Review of Paper 9 Ramkumar Rajabaskaran (85241493)

This paper introduces the concept of MAP-REDUCE that is applied in areas where parallelization of a process is possible. This is particularly useful while handling Large data (More than 1 TB) and performing operations on it. By using a concept such as MAP-REDUCE, we can parallelize the process and hence achieve the results in a faster time.

What MAP-REDUCE essentially does is to take a process P and break it down into M chunks (16-64 MB). The Master machine then assigns M machines to work as Mappers and R machines to work as Reducers. Each of these M chunks is then mapped onto a machine that is present in clusters. These machines then process the data via its (Key, Value) pair and writes them to the memory (or Storage device). Once this is done, the intermediate Key/Value pairs produced is then buffered onto the memory or periodically written to a storage device partitioned into R regions. Then a reduce worker is notified by the master about the stored data and it reads this data and sorts it according to the keys and merges values with the same key. The reduce worker finally iterates over the sorted intermediate data and passes the value to the user's reduce function to be appended to the final output. This is the process of a MAP-REDUCE.

One of the biggest considerations that a MAP-REDUCE faces is how to handle any faults or errors that may happen in any of the machine. Since if the machines are like a serial circuit, even if one machine is faulty it may affect the entire system. Hence MAP-REDUCE handles this like a parallel circuit by constantly pinging the worker periodically and if no response is received, it marks it as failed. The work that was done by this machine is marked as idle and given to another worker. Similarly, work that is currently in the machine is also set to idle. Completed reduced tasks need not be reset as they are in the global memory.

The MAP-REDUCE must also consider balancing of loads between machines such that it equally distributes the workload among all the machines. The number of M and R pieces must be larger than the number of workers. There must also be some backup workers just in case some worker has a faulty memory that slows the disk speed from 30MB/s to 1MB/s. That will affect the overall performance and hence the master must schedule these on the backup workers and complete the job faster.

In a test of two programs, a sort and a grep performed on a cluster of 1800 machines, the grep took about 150s with a max of 1764 workers assigned with a peak data rate of 30GB/s. The sort meanwhile has a peak of about 13GB/s.

The popularity of MAP-REDUCE is increasing everyday as it is practically useful across a wide domain of applications such as large-scale machine learning, processing of satellite data, large-scale computations and clustering problems for Google news. Furthermore, it is also useful for large-scale indexing such as for the Google search engine where, documents retrieved by the search engine as large as 20TB of data is distributed and the information is retrieved. The MAP-REDUCE allows us to write simpler and shorter code and get our results at a faster pace.

MY VIEWS: - I Personally feel that MAP-REDUCE is a great concept to be used when processing large corpus of data and especially in fields like machine learning and NLP. In the current project of Web Search engine, I feel that implementing a MAP-REDUCE concept to my program will be beneficial as currently I am taking more than 4-5 hours while tokenizing and building an inverted index. I feel that applying a MAP-REDUCE technique to map the various sub-folders to a Mapper and Reducing them to a A-Z inverted index will be able to reduce the time that I have taken by more than half so that I shall be able to easily correct any mistakes or add more useful information to my index while not having to wait for a long time to process the data. Thus MAP-REDUCE is a very important concept that will be helpful in parallelizing processes for large-scale data retrieval.