A Comparison of Approaches to Large-Scale Data Analysis

Magnus L Kirø

November 15, 2012

Abstract

The article looks at the two methods of managing data. MapReduce and Parallel Database Management Systems. They use three different systems for testing , Vertica, DBMS-X and Hadoop. Results from the tests are displayed and at the end they conclude with their findings. The findings of the article is somewhat narrow and only sheds light on the descibed environment.

Two Approaches

MR is a techneque to store data i a base. It has two functions. Map and Reduce. This makes MR simple. The simplicity is what makes MR attractive. Basically it's like throwing data in a bucket and read it when needed.

The mapping cunction, **Map**, map the data into files that are stored in the underlying distributed file system. The reducing function, **Reduce**, compiles the output data from a mapping function to creata a combined result to a query. The two functions has to be implemented by the eveloper. This is one of the negative points about MR.

The main aspects of a parallel DBMS, compared to the MR method, is that the tables are partitioned across nodes and that a query optimizer is available. The query optimizer makes sure that the query given to the DBMS is executed as fast as possible. The optimizer can also spliti queries into sub parts to distribute the work load on the given cluster. When a query is run the distributed execution works on the underlying storage device whitout any interaction from the developer.

The main difference in schema support is that MR don't have it, wile DBMS has it. For MR this means that the developer have to handle data consistency enforcement. While in a DBMS the data rules are set on system setup. Queries on a DBMS that violates the data structure are disregarded.

Indexing is provided in a DBMS, while it has to be explicitly implemented in a MR system.

MR and DBMS has different styles of datastorage. Historically the two main variants was Codasyl and Relational. MR is of the Codasyl style, while DBMS is a relational database.

The datadistribution of the two types of databses are quite different. MR works on the basis of retrieving all the data, then calculate the wanted result. While DBMS pushes queries to the nodes for distributed execution. When the distributed results come back the total result is complte.

A bit like the data distribution, the execution strategies ar divided. MR pulls data from the nodes and DBMS pushes data to the nodes that push data back. This can be a potential performance reducer for MR, as there is a lot of data that has to be transfered on the network.

The general division of flexibility is: MR is flexible, DBMS is strict. With MR you can do nearly whatever you want, but you have to enforce the integrity of the data yourself. DBMS is regarded as strict and limited. mostly because the data structure can't be changed easily after initiation. Although this is not often a problem due to the good support for DBMS's.

As for fault tolerance, this is where MR really gets into the game. When MR distributes a lob to a node, that job i tiny. If the node fails the tiny job is just redistributed to another node. In a DBMS the whole transaction would faile and the time lost i quite significant.

Benchmark

In the test environment the article describes the three systems, Hadoop, DBMSX and Vertica. The main difference is that Hadoop was run without compression, while thw two others was. All taske was executed three times. And the tested systems was optimized for each task.

The first task, Grep task, is a standard linux grep command. All files containing the given argument is returned. The data for this task was distributed with 100byte records, 10byte key element and 90byte random data. The serached for element was found once in every 10.000 records. hadoop showed a significant startup cost of the system, while DBMX-X and Vertica was easy to load data, and the startup time was quite good.

Selection was the next task. The data consisted of 36.000 records per file, per node. Hadoop is outperformed by the DBMSs. Mainly because Hadoop finishes the data extraction so quickly that a torrent of control messages delays the system in compiling the final result.

Much the same happens with the Aggregation task. Hadoop is outperformed again, but this time we can see that vertica slows down. Although Vertica only reads the collumns needed to create the answer, we have much fewer I/O operations i Vertica.

The joining task was to compile a page ranking given a time period. In MR this is a complex operation with three phases. The implementation time

was considerable. While on both system types I/O was the most significant part of the execution time.

UDF Aggregation. The task of counting links i documents. Vertica is fastest, while DBMS-X and Hadoop ha close to constant execution time. The systems slows down when they write the results to file.

Discussion

The install process was somewhat a bother for the authors of the article. The Hadoop setup was feirly straight forward. Get file, install. Although the tuning had to be done manually by trial and error. And the system had to be tuned individually for each task. DBMS-X and Vertica had quite streight forward installs. But the configuration and tuning was not that easy, DBMS-X was rater difficult to tune and Veritca has auto tuning capabilities that often don't make sense. When tuning Vertica, one parameter was changed and some other ones was change automatically. As for DBMS-X some of the configuration crashed the system comletely. Change a parameter and restart, the system might not start again. Bad documentation was a contributing factor to the difficulty of configuration. Reclecting on the complexity of installation Hadoop scores above all the others. Strangely enough newer open source project are often like this.

