Phanindra Chandraprakash (A20376416)

Illinois Institute of Technology, Chicago, Illinois

Article Link:

http://homes.cs.washington.edu/~billhowe/mapreduce_a_major_step_backwards.html

About Authors:

David J. DeWitt:

David DeWitt is the John P. Morgridge Professor of Computer Sciences, Emeritus at the University of Wisconsin, Madison. Professor DeWitt received a B.A. degree from Colgate University in 1970 and a Ph.D. degree from the University of Michigan in 1976. In 2008 he retired from the University to start the Microsoft Jim Gray Systems Lab in Madison, Wisconsin. Professor DeWitt is a member of the National Academy of Engineering. In 1995, he was named a Fellow of the ACM and received the ACM SIGMOD Innovations Award for his contributions to the database systems field. In 2009, ACM recognized the seminal contributions of his Gamma parallel database system project with the ACM Software Systems Award. IEEE awarded him the 2009 IEEE Piore Award for "fundamental contributions to the architecture, algorithms, and implementation of innovative database systems." He has published over 120 technical papers. (dewitt, n.d.)

Michael Stonebraker

Michael Stonebraker has been a pioneer of data base research and technology for more than a quarter of a century. He was the main architect of the INGRES relational DBMS, the object-relational DBMS, POSTGRES, and the federated data system, Mariposa. All three prototypes were developed at the University of California at Berkeley where Stonebraker was a Professor of Computer Science for twenty-five years. He is the founder of three successful Silicon Valley start-ups, whose objective was to commercialize these prototypes.

Professor Stonebraker is the author of scores of research papers on data base technology, operating systems, and the architecture of system software services. He was awarded the

prestigious ACM System Software Award in 1992, for his work on INGRES. Additionally, he was awarded the first annual Innovation award by the ACM SIGMOD special interest group in 1994, and has been recognized by Computer Reseller News as one of the top five software developers of the century. Moreover, Forbes magazine named him one of the 8 innovators driving the Silicon Valley wealth explosion during their 80th anniversary edition in 1998. He was elected to the National Academy of Engineering in 1998 and is presently an Adjunct Professor of Computer Science at M.I.T. (stonebraker a. , n.d.)

Comprehensive Analysis:

In this article the authors try to compare the MapReduce framework with Parallel RDBMS by highlighting the main disadvantages of the MapReduce framework over Parallel RDBMS. They also clearly explain the concept of Map and Reduce by drawing an analogy with SQL. Author says that the MapReduce framework is developed only for general-purpose computations but not for data intensive application and it is not scalable. Authors describes MapReduce framework as, building a data center with several low-end processors/servers, instead of using very less number, powerful high-end processors/servers to solve a computing problem. In one of the articles, the authors quotes MapReduce framework as an Extract-Transform-Load (ETL) System, and it is a complementary technology to the parallel RDBMS and not a competing technology, because RDBMSs are not good at ETL tasks (stonebraker M. v., n.d.). The authors conclude the article by five disadvantages of MapReduce framework over parallel RDBMS. (Team, n.d.)

The authors juxtapose the parallel RBBMS by quoting below drawbacks of MapReduce framework:

1. Not suitable for large-scale data intensive applications.
2. Forcefully acts upon the data instead of indexing.
3. Represents a specific implementation of RDBMS, and hence the idea is not novel.

4. Most commonly used RDBMS features are missing.
5. Incompatible with external and internal tools of RDBMS.

Authors explains the contradicting viewpoint of MapReduce over conventional RDBMS by describing that, RDBMS follow schemas, application and schemas are separated, and high-level access languages are compatible with RDBMS, on the other hand MapReduce will not follow any of the above rules and regulations, since RDBMS follows schemas, there will be very less garbage which will be added to data, where as in the MapReduce framework, there are no such functionalities and hence it's difficult to control the garbage entering into the good data. They also argue that since there is no schema concept in MapReduce or if it is buried in the application, the programmer or the user who wants to work on the related application must do exploratory work to identify the schema, which is a tedious work. The key issue during 1970's "great debate" between relational advocates and the Codasyl advocates was "How to program the database access?". Relational advocates argue stating it should be "What do you want?" on the other hand Codasyl advocates contend stating "There must be an algorithm to access the data". In this debate the Codasyl advocates failed to explain their stance, because Codasyl was an assembly level language of DBMS access. MapReduce might contradict this verdict since, they are only targeting datasets which does not have any schema, but the Relational advocates dismisses this version stating "In extracting a key from the input data set, the map function is relying on the existence of at least one data field in each input record. The same holds for a reduce function that computes some value from the records it receives to process".

