MapReduce vs RDBMS: Is the debate Worth It?

Arjun Gupta

Illinois Institute of Technology, Chicago

Abstract

In the background of the release of the notion of MapReduce and Cloud Computing, 2 people, David DeWitt and Michael Stonebraker released a paper expressing their views on the new developements in the field of technology. They were very crticial of MapReduce and denounced the notion by giving five supporting reasons for the same.

Mark C. Chu-Carroll, an employee of Google wrote a paper expressing his response to David DeWitt and Michael Stonebraker's paper. He gave justification to each reason given by David DeWitt and Michael Stonebraker to outrightly reject MapReduce and towards the end of the paper, Mark C. Chu-Carroll virtually mocked the proponents of RDBMS with technological facts supported by sheer sarcasm.

Using this paper, I have summarized both the papers presented by David DeWitt, Michael Stonebraker and Mark C. Chu-Carroll. I have also given my views on why I support 1 paper and reject the views of the other paper.

Keywords: MapReduce, RDBMS, Mapper, Reducer, Key-Value

SUMMARY OF FIRST PAPER

Paper 1 has been authored by David DeWitt and Michael Stonebraker on January 17, 2008 in the backdrop of the news about a new revolution about “Cloud Computing”. The authors talks about how big companies and educational institutions have used the concept of the new revolution to their best efforts. For eg. IBM and Google have reported arrangements to make a 1,000 processor cluster accessible to a couple select colleges to show understudies how to program such bunches utilizing a product apparatus called MapReduce. Berkeley has gone so far as to anticipate showing their first year recruit how to program utilizing the MapReduce system. Both the authors have also expressed their amazement about the buildup that the MapReduce advocates have spread about how it speaks to an outlook change in the advancement of versatile, information serious applications.

However, both the authors have used this paper to express their outright rejection of the new concept of “cloud computing” and more importantly, “MapReduce”. They have given 5 reasons for their stance against “MapReduce”. These 5 reasons are as follows:

A mammoth stride in reverse in the programming worldview for huge scale information concentrated applications.

An imperfect execution, in that it utilizes savage compel as opposed to indexing.

Not novel by any means - it speaks to a particular execution of understood methods grew about 25 years prior.

Missing the vast majority of the elements that are routinely incorporated into current DBMS.

Contradictory with the greater part of the apparatuses DBMS clients have come to rely on upon.

The authors have first talked about the concept of “MapReduce” itself and then have elaborated upon the five reasons for outrightly rejecting the new data paradigm.

The authors have drawn an analogy between MapReduce and SQL to actually explain what MapReduce is. The analogy is as follows. The MapReduce contains of basically 2 phases, Map Phase and Reduce Phase. The Map Phase is like the “Group-by” clause of SQL and the Reduce Phase is like the “Aggregate function over a set of records”. The authors begin their explanation by categorising the 2 phases of MapReduce, ie. Map Phase and Reduce Phase. The Map program reads an arrangement of "records" from an info document, does filtering or potentially changes, and afterward yields an arrangement of records of the shape (key, data). As the Map program produces yield records, a "split" function segments the records into M disjoint containers by applying a function to the key of each yield record. This split capacity is normally a hash function, however any deterministic function will work. At the point when a basin fills, it is composed to the disk. The map program ends with M output records, one for each bucket. Basically in the event that N nodes take an interest in the map stage, then there are M records on disk storage at each of N nodes, for a sum of N * M documents. Like the map program, the reduce program is a subjective calculation in a universally useful language. All output records from the map stage with a similar hash value will be devoured by the same reduce occurrence regardless of which map instance created them. In the wake of being gathered by the map-reduce system, the input records to a reduce example are assembled on their keys (by sorting or hashing) and sustain to the reduce program. This is the whole concept about “MapReduce” as explained by the authors in this paper.

Moving on, the authors now explain the rationale behind their five reasons for rejecting “MapReduce”.

In the first reason, the authors claim that MapReduce has not learnt anything which the Database Community learnt from the three lessons from the 40 years that have unfurled since IBM initially discharged IMS in 1968 about schemas and high level access languages being good or about the benefits of of keeping the schema away from the application. The reason for keeping the schema away from the application according to the authors is that if a new code or logic is to be built, the concept can be built using just the schema. In case of SQL, the schema is well defined and kept away from the application for this very purpose only which sadly cannot happen in MapReduce. Finally the MapReduce has been compared to a Codasyl view as well by the authors. In the second reason**,** the authors claim that almost all modern DBMSs use the concept of hashing or indexing using B-tree to access data. The authors claim that MapReduce does not use the concept of indexing and thus relies on brute force for processing the data. Explaining the third reason, the authors accuse MapReduce of not being a novel idea and claiming that the idea is more than 20 years old. In the fourth reason, the authors say that MapReduce misses critical features like indexing, Integrity Constraints and referential integrity. In the final reason**,** the authors criticise MapReduce for not being compatible with DBMS tools.

In a nutshell, even though the authors do acknowledge that DBMS and SQL does have limitations and that those concepts are not full proof, they renounce the idea of MapReduce in totality.

SUMMARY OF SECOND PAPER

The second paper has been written by Mark C. Chu-Carroll on 22nd January 2008. This paper is a sort of a reply to the paper written by David DeWitt and Michael Stonebraker. The paper begins with a declaration that the views are of the author only and that the company the author works for does not endorse any views presented. Like the previous paper, this paper also begins with explaining what MapReduce actually is and then counters the 5 reasons listed in the first paper.

The paper firstly talks about the benefits of using MapReduce. The author says that the beauty of the language used by MapReduce is that if we have a bunch of machines which are not doing any task, those machines can be used and the stylised way of programming gives us ease in splitting the task among those bunch of machines.