As for the task startup delay, the Hadoop system had a 10-15% decrease in startuptime by reusing the JVM's. But still MR uses 10 seconds to distribute the task completely, and not all the nodes are working before 25 seconds. Historically the startup time of reguler DBMSs have been improved continously the last 25 or so years. So it's no surprice that a DBMS has faster startup times then systems like Hadoop. That said, the article was written i 2009 and I will presume that there have been significant improvements the last three years regarding startup time and otherwise.

Compression was a roblem in the Hadoop system. Hadoop ran faster without compression, quite as expected really. But what I didn't anticipate was that DBMS-X and Vertica was faster with compression than without. After a while it becomes aparrent that a smaller amount of data is processed faster then a large amount and the surprice was turned up side down. Why would Hadoop work faster on a larger amount of data? This is one of the points that certainly has been changed the last three years. One of the reasons for Hadoops lacking compression support quite certainly has something to do with the age of the system. Hadoop was not that old in 2009 and the amount of time that has passed to perfect the system is minimal. And definitly not comparable to DBMS.

Loading of data was quickest in Hadoop. Much because of the way the data is loaded. Pour the data in a bucket an be done, no data checking or anything. Negative for Hadoop is that it is CPU intensive. It uses a lot

of system resources. DBMS can reorganize data while loading them. This can in many cases be quite useful. But also quite slow. In total DBMS was beaten badly when loading data, from the system and to the system.

In regard to the execution strategies Hadoop isn't mature enough. Hadoop pulls all the data from all nodes and then compiles a result. In contrast the DBMSs creates a query plan and then distributes the query to the nodes that compile sub answers. The returned answers can then be mashed into the same result set. Boiling it down we have Hadoop with a pull strategy and DBMS with a push strategy. The push strategy has less overhead and provides for better data efficiency. Data pushing strategies widely considered in the telecomunications filed to lessen the data traffic on telecom nets.

Failuers occur. Frequently. Companies like Google and Facebook looses something like a server a day and a HDD an hour. This is grim numbers for data management. It means that the system fails almost all the time. So it's very important to have good fault tolerance and good techniques for handeling the errors. Here MR and DBMS have one point each. MR has the faul tolerance, while DBMS has the supperior error handeling techniques. When a node crashes in a MR system, the part of the job delegated to the crashed node is redistributed to another node. Only a tiny bit of the total execution is lost. DBMS often work in transactions. When a node crashes the whole transaction have to run again. This can in some cases be a tiny amount of time, while in other cases the amount of time lost can be huge. The good thing about DBMS is that they have had time to complex and quick error recovery methods to improve execution time and error recovery rate. MR systems could benefit a lot from error recovery methods from DBMS.

Ultimately the decision of system choice is made from ease of use. Is the system hard to set up and hard to use, it's disregarded completely. While systems that are plug and play styled, one click to work style, is greatly favored. Many open source systems are like this. Including Hadoop. The authors of the artifcle describes Hadoop as having a great comunity and good installation guides. When DBMS-X and Vertica was not nearlyt as easy to set up and use. The simple structure of Hadoop was also favored. While the strictness of DBMS might be preferred in smaller systems where the amount of data is small. And in systems that has to maintained on a low budget with few people. A strict set of rules on the data makes it easy for the developers to conform to the data model.

Tools of the trade is another factor that Hadoop looses. Hadoop being young has not yet had time to gain ground in the addons section. DBMS implementations has been around for a long time and therefore has quite an arsenal of user friendly and useful tools to help developers.

Conclusion

To wrap up the article nicely in accordance to the message in the article would be naive. The article greatly prefers DBMS. And to some extent I think the article tries to tell the reader that DBMSs are supperior to MR approaches to databases. This ofcourse greatly undermines the fact that the two approaches has different areas of use. The areas might be close, but the difference is huge. Quite literally huge. DBMSs work best with small and not to large amounts of data. While the MR approach is designed to deal with massive amounts of data. The increase of data storage today indicates that a approach that can handle petabyte after petabyte with data is the way to go. While lesser systems, containing smaller amounts of data, is way easier to handle with a DBMS.

From the situation in the artcle MR systems such as Hadoop can gain quite a lot by adopting methods for error correction and other metods from the DBMS realm. In the time after the publication of the article Hadoop and other MR implementations has certainly improved massively and will probably match DBMS execution times in a foreseeble future. DBMSs will maintain their usefulness and status as a small scale, quick and maintainable system for data storage.