MapReduce Versus Relational Database Systems  
 proponents and Critics

Shankari Seethalakshmi Mohanakrishnan

Illinois Institute of Technology, Chicago

Abstract

Relational database systems is a relational model that is in the form of a table that has rows and columns and it was introduced by E.F. Codd. They really go well organized data. On the other hand, MapReduce is involved in unstructured large complex data and its calculation. MapReduce performs parallel processing. The paper deals about articles written by two authors one is a critic of MapReduce and he supports and give the advantages of relational databases. The other author gives the advantages of MapReduce and tries to give the reasons MapReduce does not or is incompatible with few relational database features. In this paper an interpretation of these two papers are made and a conclusion from the authors stand point is being made.

Keywords: MapReduce, RDB.

MapReduce Versus Relational Database Systems  
 proponents and Critics

David DeWitt (2008) has explained the concept of MapReduce pronouncing that they consist of two phases the Map and the Reduce phase. In the Map phase records are read from input and they are transformed in the form of a key/value pair. The output of the map function is partitioned into buckets with a split function which is a hash function. In the reduce phase the reduce instance considers map phase output along with hash function which is assembled by Map-Reduce framework and is given as input to the reduce phase. Now the reduce function performs various computation with records which is later written to the disk.

Mark C. Chu-Carroll (2008) states that MapReduce is a library that has a specific stylized way of programming. The idea MapReduce follows is to split the job into Map and Reduce. The input data is considered as a record in the Map phase. The Map splits the problems into sub-part and sends to different machines for processing simultaneously. The result that is obtained of the Map phase is a key value pair. Once the execution is complete the reduce phase reassembles the various chunks of data in the reduce phase.

From the money of Jeffrey Dean and Sanjay Ghemawat (2004) has stated that programs written in functional style are parallelized automatically. In case of MapReduce the data that is used as input, scheduling parallel execution, fault tolerance and communication is taken care by the run time system. This gives a major advantage to novice distributed and parallel computing users to utilize distributed resources.

# Disadvantages of MapReduce

The paper written by David DeWitt (2008) states the drawbacks of using MapReduce which are as follows:

1. Step backward for large-scale data intensive applications;
2. Implementation is not optimal as it uses brute force rather than using indexing;
3. Implements techniques is not novel;
4. DBMS features are missing; and
5. Incompatible with DBMS tools.

On the other hand, Mark C. Chu-Carroll (2008) contradicts the views stated above and he applauds MapReduce for what it has. Let’s discuss in brief each of the impediments and what their supporters state for the above disadvantages:

## Step backward for large-scale data intensive applications

Database access is a step backward in MapReduce as it does not implement the concepts a database must follow like schema, schema separation from application and the need for high level access language as given by David DeWitt (2008).

The schema is important as the field and their respective data types are preserved in the storage. This would ensure records that are entered obey schema rules and remove bad record. MapReduce does not have any such schema implementation to determine bad record.

Separation of schema and the database is required mainly when a programmer create a new application which is present in database. In case of MapReduce the schema is either present along with the application or is not present at all. This would become difficult for a programmer to examine as they need to go through the code to get to know the schema of the database.

DBMS uses high level languages and relational systems as they are easier to write, modified and are easily understandable. On the other hand, Map reduce is a language that is like an assembly language which is a low-level language.

Mark C. Chu-Carroll (2008) projects that for large data intensive application which is in tabulated format with proper indexing and when the machine that handles this large data set has enough capacity to handle this data then RDBMS would be the best application to use. On the other hand, when there are complex computations which cannot be run in a single system and when parallelization is mandatory then MapReduce would no doubt lead the crowd.

## Implementation is not optimal as it uses brute force rather than using indexing

As per the paper published by David DeWitt (2008) databases use data structures like B-tree for their indexes and query optimizers to speed up the access of data. MapReduce on the other hand uses a brute force method for performing their processing as they do not have the concept of indexes.

But an interesting fact to be considered about MapReduce is that it provides parallel execution on multiple computers simultaneously. To disapprove this a group of professionals did multiple prototyping with DBMS and they proved it worked with better performance and it was more commercial.

One more important problem to consider with MapReduce is data interchange and skew problem. This issue mainly affects the performance in the map phase when there is distribution of records with the same the same key as given by David DeWitt (2008) in his paper. This in turn leads to a situation where the execution time being the running time of reduced instance.

From Mark C. Chu-Carroll (2008) gives his point that in case of relational database only indexes are present but all data computation in real world would not be in relational format with indexes. In such situation, MapReduce plays a major role by performing calculations for large scale in limited time.

## Implements techniques is not novel

The implementation techniques used in MapReduce is 20 years old and is not something that is discovered. The various discoveries claimed by the MapReduce team are:

* Partitioning large datasets into smaller partition;
* Executing joins in a parallel shared nothing cluster by using concepts like parallel execution, partitioned table and hash based splitting; and
* Executing aggregate function with or without group by function in parallel systems.

But the DBMS people disagree with this as they have sold a product called Teradata that already has implemented all the features that the MapReduce team argue to have discovered.

On the contrary, MapReduce never claimed to be novel as stated by Mark C. Chu-Carroll (2008). He adds to his point that MapReduce is a type of parallel computing which is easy to use model and has a scalable implementation.

