MapReduce VS Relational databases

Aneesh Partha

Illinois institute of technology

Author Note

Aneesh Partha, Department of Information Technology and Management, Illinois Institute of Technology.

Correspondence concerning this article should be addressed to Aneesh Partha, Department of Information Technology and Management, Illinois Institute of Technology.

Contact: apartha@hawk.iit.edu

Abstract

This paper explores two published articles that performs discussion and comparison between MapReduce framework and relational database management systems(RDBMS). David, author of "MapReduce: A Major step backwards" is an advocate of relational database systems and states that MapReduce are still left back in 1960s. He supports relational database systems which is available for several decades and against the MapReduce programming. On other hand, Mark the author of "Databases are hammers; MapReduce is a screwdriver" is an advocate of MapReduce and argues that the information discussed in the article by David is not right and MapReduce has its own set of operations and it cannot be compared with a traditional database systems. He also states that MapReduce is not an replacement for database systems.

Keywords: Relational Database Management Systems

MapReduce VS Relational databases

The paper focuses on two articles which supports Relational database systems on one hand and MapReduce on other hand. The discussion gives us an insight of what a relational database can perform and what a MapReduce programming can perform. The argument gives us an idea of which method to choose based on the amount of data which we have. Before beginning with the discussion, it is essential to understand the concept of Relational database and MapReduce.

Relational database

It is a database whose structure is based on relational model of data. Special softwares known as relational database management system are used to manage the relational databases. Few examples of RDBMS are Oracle, MySQL, Microsoft SQL Server, PostgreSQL and IBM Db2. These are some of the most widely used RDBMS in the world. Apart from these RDBMS there are plenty of other systems available in the market. In a relational database data is stored in the tables consisting of rows and columns. Each table has a unique primary key which distinguished one row from the other. The data in a relational database can be accessed using queries. Structured query language which in short is called as SQL is the language used for accessing the data in the databases. Tables in a database are linked to each other using keys.

MapReduce

MapReduce is a generally a programming model which is used for processing and generating large datasets. It is called as “MapReduce” due to its two main functions which is a Map and Reduce function. MapReduce focuses on parallel processing of data. A data set is broken down into many blocks and it is assigned to cluster of machines for processing the data. Each cluster node is programmed to perform Map and reduce function. The map program reads a

set of records from an input file does desired filtering and transformation and then outputs a set a records of the form (Key,Value) pair. Key in the key value pair are always unique and it is used to identify the value. This output of (Key,value) pair from Map phase is pushed to the reduce phase. Before sending it to the reduce phase the output of map phase is shuffled and sorted. This sorted output is sent to reducer for performing the reduce operation. The computation in the reducer is based on the output required. In general, the reducer can do anything it wants with its records. The output is then written to a file by the reducer from which it can be accessed. MapReduce is well suited for performing computations on large datasets than small datasets.

Both relational database and MapReduce are good at specific operations. It is very important to understand the working of both the systems and decide the place where it can be implemented. Both the systems have equal number of pros and cons.

The author in both the articles state their view on database and MapReduce based on the below five viewpoints. It is in fact David De Witt author of "MapReduce: A major step backwards" who started his article revolving around the below points. Later Marc C. Chu-Carroll author of 'Databases are hammers; MapReduce is a screwdriver." has stated his contradictory views to what David has said in his article.

- Backward in programming paradigm
- Indexing
- Implementation of previously developed techniques
- Lack of features
- Incompatible with tools.

Backward in programming paradigm for large-scale data intensive application

In the first criteria, the author has stated that MapReduce is step backward in database access. He has given this statement due to fact that MapReduce do not follow a schema and this pushes the concept of MapReduce to back in late 60's. Presence of schema helps in aligning the data into tables as rows and columns. Since there are many restrictions to add a data into table like referential integrity, primary key constraint, foreign key constraint etc. it prevents unwanted data being added into the database. This is a great advantage of relational database and it is not available in the MapReduce framework. In this article, it is also stated that the schema must be separated from the application. This is because when the schema is separated it is easy for programmers to find the structure of the data. The programmers can query the database to find the structure of the database which reduces the time of discovering the structure. This is contradicted by mark in his article where he states that MapReduce does not follow schema as the large data sets do not follow a schema oriented information. MapReduce is designed for large data sets and we cannot expect the data sets to follow a schema. MapReduce is intended for unstructured and structured data. It Interprets data during processing time. This provides flexibility and avoids the costly data loading phase of an RDBMS.

