**OS Project summary on MapReduce**

**By Emmanuel Nimo**

**Paper:**

**MapReduce: Simplified Data Processing on Large Clusters**

**by Jeffrey Dean and Sanjay Ghemawat**

Introduction.

Mapping is one of the very important techniques in data processing, especially when it comes to input and output related functions. MapReduce is a model designed to make programmers do a much easier work on large raw data. The algorithm can be derived from the name itself; it involves mapping the data set into a different set and reducing it into another smaller form. The paper shows how interesting MapReduce can help in creating large data set for tech companies or companies that need it. Examples of such data that were processed by tech companies such are Google are inverted indices, various representations of the graph structure of web documents, summaries of the number of pages crawled per host, the set of most frequent queries in a given day. These data were too large and required complex computations. Because the input data was too large, and computations must be shared across hundreds and thousands of machines to finish the computation within a reasonable amount of time. This led to issue in terms of parallelization computation, and distribution of data. To handle this, there were large complex of code written to control these issues, but they were inefficient. The solution was MapReduce algorithm, and in this algorithm, users can define a map function that possesses a key/value pair to generate a set of intermediate key/value pair that is reduced in the end to fit into memory. This enables users to process huge data set in parallel, on a large cluster of commodity machines in an efficient manner.

Body

In this section, the author describes how MapReduce creates the platform for people to do the computation of large raw data. The implementation of the MapReduce was run on a large cluster of commodity machines which proved very scalable. Due to this, a lot of programmers love to use MapReduce model because it is easy to scale data processing over multiple computer nodes.

Programming model.

The authors simplify the understanding of MapReduce in the programming model section of the paper. According to them, there are two basic functions: Map and Reduce. The map is always written by a user and it takes an input pair and produces a set of intermediate key/value pairs, and they are grouped. The result is passed through a reduce function, which is also written by the user, it accepts the immediate keys and set of values for the key. It then merges together these values and to form a smaller set of values. The intermediate values are then returned to the users reduce function via an iterator. In so doing, we can handle lists of values that are too large to fit in memory

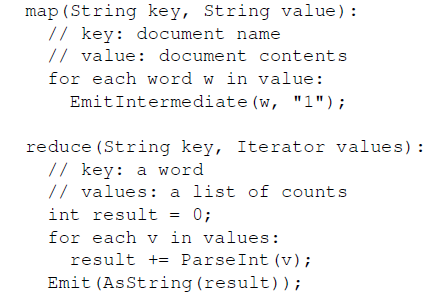


Figure 1. A pseudocode of the MapReduce function.

The above pseudocode was followed to write the code to implement the MapReduce model. Suppose you were to count the occurrences of each word in a large collection of document. The map function emits each word plus an associated count of occurrence, and the reduce function sums together all counts emitted for an emitted word. You then do the specification, thus, write a code to fill in a map reduce specification object with the names of the input and output files, and optional tuning parameters.

Type: While input and output were strings, the map and reduce function had its own type and its below.



You may be wondering about some of the programs that can be expressed as MapReduce computation. In the paper, the following were cited as examples:

1. Distributed Grep
2. Count of URL Access Frequency
3. ReverseWeb-Link Graph
4. Term-Vector per Host
5. Inverted Index
6. Distributed Sort

Implementation

According to the authors, there are different implementations of MapReduce depending on the user’s choice. They touched on an implementation used at Google and their environment included

1. Machines are typically dual-processor x86 processors running Linux, with 2-4 GB of memory per machine.
2. Commodity networking hardware is used (100 megabits/second or 1 gigabit/second at the machine level).
3. A cluster consists of hundreds or thousands of machines, and therefore machine failures are common.
4. Storage is provided by inexpensive IDE disks attached directly to individual machines.
5. Users submit jobs to a scheduling system. Each job consists of a set of tasks, and is mapped by the scheduler to a set of available machines within a cluster.

Execution

In this section, the authors talk about the map and reduce invocation. The map invocation is performed by distributing across multiple machines by automatically partitioning the input data into a set of M splits. This is then processed in parallel by different machines. The reduce invocations are then distributed by partitioning the intermediate keys space into R pieces using the partitioning function. In this case, the number of partitions (R) and the partitioning functions are specified by the user.

When the MapReduce function is called by the user program, a series of sequential activities are performed using workers and master data structures, then the output of the MapReduce execution is available in the partition output files. The MapReduce function also finds means to tolerate worker and master failure since the machines are processing large data set.

After the execution, it is then refined using partition functions, Ordering guarantees, and combiner functions.

There are also side effects of using MapReduce function and they are:

1. It is not always very easy to implement each and everything as a MapReduce program and also when your processing requires lots of data to be shuffled over the network.

CONCLUSION

The MapReduce model has been successful in running multiple data set at Google. It is easier to use the model, even non-experienced programmers with parallel and distributed systems can use the model because it encapsulates the details such as parallelization, fault-tolerance, locality optimization, and load balancing. As a result, Google use it for the generation of data for production web search service, for sorting, data mining, machine learning, etc. The model is very efficient and can reduce the impact of slow machines and handle failure and data loss through redundant execution.

REFERENCE

Dean, J. and Sanjay, G. (2004). MapReduce: Simplified Data Processing on Large Clusters. Retrieved from https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf