BUILDING AND INSTALLING A HADOOP MAP/REDUCE CLUSTER FROM COMMODITY COMPONENTS

JEFFREY JOHN GEEVARGHESE, KARTHIK.V, KARTHIK R, SHERIN THOMAS
NATIONAL INSTITUTE OF TECHNOLOGY [NIT], CALICUT

We describe a straightforward way to build, install, and operate a compute cluster from commodity hardware. A compute cluster is a utility that allows you to perform larger-scale computations than are possible with individual PCs. We use commodity components to keep the price down and to ensure easy availability of initial setup and replacement parts, and we use Apache Hadoop as middleware for distributed data storage and parallel computing.

MAP REDUCE / HADOOP CONCEPTS

1. MapReduce

MapReduce is a patented software framework introduced by Google to support distributed computing on large sets of clusters, i.e. for processing huge datasets on certain kinds of distributable problems, using a large number of computers, collectively referred to as a cluster. In the cluster, we recognize two types of systems – the master and the worker. The master assigns jobs (tasks) to the worker nodes and co-ordinates the overall computation.
The data used by the cluster computing system may be unstructured (as when stored in files) or structured (as in databases). There are two phases associated with a MapReduce problem:

Map Step: The master node takes the input, chops it up into smaller sub-problems, and distributes those to worker nodes. A worker node may do this again in turn in multi-level setups.

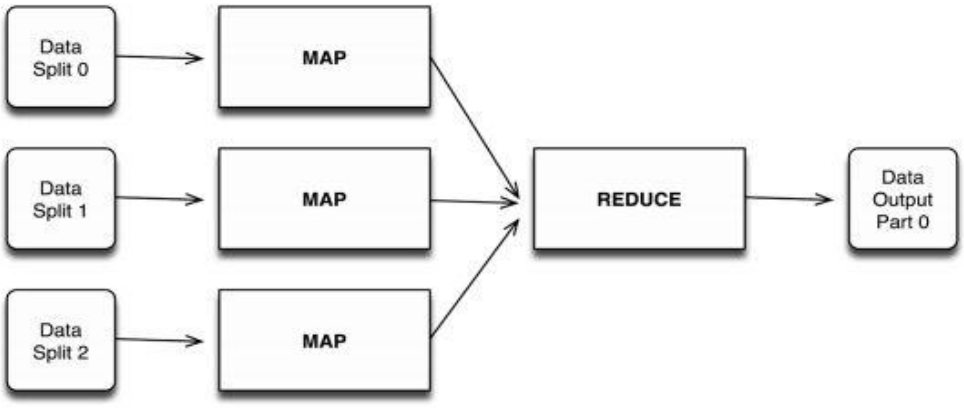

FIGURE 1: MAPREDUCE PROCESSING MODEL

Reduce Step: The worker node processes that smaller problem, and passes the answer back to its master node.

MapReduce is inherently distributed allowing independent sub problems to be performed in parallel. Hadoop automatically addresses issues with synchronization and distribution under the hood.

2. Hadoop

Hadoop is a Java-based software framework that supports data intensive distributed applications. It allows applications to work with thousands of computing nodes and huge amounts of data.

Hadoop features a Rack-aware distributed file-system called the Hadoop Distributed File System (HDFS). HDFS allows data to be stored on multiple machines and takes care of synchronization and obsoleteness issues.

HIGH LEVEL IMPLEMENTATION DETAILS

To setup a multi-node Hadoop cluster, each node must be capable of single node operation (i.e., in isolation, without explicit master or worker nodes). Ubuntu Linux, running Hadoop 2.0.20 was used as the test platform in our systems. Each node in the cluster had to be set up for single node operation based on **[1]**).

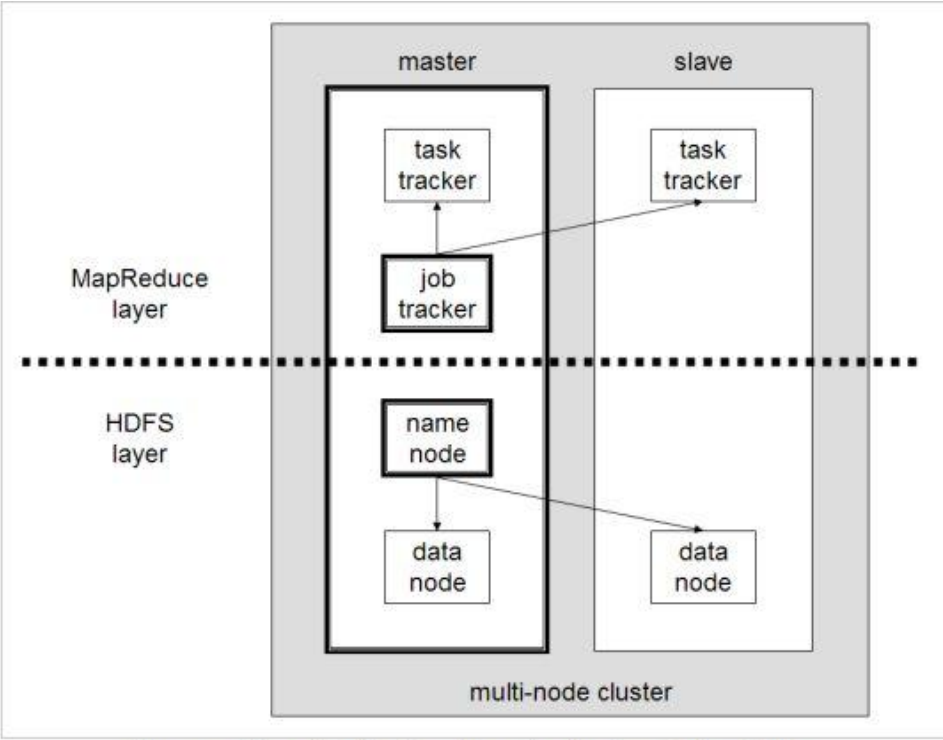

Figure How the final multi-node cluster will look like.

Once every system is paired and running Hadoop in single node mode, the nodes must be demarcated as master and workers. To do so, the */etc/hosts* on both the master and worker machines has to be updated to include:

```
<IP of master> master
<IP of slave1> slave1
<IP of slave2> slave2
<IP of slave3> slave3 . . .
```

Next, the *conf/masters* file in the Hadoop package is updated to include the alias of the designated master node, which in this case is `master`. Also, the *conf/slaves* in the Hadoop package contains the aliases of all the slaves – `slave1`, `slave2`, etc.

Next, the Hadoop configuration files are updated to reflect the number of slave nodes and the designated locations of the name and data nodes. Finally, format the distributed file system and initialize the cluster nodes on the master and workers. Start the MapReduce job on the master and it automatically maps it to each of the worker nodes, finally reducing the distributed result.

CONCLUSION

The steps above creates job handler nodes in the server and worker systems, enabling distributed applications to be run across the cluster, by MapReducing the problem. On this setup, standard MapReduce jobs can be run.

REFERENCES

[1] *Setting up a Single Node Cluster*, Michael G. Noll

[2] *Building and installing a Hadoop/MapReduce cluster from commodity components: A case study*, Jochen L. Leidner; Gary Berosik