Second point is, all the modern DBMS systems will use hash or B-tree indexes for effective data retrieval, but MapReduce framework doesn't use index functionality, especially while dealing with skew and data interchange and hence MapReduce is a low-level and poor implementation framework. Continuing this argument, they also discretize the MapReduce

mechanism and its effect on local disk. For each of the N map instances produces M output files, and these output files are destined for different reduce instance. This process or Map and Reduce will slow down the effective disk transfer rate by a factor of 20, which is not the case in parallel RDBMS systems, because parallel database systems will not substantiate split files and it will use push (the data) instead of pull. (Stonebraker, n.d.)

Thirdly, MapReduce framework is not new, this method employs the techniques more are 20 years old. The idea of splitting the larger data into fragments was first proposed in "Application of Hash to Data Base Machine and Its Architecture". Teradata has been using this same principle to develop commercial RDBMS systems for more than 20 years. Also, MapReduce framework claims of using MapReduce functions which are eventually like POSTGRES supported user-defined functions and user-defined aggregate functions of parallel RDBMS systems. Finally, the authors criticize the MapReduce framework by pointing to missing features like; Bulk Loader, Indexing, Updates, Transactions, Integrity Constraints, Referential Integrity, and Views. Also, MapReduce framework is incompatible with many RDBMS tools like; Report Writers, Business Intelligence Tools, Data Mining Tools, Replication Tools, and Database Design Tools. (Research, n.d.)

Authors ends their argument by drawing following conclusions; In an analogy with SQL, map is like the group-by clause of an aggregate query. Reduce is analogous to the aggregate function (e.g., average) that is computed over all the rows with the same group-by attribute and MapReduce framework is not new, and this technology has not learnt anything from a 40-year-old database technology, and MapReduce is an assembly level language and nobody should be forced to program in MapReduce.

---------- End of Article 1----------

Article Link:

http://scienceblogs.com/goodmath/2008/01/22/databases-are-hammers-mapreduc/

About Authors:

Mark Chu-Carroll:

Mark Chu-Carroll is a software engineer at Google. He's been working on programming languages and software development tools for close to 20 years. In his free time, he's the administrator/developer of Scientopia.org, where writes the blog Good Math/Bad Math. You can visit his blog at http://scientopia.org/blogs/goodmath.

Comprehensive Analysis:

The author of this article is an employee of Google and has never used MapReduce at his work, he explains the concept of MapReduce framework by prompting realistic examples. While working on structured data in non-scientific applications, he proposed a hierarchical scattered/gathered solution to handle these kind of data, which is very similar to the way that MapReduce works. He explains MapReduce by giving following example; if you want to execute a job which is going to take long time on your computer, in that case it's better to split the jobs in pieces and run across multiple computers, once the job is completed integrate the outputs from all the machines and make it as a single output.

In general, doing these kind of job is vert tedious task but MapReduce is a library that gives you a chance to receive a specific, adapted method for writing computer programs that is anything but difficult to part among a pack of machines. The essential thought is that you isolate the occupation into two sections: a Map, and a Reduce. Map takes the issue, parts it into sub-parts, and sends the sub-parts to various machines – so every one of the pieces keep running in the meantime. Reduce takes the outcomes from the sub-parts and consolidates them back together to find a solitary solution. The aftereffect of the guide calculation is a rundown of key/value sets. Reduce takes each arrangement of qualities that has a similar key,

and joins them into a solitary value. So, Map takes an arrangement of information lumps, and delivers key/esteem sets; reduce blends things, so that rather than an arrangement of key/esteem match sets, you get one outcome.

This author states that relation between RDBMS and MapReduce is like a Hammer and a Screwdriver. He states that RDBMS is like a Hammer, and it will be useful if you are dealing with nails but if you have to work with a screw then you would need a screwdriver. MapReduce framework acts like a screwdriver for specific set of computations. He also states that RDBMS are poor at handling non-tubular data and they are also notorious for performing a poor job on recursive data structures.

He gives his opinion on the following points mentioned by RDBMS specialists with respect to MapReduce framework.

- A giant step backward in the programming paradigm for large-scale data intensive applications
- A sub-optimal implementation, in that it uses brute force instead of indexing
- Not novel at all — it represents a specific implementation of well-known techniques developed nearly 25 years ago
- Missing most of the features that are routinely included in current DBMS
- Incompatible with all the tools DBMS users have come to depend on.

