Ikhlas Attarwala
MSAI 339 – Data Science Seminar
10-10-18 – Background Reading

Summarizing "MapReduce: Simplified Data Processing on Large Clusters" by Dean, Ghemawat

Jeffrey Dean and Sanjay Ghemawat from Google, Inc. have put forth an interesting paper detailing the advantages and usefulness of MapReduce for large clusters of data. They've claimed that Google has branched out to find methods to compute large stores of data, but met issues with parallelizing computation, distribution, and handling failures. Based off of the *map* and *reduce* primitives from Lisp, they designed a model that expresses both as functions to receive inputs and produce outputs while parallelizing large computations, strategically handle fault tolerance and speed up a user's program development, all while making it very easy for a user with no prior experience to utilize MapReduce with ease. In short, the library splits input files into pieces where "worker" machines are assigned tasks to parse, store for processing, provide locations for, sort and then append an output file for the user to readily access.

The paper has many strengths, and some of these include structure organization, soundness of implementation, and discussing performance comparisons. On page 140, the authors perfectly lay out the step-by-step processes of how their program runs, making it very easy for the reader to conceptualize their work. To provide an even better representation, on page 139 a schema to outline the MapReduce execution model is drawn out. In addition to the organization of execution, the authors do well to explain fault tolerance, a major issue when dealing with large amounts of data over thousands of machines. They explain with confidence how MapReduce has been designed to cover for "worker" machine failure, and "master" machine failure, and even relay an event in which 80 machines had become "unreachable" for several minutes, but the master machine re-executed the work in a way that led to no loss of data or failure. This example is fantastic for getting a read to believe that MapReduce is a steady and reliable choice. Dean and Ghemawat also cover performance comparisons on pages 144-146, showing tests of data transfer rates in searching, sorting, and backup tasks, to prove that they're not exaggerating on MapReduce's effectiveness.

Unfortunately, there were a few gaps in the paper where I felt they could have done more for promoting MapReduce. Firstly, this paper seems geared toward an audience who has familiarity with processing large data sets, however on the first page to last page, they say that "the model is easy to use, even for programmers without experience with parallel and distributed systems…" If novice programmers were an audience they're looking to infatuate, they could have transitioned into their work from less specialized knowledge to allow them a better understanding. Additionally, while stressing Fault Tolerance and risk management, the paper glances over failure of a "master" machine, suggesting that while the likelihood is small, they don't have any alternatives or solutions for this type of failure, other than aborting the computation altogether. Finally, the authors spend very little time discussing the limitations of MapReduce. I understand this is a product they're featuring, but it's important to highlight both strengths and weaknesses.

Written by Google executives, and even showcasing a table of MapReduce's amount of work completed at Google, it'll draw the attention of every major data scientist. Calling to attention the success and innovative fault tolerance of this program for all types of data engineers (new or experienced) will no doubt take MapReduce in a direction where it's user base could explode overnight. As explained earlier, a few important research questions do remain, including limitations and user-prone error. I think the research should look further into this in the future.