The paper then talks about what MapReduce actually is. It informs the reader that MapReduce is a two-phased paradigm, Map Phase and Reduce Phase being the two phases of the concept. The Map Phase reads in the data and splits it into Key-value pairs which is then passed onto the Reduce Phase which basically combines the Key-Value pair into a single value. How many key-value pairs were generated in the process cannot be determined in this case.

The author moves forward in the paper to discuss the shortcomings of DBMS by telling a story narrated to him by his boss. The story is about how RDB people have developed a Hammer which they feel is ideal for them. They think that all tasks related to DBMS can be solved using that hammer and they have aced the concepts related to that hammer.

The author after telling the story then acknowledges the beauty of Relational Databases by himself admitting that he admired RDBMS and has worked with the tools of RDBMS.

But the main criticism of RDBMS begins now. The author exclaims that everything is not relational databases. There is more to computing that just databases. The author talks about the fact that MapReduce was never made to REPLACE RDBMS. It is just an alternative for RDBMS by doing the same task on parallel machines. The author then counters each point given in the first paper as a reason for rejecting MapReduce.

The author begins with the first reason which says that MapReduce is a step backwards in the world of programming. The author ridicules this notion that MapReduce is a step backwards in the world of programming. Infact, the author lauds the idea of MapReduce and admits that for a large dataset, MapReduce is better than RDBMS in terms of time taken to get the desired output by running the tasks on parallel machines instead of one machine which is the concept used by RDBMS. The author then explains the concept of MapReduce by giving an example of image of related fractals called NebulaBrot created by a reader. He explains how the reader began by considering points and then tracing those points. If a path was crossed twice, that path would be darkened. However the end image is not zoomable and the path taken cannot be determined. However the final image is very cool according to the author. The author drew the analogy of this example being just the process used by MapReduce. The initial part of beginning with points, splitting many points into groups and then tracing the points in one group. It is like the Map Phase. The resulting output was a set of points(keys) and the number of times a path (value) was crossed. The author then talks about why this task cannot be performed using RDBMS because RDBMS needs tables, rows and columns and this example has none. It just has a set of points. The author then moves onto the second reason which states that MapReduce is a primitive concept and does not use the concept of indexing. The author uses the above example to counter this reason as well. The author states that to use indexing, one needs to have some sort

of schema having rows, columns and tables and not every problem has all(or any of them for that matter) of them. The author also claims that for huge dataset, indexing is not good as the time complexity is very high and MapReduce resolves the problem of time complexity while RDBMS does not. Moving onto the third reason, the author ridicules the idea behind this reason. He assures that MapReduce was never meant to be a novel idea. Infact, it was inspired by RDBMS and works on the concept of parallel programming. Then the author counters the fourth reason which talks about MapReduce missing the vast majority of the elements that are routinely incorporated into current DBMS. The author states that no one has compelled anyone to use MapReduce if the task is of RDBMS and the tools of RDBMS are present. The only problem according to the author about RDBMS is that RDBMS is not built for huge data, data that is of size running into terabytes and petabytes. Finally the author talks about the final reason and virtually mocks the proponents of RDBMS. This is because the fifth reason talks about the incompatibility of RDBMS with MapReduce. The author reiterates his story about the hammer and exclaims that the proponents of RDBMS cannot think above RDBMS.

The author ends the paper with criticising the proponents of RDBMS and says that they must think beyond the ‘hammer’. They must realise that RDBMS is not the ONLY tool to get the desired outputs. MapReduce is a great concept which is time saving and more importantly uses a programming language which is very simple.

MY VIEWS

To be really honest, the view presented by David DeWitt and Michael Stonebraker looks a very childish view to me. It is as if the authors have been blinded by the whole concept of RDBMS. It seems they are not willing to accept a new concept. I feel they do accept that RDBMS has its own limitations. Ofcourse, nothing in this world is full proof. But the fact is that the reasons given by them in voicing their dissent against MapReduce are not up to the mark.

They biggest reason for their dissent is that they feel MapReduce is not a novel idea and that it is incompatible with the tools of RDB. The interesting thing to note is that MapReduce was never made as a REPLACEMENT for RDB. MapReduce never CLAIMED to be better than RDBMS. It is as if the HAMMER that the proponents of RDBMS cannot match up to MapReduce and hence the its proponents are rejecting MapReduce.

Having worked on MapReduce as an assignment in my class in the current semester, I understand the benefits of MapReduce. I do understand that these days, data is in terabytes and petabytes. And data is not necessarily in the form of rows and columns. Hence RDBMS cannot solve any questions that arise on that data. However if MapReduce is used, we can get desired output easily. One reason is because of the sheer enormous size of the data which RDBMS cannot manage.

Hence I agree with Mark C. Chu-Carroll's stance and his paper on this issue.

References

David J. DeWitt, Michael Stonebraker (2008), MapReduce: A major step backwards

Mark C. Chu-Carroll (2008), Databases are hammers; MapReduce is a screwdriver.

Appendix A

A FEW WORDS ABOUT MICHAEL STONEBRAKER

Michael Stonebreaker's contribution to the field of databases cannot be ignored. Infact the contribution is massive. He worked with a colleague to develop and implement a practical implementation which was called INGRES. This was one of the main frameworks (alongside System R from IBM) to exhibit that it was conceivable to construct a reasonable and proficient execution of the social model. Various key thoughts from INGRES are still broadly utilized as a part of social frameworks, including the utilization of B-trees, essential duplicate replication, the inquiry change way to deal with perspectives and honesty limitations, and principles/triggers for uprightness checking in a RDBMS. Furthermore, much exploratory work was done that gave bits of knowledge into how to construct a locking framework that could give acceptable exchange execution.