I remember almost eight years ago, in a special lecture for my first ever computer science, two guys from Google were talking about their work. My assignment for that week was to read a paper and write some simple code. At the time, in my naivety, I did not think too much of the work, and thought it was not too different from the map and reduce functions that I had written in scheme for assignments in the previous few weeks. Perhaps that is the beauty of Jeffrey Dean and Sanjay Ghemawat’s MapReduce model, as many of the practical considerations of distributed problem solving is abstracted away from the developer so much so that to the untrained eye, it appeared mundane. At the heart of the model are two functions: map and reduce. Map takes a specified input and creates a set of key-value pairs, while reduce takes all the key-value pairs output by map and aggregate them into buckets by sorting keys and process the values. Based on the functional programming paradigm, the map functions do not need to interact with each other, and thus performance gain can be achieved through horizontal scaling on commodity hardware—to the point where it makes sense with respect the the divisibility of data, job overhead, reducer complexity, and probably more considerations that I am not aware of. The coordination of work is done on a master node, where it divides the cluster into a set of mapper and a set of reducers, assigns work to each and tracks job progress and failures. Invisible to the user, the authors made optimizations made to ensure that minimal network bandwidth is used (by maximizing locality of computations), to minimize the impact of hardware failures (by periodic polling of worker nodes to identify failures, and creation of backup tasks to reduce stragglers), and to maximize efficiency of reduce function on intermediate key-value pairs (by processing inputs in order key and combiner function to aggregate identical keys). Thus, the strengths of the MapReduce model has been its ease of use, applicability to a wide variety of problems, and horizontal scalability.