Mark C. Chu-Carroll (2008) contradicts the point by stating that these concepts mentioned in the above paragraph is implemented only in a relational database whereas MapReduce does not work out with relational databases. MapReduce is designed for high computations like cloud and cluster computing.

## DBMS features are missing

The features that MapReduce is missing which database systems already have are bulk loader, Updates, transaction, integrity constraint, referential integrity and views. The bulk loader is used to change a file into a desired format and load it into the DBMS. Updates change the data in the database. Integrity and referential constraint help to drop bad records from the database. Views help to change schema without changing the application.

According to Mark C. Chu-Carroll (2008) relational databases only support DBMS tools. MapReduce is a non-relational thus it cannot implement the features of DBMS.

## Incompatible with DBMS tools

MapReduce is incompatible with tools like:

* Business Intelligence tools;
* Data mining tools;
* Replication tools;
* Report writing tools; and
* Database designing tools.

From the books of Mark C. Chu-Carroll (2008) he pronounces that all these are properties of a relational database and he also adds that how can other software or tool have all the properties of the other software.

## Advantages of MapReduce over Relational database

As stated by Mark C. Chu-Carroll (2008),

* **Parallelization**: Parallelization in RDB is not as in the level of MapReduce. RDBMS cannot split task over multiple computers as MapReduce can do;
* **Recursion:** RDBMS cannot work on recursion well;
* **Database Models:** RDBMS works only with relational databases for non-tabulated database MapReduce is better.

# Conclusion

From my books, MapReduce is a boon although it has certain drawbacks like:

* Implementation difficulty;
* Processing graphs;
* Real time processing;
* When implementing complex algorithms; and
* When processing repeated set of data again and again with the same computation.

But anything in this world like a coin has two phases the positive and the negative. When compared to the negative I consider the positives of using MapReduce is higher. The advantages of MapReduce are:

## Scalability

MapReduce is highly scalable. This it manages by distributing data sets across many servers which process in parallel. More the number of servers more the power of processing. This aids MapReduce to handle thousands of terabytes of data. Contradicting to this the RDB cannot handle increased data thus lacking scalability.

## Cost Effective

MapReduce programming is used to scale out architecture for Hadoop. This effectively stores and processes data. Apart from this data could also be available at a later point. The cost reduction is also massive when using MapReduce. In case of traditional database systems scaling of data becomes very costly hence business down size it. But this could create problems in a later stage.

## Security and Authentication

HDFS and HBase security work along with MapReduce and allow only authorized and authentic users to access data that is present in the system.

## Parallel processing

In MapReduce, parallel processing take place by dividing the task to multiple processors simultaneously. This reduce the time required to process the data.

## Flexibility

MapReduce can work with data from various sources, the data could be either structured or unstructured. Thus, creating value to all the data that they encounter. But only structured data could be handled by relational database systems.

## Speed

Hadoop uses a distributed file system which implements a mapping for locating data clusters. In this case MapReduce is the program that is used which runs on the same server that improves process speed such that it just takes minutes to process terabytes of data and hours for processing petabytes of data.

## Fault Tolerant and Available

In case of MapReduce when data is forwarded to a node they are multiplied and sent to another couple of nodes in the network. Replicas for the original copy is made to handle failures. Apart from this Hadoop offers an automatic recovery technique for faults.

## Simple programming model

The MapReduce programming is very simple which enables developers to write programs that can handle multiple tasks with ease. Data processing needs met efficiently by using MapReduce for programming.

In case of having a structured data, which is in a relational format and when the results are to be obtained in real time then relational databases would really do fine but on the other case when data is not structured which requires lot of processing and when the data must also be handled safely then MapReduce would be a better option. Thus, it mainly depends on the business requirement which type of database could be chosen.

References

Jeffery Dean. and Sanjay Ghemawat. (2004). MapReduce: Simplified Data Processing on Large Clusters. *OSDI conference*, Pages 1-13.

David Dewitt, J. (2008 January 17).*Map Reduce: A major step backwards*, Retrieved from http://homes.cs.washington.edu/~billhowe/mapreduce\_a\_major\_step\_backwards.html.

Mark Chu-Carroll, C. (2008 January 22).*Database is a hammer; MapReduce is a screwdriver*, Retrived from http://scienceblogs.com/goodmath/2008/01/22/databases-are-hammers-mapreduc/.

MapReduce. (n.d.). Retrived from https://en.wikipedia.org/wiki/MapReduce.

SQL - RDBMS Concepts. (n.d.). *Journal Title*, Retrieved from https://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm.

Advantages of Hadoop MapReduce Programming, F. M. (n.d.). Retrieved from http://www.tutorialspoint.com/articles/advantages-of-hadoop-mapreduce-programming.

Tom White. (2015). *Hadoop: The Definitive Guide.* California: O’Reilly Media.

**Appendix A**

David DeWitt is professor of computer science department in University of Wisconsin-Madison. He is the founder of Wisconsin Database Group in his university. He has done research in various fields of database like parallel, XML, benchmarking and object oriented databases. He is currently working as a Technical Lead at Microsoft.

Mark Chu-Carroll has obtained his doctorate degree from University of Delaware. He has worked as a research staff member at IBM Research in software configuration management compilers and in programming languages. He has worked as a senior software engineer in google following which he has worked as a server software engineer at Foursquare Labs. He is currently working now in Dropbox as software engineer.