Point one is a superfluous. M/R is an incredible method for programming some vast scale information serious applications. If you have an extensive calculation whose information is actually depicted relationally, and which is doable to keep running on a solitary huge machine, and you have admittance to said huge machine, then no question – you ought to utilize a RDB. What's more, for some calculations that appear to be excessively unpredictable, making it impossible to run is a sensible measure of time on a solitary machine, working out the correct format in tables, with the best possible ordering plan, can

make what appeared non-achievable absolute simple. However, that is not each calculation. That is not in any case generally calculations. MapReduce isn't planned to supplant the relational database. He substantiates his view by giving an example of NebulaBrot.

Proceeding onward to point two, which was that MapReduce is primitive and problematic, because it doesn't utilize indexing. Same essential reaction: Indexing is an incredible device if your information is tabular, and you have a focal list that you can work with. Nevertheless, if your undertaking isn't essentially relational, and what you truly need is calculation – like the enormous quantities of gliding point multiplications in the NebulaBrot – then the concept of indexing wouldn't help. The issue that MapReduce is attempting to unravel is making it simple to compose a program that does an immense measure of calculation in a sensible measure of time. Records don't help if the fundamental assignment is computationally extreme. (CATALIN BOJA, n.d.)

Point three: “it’s not novel.”, MapReduce never claims that it is novel, in one of the earlier publications they have specifically stated that “it was inspired by the Map and Reduce primitives from functional programming languages! MapReduce is, basically, a form of data parallel computing – that is, a scalable implementation of a well-known, well understood, easy-to-use model of parallel programming”. Hence, this is an irrelevant argument.

Point four: “missing features in RDBMS”. Since MapReduce isn't a database application. On the off chance that you have a relational database issue, utilize a relational database. In the event that you have a huge computational undertaking that you need to disperse among handfuls, or hundreds, or a large number of little machines, then utilize MapReduce. DBMS tool are not meant for handling massive cluster/cloud computing. He contradicts point five “incompatible with tools that DB developers have come to depend on” by stating hammer and screwdriver story, and he by again stating that MapReduce is not a database and hence these tools won’t be compatible with it.

He summarizes his stance by stating that MapReduce is not an alternative for relational databases, it's entirely different functionality. He also criticizes the database guys by stating "They really don't seem to understand what M/R is designed for, or why it's designed the way it is." He concludes his argument by affirming "Just because you've got the best hammer in the entire world doesn't make everything a nail. If you've got a screw, even a cheap, old, rusty screwdriver is going to do a better job. And MapReduce is a lot better than a cheap, old, rusty screwdriver".

---------- End of Article 2----------

Conclusion:

From the above articles, I can convincingly make out that the MapReduce is originated or developed using relational database technologies, but the implementation and presentation of this framework is different because it is used for computing disperse and huge volume of data, which is not possible with parallel relational database systems. MapReduce is a complementary system to RDBMS and it is very effective on unstructured or semi-structured data. One should not compare MapReduce and RDBMS systems for their functionalities or logical structure, they both go hand in hand. I can conclude this comparison by mentioning subtle differences between these two systems: (RDBMS, n.d.)

Attributes	RDBMS	MapReduce
Data Size	Gigabytes (10^9)	Petabytes (10^{12})
Access	Interactive and Batch	Batch
Updates	Read and Write many times	WORM (Write Once, Read Many times)
Data Structure	Static Schema	Dynamic Schema
Integrity	High	Low
Scalability	Nonlinear	Linear

References

CATALIN BOJA, A. P. (n.d.). Retrieved from https://pdfs.semanticscholar.org/c2b4/756fa5daa85ae2a353748d812622e8787af6.pdf

dewitt, a. (n.d.). Retrieved from http://pages.cs.wisc.edu/~dewitt/

RDBMS, C. M. (n.d.). Retrieved from https://www.coursera.org/learn/big-data-cloud-computing-cdn/lecture/Mv70w/mapreduce-vs-rdbms

Research, M. (n.d.). Retrieved from http://www.dbms2.com/2008/09/04/mike-stonebraker-mapreduce/

stonebraker, a. (n.d.). Retrieved from https://www.csail.mit.edu/user/1547

Stonebraker, M. (n.d.). Retrieved from http://istc-bigdata.org/index.php/no-hadoop-the-future-of-the-hadoophdfs-stack/

stonebraker, M. v. (n.d.). Retrieved from http://db.csail.mit.edu/pubs/p64-stonebraker.pdf

Team, T. (n.d.). Retrieved from http://www.tamr.com/stonebraker-on-hadoop-in-managing-big-data/