A Comparison of Approaches to Large-Scale Data Analysis

Magnus L Kirø

November 14, 2012

Abstract

This is the paper's abstract ...

Two Approaches

Map Reduce (MR)

MapReduce(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.

Parallel Database Management System (DBMS)

1. Tables are partitioned across nodes
2. Query optimizer, that translates SQL to a query plan. Execution of the query plan is divided among multiple nodes.
3. Underlying storage details can be disregarded by the programmers.

Schema Support

1. MR does not have Schema support. Manual data integrity enforcement is required.
2. DBMS has Schema support. Data integrity is automatically enforced by the schema.

Indexing

1. MR does not have inbuildt indexing. Again the programmer has to implement it, if the functionality is wanted.
2. DBMS provides indexing.

Programming Model

1. MR, Codasyl style, provide an algorithm to get the data you want.
2. DBMS, Relational style, state what you want.

Data Distribution

1. MR: get all documents, then compute the result.
2. DBMS: distributes code to all nodes, the nodes compute partial answers, answers are combined into the result.

Execution Strategy

1. MR: Pull data. Nodes*Maps files - potentially a severe performance problem.
2. DBMS: Push data.

Flexibility

1. MR has the most flexibility. You can do nearly whatevery you want. But you have to enforce your own rules.
2. DBMS is strict and limited, but comes with great support after a long development time and lots of use.

Fault Tolerance

1. MR: Node crash - task is recheduled to another node. Only that subtask is lost in computing time.
2. DBMS: Node crash - the whole transaction has to be restarted. Might be very expensive.

Benchmark

Environment

1. Hadoop, DBMS-X and Vertica.
2. Hadoop whitout compression. The rest with.
3. Task execution: Each task was executed three times.
4. All systems was optimized for the tasks given.

Grep Task

1. Scan all files for a string pattern.
2. 100byte records, 10byte key, 90 byte random data. once in every 10.000 records.
3. Hadoop: Command line to copy data to FS. Significant startup cost.
4. DBMS: Hash aware load data.
5. Vertica: Provides a copy cmd.

Selection Task

1. 36.000 data records per file on each node.
2. Hadoop: Fisnishes so quickly that a torrent of controll messages increases the total execution time.
3. Again Hadoop is outperformed by the other two.

Aggregation Task

1. Task: calculate total revenue by IP.
2. Produces 2.5 million records(53MB) and 2.000 records(24KB).
3. Vertica slows down. But does not read unnecessary data columns.
4. Hadoop: finds all elements of correct type, then sums up the results.

Join Task

1. Task: Page rankings in a time period.
2. Complex MR program with three phases.
3. Reading and processing data is the most time consuming.

UDF Aggregation task

1. Task: Counting links in documents.
2. DBMS-X and Hadoop has close to constant execution time.
3. Result writing gets slower with increased number of nodes.

Discussion

Install

1. Hadoop: Easy install, trial and error optimization. Task tuning.
2. DBMS-X: Straight forward install. But the configuration proved difficult.
3. Vertica: Quite easy install. But too automated tuinig capabilities.

Task Startup

1. MR: 10 sec until the task is distributed. 25 sec for all nodes to start executing.
2. Hadoop reuse JVM reduced startup time by 10-15%
3. DBMS: startup time was one of the first things that was improved.
4. Resent improvements (article from 2009).

Compression

1. Both DBMS-X and Vertica worked better with data compression.
2. Hadoop worked better without compression.

Data Loading

1. Hadoop was the best system to load and read data.
2. Hadoop was more CPU intensive.
3. DBMSs can reorganize data on load.

Exectuion Strategies

1. Hadoops overhead messaging slowed it down.
2. DBMS data push strategy
3. DBMS query plan.

Failure Models

1. More HW = more failures.
2. MR is more tolerant to failure.
3. Sophisticated error recovery could improve performance.

Ease of Use

1. MR(Hadoop) was easier to get up and running. Simple structure. But algorithms have to be implemented.
2. DBMS: might be easier to maintain later. Less data enforecement to do.

Additional Tools

1. DBMS have a long history of development and have a lot of extrernal tools to use.
2. MR is still young so there is not to many tools available yet.

Conclusion

Summary of solutions and drawbacks

1. Small scale data analysis will work better with DBMSs.
2. Large scale data analysis today is way bigger then it was in 2009.
3. Hadoop and MR systems has room for improvement and will probably be improved over time.
4. Both architectures will probably remain, due to their different strenghts and areas of use.