Big Data Summary:

MapReduce: Simplified Data Processing on Large Clusters
Jeffrey Dean & Sanjay Ghemawat

A Comparison of Approaches to Large-Scale Data Analysis
Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden & Michael Stonebraker

Melissa Iori, Dec. 4 2014

MapReduce

- The MapReduce programming model merges large data sets of key/value pairs by taking advantage of redundant keys, making the data easier to query and parse.
- Rather than being strictly defined, MapReduce users must create their own Map and Reduce functions specifically to handle their own data.
- The map function provides a series of “intermediate” key-value pairs.
- The reduce function merges the pairs that have the same key.
- Example output of a wordcount map function after reading in some input.txt file of words where {word, intermediate_count} : {“hello”, 1}, {“world”, 1}, {“world”, 1}, {“hello”, 1}
- These pairs are sent to the user’s reduce function and are output as:
- {“hello”, 2}, {“world”, 2}
- This cuts the total number of pairs in half by taking advantage of redundancies.
- There are many real-world applications including machine learning, document and URL counts, rankings, etc.

Implementation: Distributed Processing

- Huge data sets like those managed by Google and Facebook can be many TB in size.
- Having one computer manage and query all of this can be expensive (for one computer that powerful), time wasting if that computer is not fast enough to handle the amount of data, and error prone if that one computer fails.
- Using distributed processing, MapReduce can be implemented across multiple machines.
- One instance runs on a computer defined as the “Master”. The Master has the job of connecting to other machines in the cluster on which MapReduce is running, and gives out either a Map task or a Reduce task to an idle machine, now called a Worker.
- When the Map Workers have their output, they send it along the network to a Reduce Worker. The Reduce Worker will create the final output and store it in the distributed filesystem.
- If a machine fails, the Master simply re-assigns the incomplete Map or Reduce tasks, handling errors gracefully.
- Because tasks are treated as “atomic commits” of work, the Map or Reduce is never left “half done” and only has to be re-assigned completely to another machine, reinforcing consistency of the entire job as if just one computer were creating the output.

Analysis

- Unlike PIG or HIVE, MapReduce is not a standard language or platform, but rather a model for handling key-value input and providing a more organized output. I wondered how loosely MapReduce could be defined, and about examples in languages other than C.
- Most of the paper focused on the distributed implementation, which was a really smart way to handle huge inputs and outputs, but it is lacking in defining this data model, what restrictions are on it, and more on how it works.
- The idea of unstructured data and key-value stores seems to integrate nicely with modern web frameworks (JSON, MVC (model-view-controller) etc.) as well, but like with NoSQL, there are serious considerations about ACID compliance.

Comparison of MapReduce to the Relational Model

- A later paper, “A Comparison of Approaches to Large-Scale Data Analysis”, responded to the claims of the MapReduce paper by questioning its veracity in the face of distributed relational platforms.
- The authors conducted an experiment using the SQL-based DBMS-X and Vertica. They compared the performance of these platforms to Hadoop, which is based on MapReduce.
- This response paper finds that the relational platforms performed faster in a distributed environment under common settings, and further argued that said platforms are stronger because of their ACID compliance and agreed-upon system catalog which provides consistent indexing and formats, unlike MapReduce which is user-defined.
- In short, they argue that parallel DBMS has existed for some time, rather than being this “new” thing that SQL and relational systems cannot handle. And, they proved that the relational performance is already just as good or better than MapReduce. Hadoop’s startup phase often causes its total query time to lag far behind DBMS-X and Vertica.

Advantages and Disadvantages of MapReduce

Advantages:

- Simple and easier to integrate with modern web services (JSON etc)
- Can be easily custom-tailored to a specific need
- May be easier to comprehend than SQL for programmers who started on languages like C

Disadvantages:

- Must scan entire input file on each query, which is time-consuming
- No system catalog / indexes
- No agreed-upon format for Map and Reduce functions
- Compression does not speed up Hadoop

Overall, the relational model will always be more consistent, but MapReduce is still an interesting project that will hopefully see improvements.