Chad Hogg

Review of "MapReduce: Simplified Data Processing on Large Clusters", by Dean and Ghemawat

The paper describes a library for collecting aggregate information from large quantites of data. Specifically, the library (with the underlying Google File System) handles issues of fault-tolerance, parallelism, load-balancing, and inter-process communication. Specifically, the user of MapReduce writes two functions: a mapping function that produces temporary output in the form of zero or more <key,value> pairs and a reduction function that receives intermediate results and computes aggregate functions over those results. The paper also describes several domains in which the MapReduce library could be useful and presents statistics regarding its use at Google.

There is a clear explanation of how the system works, as well as a programmer's perspective on how it is useful. Additionally, the use of concrete examples of problems that are well-suited to the MapReduce model and that have in fact been implemented with MapReduce and are currently in production use by a major corporation made it more clear in this paper than most how the results are applicable to real-world problems. However, the usefullness of the system seems rather dependent on a cluster system like Google uses, and it is not clear how easy porting the library to work with other types of multi-processor systems would be.

In contrast to the well-written and expansive discussion of the system architecture and problems it solves, the evaluation section of the paper is rather weak. Wall-clock times are given for several different algorithms that have been written using MapReduce, but there is no reference value provided to compare these to. The only metric with which they compare MapReduce directly to other methods is a mention that MapReduce programs are significantly fewer lines of code than equivalent ones written without it. While this may be a compelling argument to adopt the MapReduce system, it alone is not enough to evaluate the technical contributions of the system.

The list of references includes the obvious descriptions of lower-level features of Google's cluster architecture and related works. However, a reference should probably be added for the "map and reduce primitives present in Lisp and many other functional languages" (page 1, column 2) for those readers who are unfamiliar with the languages and wish to understand the authors' motivation before reading the remainder of the paper.

This paper is well-written and, other than the evaluation section, the content is both concise and sufficient to convey the important points the authors should be making. The research itself does not provide any scientific breakthroughs, but describes a system to simplify and conglomerate common programming tasks. Thus, this paper may not be appropriate for a research journal. It would, however, make quite an interesting article in a book on grid computing systems or software library design.

10/10