Greenlee 560 Summary 4

Elliot Greenlee

February 23, 2017

1 MapReduce: Simplified Data Processing on Large Clusters

2 Problems

What problems does this paper try to solve? Why are such problems important?

This paper provides a programming interface solution for generating and processing large data sets. There are many problems associated with work on large data sets. Without parallelization or distribution, the run time becomes untenable. With, development time increases as synchronization and fault tolerance must be included. Solving this problem is important, as there are many real world tasks which operate on large data sets.

3 Assumptions

What are assumptions made by this paper? Are they verifiable? Are there logical holes in such assumptions?

The paper makes the assumption that mapreduce is a model to which many real world tasks apply. The paper both gives many examples of this claim and reports on the results growth in use over 9 months at Google. While these do not prove its usefulness, they make a strong case for applicability.

The paper also assumes that mapreduce is easy to pick up and use even for those unfamiliar with distributed computing. While no proof is given in the paper, the description of what mapreduce handles that previously was left to the programmer is convincing.

4 Solutions

What are the major solutions of this paper? Do you think the solutions in this paper will work for the problem? Do you think the paper evaluates the solutions in a convincing manner? List two limitations of the solutions proposed in this paper, and outline your method to fix them.

This paper provides the map and reduce interface, as well as an implementation used at Google. The solution for Google most certainly worked for their problem, as shown by the evidence of growth and speedup in the paper. In general, mapreduce has been widely accepted and used by many others for big data and machine learning tasks. This paper has evaluated the model in a specific implementation used across hundreds of groups and tasks.

4.1 Limitations

Google prioritized bandwidth in their implementation, meaning that map results on machines that fail before the data is accessed are lost. For other situations, the loss of data could cost more in terms of time than taking up bandwidth to transfer the data as it is completed.

A programmer must convert their problem to this model currently. Using programming tags or some other mechanism to suggest how existing code can be modeled as a mapreduce would be very helpful.

Discussion Fault tolerance, easy to get started. Mostly right.

At the time, it was set up for googles file system. Prioritized bandwidth, which caused data loss on failure. By now though, there are versions for everyone. openMP type pragmas, or even having an IDE or something suggest how to model as mapreduce

Spark, storm -¿ streaming. Microbatching.

Google grew larger, changed from DFS which their mapreduce was based on.