MapReduce is a model that map and reduce. Firstly, create a map function in order to process the split collection of data. Then create a reduce function in order to merge all the values. Its big advantage is reducing a lot of time, using the MapReduce model, combined with the user-implemented Map function and Reduce function, we can very easily achieve massively parallel computation. The model changes the task from a stand-alone process to a task in a distributed system environment. The paper shows how to use this model to solve many real world tasks such as Distributed Grep, Count of URL Access Frequency, Reverse Web-Link Graph and so on.

The problem should be split into map and reduce operations. The process of Map includes three parts: initialization, map operation and cleaning. The data is sliced into irrelevant chunks through the map function, and then the results are collated and output through the reduce function. The rules for map function are specified by a function such as the mapping of {2,4} is {4,8} , and the rules for reduce function are also specified by a function such as the result of summing {2,3} is 5. The calculation process of the MapReduce model is iterating over the input data and parsing it into key/value pairs; then the input key/value pair maps the map to form some other key/value pairs; merging and iterating on the data, and the final key/value output will be generated.

Some classic questions use this model. For example, the Count of URL Access Frequency question uses it. The map function processes the record of web page request log, and outputs (URL, 1). The reduce function adds together all the values of the same URL, and output the pair (URL, total count).

The paper also introduce an important mechanism about fault tolerance. The problem maybe have very large amounts of data, and it may uses thousands of machines. So the tolerance machine failures are important. It is able to complete their tasks most quickly in a variety of unexpected situations. It mainly introduces the worker failure, master failure and semantics in the presence of failures. The detection of worker failure uses runtastic Heart Rate PRO. The worker node tells the master every once in a while that it is still alive, and if the master does not receive a message from a worker in a certain amount of time, it concludes that the worker has failed, at which point the worker's tasks will be reset and then reassigned to other workers. The solution of master failure is if the master task dies, a new copy can be started from the last checkpointed state. If a machine takes a long time to complete the last few Map or Reduce tasks, causing the MapReduce operation to over the execution time, the master schedules a backup task process to perform the remaining in-progress tasks.

The paper said that the most successful application of MapReduce was a complete rewrite of the production indexing system that produces the data structures used for the Google web search service. The input data of the indexing system is a large number of documents crawled by the web crawler, and these document data are stored in the GFS file system.

As an efficient distributed computing model, the MapReduce model will play a more important role in the face of increasing computational tasks for large-scale data. The paper highlights the fault tolerance and parallel computing, which was originally designed to meet the require of a lot of computation. This model is easy to use because MapReduce provides an abstract mechanism to isolate the programmer from the details of the system layer. It hides the details of parallelization, fault-tolerance, locality optimization, and load balancing. The programmer only needs to describe what needs to be computed, and the specific how to compute is handled by the system's execution framework, so that the programmer can be freed from the system layer details and work on the algorithm design of the computational problem of its application.

I think this paper provides a meaningful solution to many problem which can be used in our learning life. For example, we can break down a complex problem into many little parts, and solve them one by one.