As you can see in picture 1.1 the MapReduce is schema on read which means that the data stored is processed only during read. Whereas RDBMS is schema on write that is the reason its data is stored as rows and columns.

Mark in his article has also stated that MapReduce is not a replacement for RDBMS. He has also stated that for large data set computations where data is naturally relational and able to run on a single huge system then it is good to go with RDBMS. In a place where RDBMS is unable to support the computation then in this scenario MapReduce is the only best way to proceed with the computation. This clearly tell us that MapReduce is a replacement only when the relational databases are unable to support the computation. Supporting this point, he has given an example of building a huge NebulaBrot. He has explained how MapReduce that is logic of parallel processing has helped the reader in building the Huge NebulaBrot.

The figure 1.2 tells the clear difference of the data which can be handled by the RDBMS and MapReduce. RDBMS can only handle data in the range of Gigabytes whereas MapReduce can extend until Petabytes. Also, as you can see RDBMS follow ACID but there are no restrictions placed on MapReduce.

MapReduce is Poor implementation

David on an overall view has stated that MapReduce is a poor implementation. He supports this statement using indexing. Relational database systems perform indexing during its search which reduces the amount time it takes for searching. Also, the author states that there is a query optimizer which decides if an indexing or brute force sequential search is necessary for a particular searching technique. This feature is not available in MapReduce and it has only brute force as the only option.

On other hand, Mark replies to this statement by saying that Indexes work best only if the data is stored in tables i.e rows and columns. When MapReduce is considered, data are generally

unstructured or semi structured and it cannot be stored in as a table. Hence it has no use of implementing indexing in MapReduce. MapReduce is used for writing program which does more of computations than any searching.

Unlike Relational databases, MapReduce is a batch query processor which allows us to run ad hoc query on large data sets and get results in reasonable time. It allows us to innovate new things with the data which is not possible in relational database systems. Page 7

David also stated that the parallel processing offered by MapReduce is something which was explored by DBMS research community in the 1980s with systems such as Teradata.
But in an article “MapReduce: Simplified Data Processing on Large Clusters” the author has stated that parallel processing was implemented earlier but in a smaller scale. Page 11.

Another problem with MapReduce is that the files from Map phase which will be sent to reduce phase are written to the local disk of the computer used to run the map instance. When the reduce, phase starts the reducer pulls the information from the local disk and there are chances that a single file is simultaneously accessed by more than 1 reducer which will in turn reduce the disk transfer rate by a factor of 20.

MapReduce is not novel

Author David states that MapReduce community is feeling themselves as being great that they have invented some new concepts and mentions that the concepts implemented by the framework is 20 years old and is already stated in “Application of Hash to Data Base Machine

and Its architecture". He also stated that Teradata has been selling DBMS since 20 years with all the techniques implemented by MapReduce.

In the article written by author Mark he has given comments against the above-mentioned points. Mark has stated that it has not said anywhere that it's a novel. In fact, in the first publication which described MapReduce stated that the Map and reduce was inspired from the functional programming. It cleared stated that the concepts were inspired from other sources and not invented on their own. MapReduce is basically a parallel processing framework which produces output as key,value pairs.

The Key Contributions of Map and Reduce framework are not the actual map and reduce functions but the scalability and fault tolerance achieved for a variety of applications by optimizing the execution engine. [Wikipedia]

MapReduce has missing features

David has mentioned in his article that the MapReduce is missing certain basic functions like Bulk loader, Indexing, Updates, Transactions, Integrity constraints, Referential integrity and Views.

The operations mentioned above by the author are all related to relational database systems. As we discussed earlier features related to relational database systems cannot be expected in the MapReduce framework. Since MapReduce programming deals with a large data sets it cannot implement all the features available in RDBMS. Moreover, for operations like indexing, integrity constraints, referential integrity and views we need a table of stored data. Since MapReduce store

unstructured and semi structured data it cannot store information in tables. Hence implementing these options in MapReduce is tedious and sometimes not possible.

