Auto-Scale Multi-Node Hadoop Cluster – An Experimental Setup

[1]Praseetha V. M, [2]Pratik Sen, [3]S. Vadivel
[1]Asst. Professor, [2]UG Student, [3]Professor
[1]Computer Science and Engineering,
[1]St. Joseph's College of Engineering & Technology, Palai, Kerala, India

***Abstract*— With the plateauing of computational hardware, the need to get the maximum performance both reliably and at low cost has seen to the revival of cluster computing. Cluster computing makes use of a large number of low cost ordinary everyday computers. Apache Hadoop is a scalable, flexible, cost effective and fault tolerant open source software system framework which enables distributed processing across clusters. As Hadoop is statically set up all configurations are defined before deploying the actual nodes. In such a case mirror images of the hardware partitions are used and then applied to the target machines. However such a case causes a very brittle set up which can cause a catastrophic failure with the failure of the master nodes. To reinstate the nodes the entire cluster would generally need to be brought down. To combat this weakness we successfully set up a novel multi node approach with two nodes and is presented in this paper.**

***Index Terms*—Cluster Computing, Hadoop, Distributed processing, Cluster**

I. INTRODUCTION

Computer clustering requires a central management approach over a set of nodes as supervised servers. It is similar to other node-based methods such as peer to peer or grid computing, and differs in the fact due to its massively distributed properties. A cluster may consist of simple binary node system, involving merely two ordinary computers, to a singular but very fast supercomputer. A first approach to building a cluster is the Beowulf architecture for a cluster [1]. The cluster may be built with a few ordinary computers to produce what was to be the first computing cluster. An early study that showed the viability of the clustering concept was Stone Supercomputer. The developers built a network of 133computers, all running Linux. Communications between the nodes was actualized via the Parallel Virtual Machine toolkit and the Message Passing Interface library. The study was able to achieve high performance at a relatively low cost. Even in cases of very simple configurations in terms of both hardware and network, the cluster architecture was used to achieve very high levels of performance.

Attributes Of Clusters

There exist a variety of configurations for computer cluster depending on differing use cases such business oriented tasks like web-service support, or computation heavy tasks like scientific simulations. It is prudent to that the attributes described below are not thought to be exclusively applicable to clusters, but may be generalized for any system using a high-availability approach. There are two primary types of clusters, the first of which are Load Balancing clusters. In this type of configuration, the overall work load is shared by cluster-nodes to impart better overall performance. An example where this scheme is applicable is in the case of a cluster tasked as a webserver. In this case each query is assigned to its own node drastically optimizing response time. However, the actual scheme used to balance workload would be different based on the task.
High-availability clusters operate by having redundant nodes, which are used as failsafe's in case of critical failures. Redundancy of cluster components is used as method to eliminate bottle necking and single point failures.
Computer clusters are designed specifically for purposes that require heavy duty computing, rather than primarily handling IO-bound tasks. A common use-case of this kind of cluster is in the simulation and analysis of complex physical dynamics such as car crashes.

Hadoop

Apache Hadoop [2] is an open-source software system framework for storage and large-scale processing of data-sets on clusters of ordinary hardware. The Apache Hadoop framework [3] consists of the subsequent modules: Hadoop Common – contains libraries and utilities needed by alternative Hadoop modules, Hadoop Distributed filing system (HDFS) – a distributed file-system that stores information on artifact machines, providing terribly high aggregate bandwidth across the cluster, Hadoop YARN – a resource-management platform responsible for managing compute resources in clusters and exploiting them for programing of users' applications and Hadoop MapReduce – a programming model for large scale processing.

Hadoop: Software Architecture

Hadoop is a software package which runs over the cloud providing OS like a file system. The Hadoop Distributed file system (HDFS) [3] is an engine for supporting the Map Reduce programming paradigm. The Hadoop Mundane package contains the

compulsory Java ARchive binaries to run all of its functionalities. For efficacious programming, all Hadoop-compatible file systems should provide the location of the child node. The applications utilize the information to enact transformations over the node wherever the data is available. HDFS utilizes this method when replicating cognizance to keep exhaustively different data facsimiles of the info on different systems. The goal is to scale back the impact of a rack breakdown or switch failure, so that the info should still be decipherable.

