MapReduce and TensorFlow

MapReduce is a programming model for processing and generating large datasets. It is used by Google and is executed on Google's clusters daily (at the time of the paper). MapReduce is useful because it is automatically parallelized and executed on a large cluster (depending on usage) of machines. This, in addition to MapReduce handling scheduling, machine failure and communication for you allows programmers with zero parallel/distributed system experience to utilize a large distributed system. Although it can be run on a large distributed system, MapReduce can be used in different hardware scenarios depending on the usage requirements. It is able to run on small shared memory machines, large NUMA multiprocessors as well as the aforementioned large collection of networked machines.

MapReduce's name contains its two main user-defined functionalities, *map* and *reduce.* The mapping function is used to take a key/value pair and generate a set of intermediate key/value pairs. It then gathers all the intermediate values associated with a specific key and passes them to the reduce function. The reduce function takes all of these intermediate values, does some processing and then returns either 0 or 1 output value. Both of these methods are useful for generating/processing large data sets and are run on the different machines in the cluster. In addition to map/reduce, there is also the partition and combiner functions. Although typical use cases are fulfilled just by map/reduce, MapReduce provides additional API to detail and configure the size and location of partitions (through the partition function) and the level of repetition of data in intermediate steps (by the combiner function.

Since there are potentially thousands of machines in a MapReduce system, there is a need for fault tolerance. The master will ping every worker periodically and if they don't ping back, then they're marked as failed and the work they were doing is reset and available for another worker. If the master fails, then the entire MapReduce computation is aborted as there is only one master and it is deemed unlikely for it to fail.

The paper gives an overview of different problems involving large data sets that MapReduce can be used to solve. Of these, there is performance analysis and we can see that MapReduce is very performant in the cases outlined in the table like Grep and Sort.

Tensorflow is less so directly involved in distributed operating systems in the traditional sense, instead focusing more on how to share data resources between multiple processors, mostly on a local machine, in order to better utilize available processing features to accelerate computations for the matrices used in tensorflow neural networks. These developments are more indicative of how modern multi-processor design architecture works, with each processor in a localized system being treated as an individual system, with resources that can be shared (memory acting like network storage), and having to make use of high speed transfer protocols to allow resources that are more localized to be transported between systems. This makes out a modern multiprocessor system to be similar to the idea of distributed operating systems from the past, albeit without the limitations of encryption over a slow network.