Realtime and Big Data Analytics - Fall 2018
Homework 1

Daniel Rivera Ruiz
Department of Computer Science
New York University
daniel.rivera@nyu.edu

1 Paper Summary

The paper *MapReduce: Simplified Data Processing on Large Clusters* by Jeffrey Dean and Sanjay Ghemawat talks about the MapReduce programming model and how it was implemented at Google to simplify many tasks that can be expressed in terms of it.
The key idea behind the MapReduce model is to express the desired computation as two functions: Map and Reduce. The Map function takes the input *key/value* pairs and emits a set of intermediate pairs. The Reduce function will process an intermediate key and all the values associated to it and merge them together into a (usually) smaller set of values.
To implement the model in a cluster consisting of hundreds or thousands of machines, there is a Master machine that assigns tasks to the rest of the machines called workers. A worker that is assigned a Map task performs the operation to its assigned partition of the input data and the resulting intermediate pairs are buffered in memory. When these pairs are written to the local disk the Master notifies a reduce worker, who will sort the data by the intermediate keys and pass it to the reduce function, generating the desired values that are appended to a final output file. Under this model, when a worker fails the master can simply reschedule the tasks that it had been assigned. A failure in the master will abort the computation, but this is a very unlikely scenario since the master machine is usually very robust.
To optimize the performance of the system, the Master tries to schedule map tasks to workers that have a local copy of the required data, minimizing data transfer through the network. Additionally, the map and reduce phases are subdivided into M and R tasks, with M and R usually much bigger than the number of available workers: this improves dynamic load balancing and makes recovery after a worker failure faster. Finally, backup tasks can be scheduled by the Master to deal with "straggler" workers: when the MapReduce operation is near completion, the Master will schedule backup executions of the remaining tasks to account for workers that may be taking to long to finish.
Additional to the MapReduce basic functionality, there are some additional features that can be useful for certain applications: providing a specific partitioning function, generating sorted output files, providing a combiner function (perform partial merging before it is sent over the network), reading and writing different data types and formats, generating auxiliary output files, skipping bad records, implementing a sequential version of the MapReduce operation for debugging and profiling, monitoring the status information generated by the Master and using the counter feature of the MapReduce library to count occurrences of certain events throughout the execution.