MAPREDUCE: SIMPLIFIED DATA PROCESSING IN LARGE CLUSTERS

NAME: AMAN SHANKAR UNI: AS5171

1. Motivation

To create a distributed system to work on large datasets by using unreliable hardware and provide good fault tolerance.

2. Goal

To create a system that can provide parallelism between are data-sets and also takes care of the error and failures occurring in the system.

3. Approach

The component is divided into Map and Reduce. Map partitions the data into M parts and each part are handled by different machines. The Reduce involves by distributing the keys to R partitions. Noting that the partition function is user defined. There is a Map function that looks after the data allocation. The worker nodes do the map and reduction work. The fault tolerance is done with the help of the Master node. Where Master node continuously checks the workers, if it finds any of the worker has failed the master will reallocate the task to another worker. Redundant data are maintained to keep the throughput high.

4. Results

The performance of Map Reduce was checked on two computations. In Grep computation, the performance improves when the number of machines assigned are increased its highest is 30Gb/s. In the sorting computations In the Sorting computation the reading operation takes 13 Gb/s. The reading is slower because it spends considerable amount of time to write to the local disks. The results positively support MapReduce Algorithm.

5. Conclusion

MapReduce has been a huge success in Google and has been instrumental in handling big datasets. The major reasons cited in the paper is that it’s easy to implement without users having experience in parallel computing. Secondly a lot of computational problems can be solved using MapReduce Algorithm.

6. Comments

MapReduce is lightweight and the it works well with most of the computations. MapReduce will not be the appropriate API when we are working with time critical tasks.