The same has been replied by the author mark in his article. He has stated that the above-mentioned points by David is another way of saying that MapReduce is not relational. He has also mentioned that if it's a relational problem use relational database whereas if it's a computational problem which you want to distribute to a large number of clusters then use MapReduce.

MapReduce is incompatible with the DBMS Tools

In the former article, it is stated that certain set of tools such as Report Writers, Business intelligence tools, Data mining tools, Replication tools, database design tools are not compatible with MapReduce.

This is obvious that the tools which work with RDBMS might not work with MapReduce because MapReduce performs different operations and deals with large datasets. These tools would have been built keeping RDBMS in mind.

Mark has stated in his article that database tools and processes don't work for MapReduce programming. Tools built for RDBMS might perform poor for MapReduce. Hence MapReduce works well with its own set of tools.

Conclusion

Relational database community has performed various analysis in the past and discovered many things supporting the database. As we all know RDBMS are being used widely for many years and as mentioned by the author MapReduce programming is something which is getting name and fame in the recent years and it has inherited parallel processing which was discovered in the past. But we must understand that parallel processing alone could not have made MapReduce a successful programming. The framework was improved with fault tolerant clusters and scalability which has led to its success. Also, it is clearly stated that MapReduce is not a replacement for RDBMS but it is a framework for dealing with computations of large data sets where RDBMS cannot be used. MapReduce has proved to be successful and I personally feel that MapReduce programming has certain unique features which cannot be implemented in RDBMS and. To conclude, both RDBMS and MapReduce must be treated as a cooccurring systems and both are required based on the type of data sets and operations being performed.

Insights about the authors

David J. DeWitt:

David is currently a technical fellow in Microsoft Corporation and also a John P. Morgridge professor(Emeritus) of computer sciences at the University of Wisconsin-Madison. He completed Ph. D at University of Michigan in 1976 and joined as a assistant professor in University of Wisconsin. Being as a professor in University of Wisconsin he has performed various research in the areas of Parallel databases, benchmarking, Object-oriented databases, and XML databases. He has received various prestigious awards such as ACM Software systems

award, IEEE Emanuel R Piore award, ACM SIGMOD Innovations award. He started Jim Gray systems lab for Microsoft in 2008.

Michael Ralph Stonebraker:

Michael stonebraker is currently an adjunct professor in MIT. He is currently involved in development of projects such as Aurora, C-store, H-Store, Morpheus and SciDB systems.He completed his Ph.d at University of Michigan in 1971. He has received various awards such as Turning award, IEEE John von Neumann medal, SIGMOD Edgar F. Codd innovation award.Michael developed Ingres and Postgres relational databases. He is founder of various database companies such as Ingres, Illustra, Cohera, Streambase systems, Vertica. In 1994 he was inducted as a fellow of the association for computing machinery.

Mark C. Chu-Carroll

Mark is current working as a software engineer at dropbox. He is the author of Goodmath,Badmath blog.He is also creator of a language known as Tag.

References

Witt,2008.MapReduce:A major step backwards

Carroll, 2008.Databases are hammers; MapReduce is a screwdriver.

White,2015.Hadoop:The Definitive Guide: O'Reilly Media, Inc

https://en.wikipedia.org/wiki/Relational_database

https://en.wikipedia.org/wiki/MapReduce

http://gsl.azurewebsites.net/People/dewitt.aspx

https://en.wikipedia.org/wiki/David_DeWitt

https://en.wikipedia.org/wiki/Michael_Stonebraker

Figures

	Traditional RDBMS	MapReduce
Structure	Schema-on-write	Schema-on-read
Integrity	High	Low
Scaling	Nonlinear	Linear

Figure 1.1 Difference between DBMS and MapReduce

Table 1-1. RDBMS compared to MapReduce

	Traditional RDBMS	MapReduce
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Transactions	ACID	None

Figure 1.2 Difference between RDBMS and MapReduce