MapReduce – Review

Yixing Chen, yc3094

The paper *MapReduce: Simplified Data Processing on Large Clusters* introduced MapReduce – a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of a map and a reduce function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks.

MapReduce is designed to easily and efficiently compute various kinds of derived data, such as inverted indices, various representations of the graph structure of Web documents and so on. The use of a functional model with user-specified map and reduce operations allows us to parallelize large computations easily and to use reexecution as the primary mechanism for fault tolerance.

The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. There are two functions: Map and Reduce. Map function takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the reduce function. Reduce function accepts an intermediate key I and a set of values for that key, merges them to form a possibly smaller set of values. The intermediate values are supplied to the user's reduce function via an iterator.

In the Implementation part, the writer tells us what happened when MapReduce function is called. The process is: The MapReduce library in the user program splits the input files into M pieces → The master picks idle workers and assigns each one a map task or a reduce task → A worker reads the contents of the corresponding input split, parses key/value pairs out of the input data and passes each pair to the user-defined map function → The buffered pairs are written to local disk, and locations are passed back to the master → Reduce worker uses remote procedure calls to read the buffered data from the local disks of the map workers and sorts it → The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's reduce function. The output of the reduce function is appended to a final output file → When all map tasks and reduce tasks have been completed, the master wakes up the user program and returns back to the user code.

MapReduce library must tolerate machine failures. It has mechanism to deal with worker failure and master failure. To achieve higher locality and conserve network bandwidth usage, the input data resides on local disks of the machines that make up the cluster.

There are also many refinements and useful extension, for example, user-specified partitioning functions, ordering guarantees, user-specified combiner functions, custom input and output types and a mode for execution on a single machine.

The performance has been measured on two computations running on as large cluster of machines. One computation searches through approximately one terabyte of data looking for a particular patter. The other computation sorts approximately one terabyte of data. The result shows that backup files are effective and machine failures are under control.

The MapReduce is applied broadly and has been used across a wide range of domains within Google including large-scale machine learning problems, clustering problems for the Google News and Froogle products and so on. The primary source-code management system, increased from 0 in early 2003 to almost 900 in September 2004, to about 4000 in March 2006.

At last, the writer attributs the success of the MapReduce programming model to some reasons: the model is easy to use, a large variety of problems are easily expressible and they have developed an implementation of MapReduce that scales to large clusters of machines comprising thousands of machines.

In my opinion, it’s a great paper to introduce the the MapReduce to us. It illustrates structures, detailed process, implementation, performance, reasons of its success and so on. We can easily know about the features and working mechanism of MapReduce, which may provide valuable experience for our future studying and working.