A minuscule Hadoop cluster includes one master and several employee nodes. A slave or employee node acts as both a DataNode and TaskTracker, though it is attainable to have a data and compute services unique to disparate nodes. These square measure commonly used entirely in nonstandard applications [4]. Hadoop needs Java Runtime environment (JRE) 1.6 or higher. The quality start-up and closure scripts need Secure Shell (ssh) to be engendered between nodes among the cluster [5]. In a more immensely colossal cluster, the HDFS is maintained via frenzied NameNode server to host the file system index, and a secondary NameNode which will engender snapshots of the namenode's recollection structures, therefore averting file-system corruption and reducing loss of data. Similarly, a standalone JobTracker server can manage job designing. In clusters where the Hadoop MapReduce [6] engine is deployed against associate alternate file system, the NameNode, secondary NameNode and DataNode design of HDFS is superseded by the file-system-concrete equipollent.

II. Hadoop Implementation Of Modern Clusters

Typical Hadoop Cluster

The typical Hadoop cluster [7] has, primarily two roles for its member nodes:

- **Masters: Which has the HDFS Name Node service** responsible for the fragmentation and tracking of data over the cluster, the JobTracker which tracks and shecdules the MapReduces tasks to various nodes based on the job. The JobTracker is also responsible for balancing the workload of the cluster.

- **Slaves: Which run the HDFS DataNode service which responsible for storing and serving its allocated data fragment to the cluster.** For the compute Nodes the MapReduce TaskTrackers service is used, which is responsible for executing and scheduling the Map and Reduce tasks allotted to it by the Jobtracker. Often times the DataNode and TaskTracker nodes are loosely bound to ensure that the working data set of the TaskTracker is speedily and reliably served with the lowest latency.

The final class of machines involved in a Hadoop cluster is the client machines, although they have Hadoop on system, they cannot be classified as the other two types of nodes. The client machines are used as the role of upkeep machines, which aid in the efficient function of the two primary kinds of nodes. They are involved in the transfer of data to and from the cluster and defining jobs on how to preprocess that data.Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts. True Type 1 or Open Type fonts are required. Please embed all fonts, in particular symbol fonts, as well, for math, etc.

Typical Hadoop Node Machine

Systems made designed for production, i.e production clusters, do not use a level of virtualization. As doing so will cause an unnecessary performance costs in all areas of the system from memory to processing and even latency in IO bound operations. As such overheads are unacceptable in a production environment, native Linux machines are used. As Hadoop is statically set up all configurations are defined before deploying the actual nodes. In such a case mirror images of the hardware partitions are used and then applied to the target machines. However such a case causes a very brittle set up which can cause a catastrophic failure with the failure of the master nodes. To reinstate the nodes the entire cluster would generally need to be brought down.

In a typical Hadoop cluster, rack servers are used, populated with levels of systems with a switch at the top of the rack with a 2Giga Bit Ethernet throughput. These rack unlike the normally used blade servers are ordinary machines and not high performance server systems. The switches of the rack are then connected to another networking layer, consisting again primarily of switches, essentially providing network uplinks to the other node, forming the network backbone of the cluster.

III. Proposed Auto-Scale Multinode Hadoop Cluster

Our small Hadoop test cluster includes one master and multiple slave nodes. The master node consists of a JobTracker, TaskTracker, NameNode and DataNode. A slave or employee node acts as every a DataNode and TaskTracker, although it's attainable to possess data-only slave nodes and compute-only employee nodes. These unit of measurement normally used alone in nonstandard applications[1]. Hadoop needs Java Runtime surroundings (JRE) 1.6 or higher. The quality start-up and closure scripts need Secure Shell (ssh) to be created between nodes at intervals in the cluster[2].

In a larger cluster, the HDFS is managed through a zealous NameNode server to host the classification system index, and a secondary NameNode which will generate snapshots of the namenode's memory structures, so preventing file-system corruption and reducing loss of knowledge. Similarly, a standalone JobTracker server can manage job comingup with. In clusters where the Hadoop MapReduce engine is deployed against associate alternate classification system, the NameNode, secondary NameNode and DataNode design of HDFS is replaced by the file-system-specific equivalent.

Prerequisites for Test Server

It is assumed that the basic installation and setup is already instantiated as a virtual machine image. Now that we have two identical and Hadoop enabled systems up and running up and running. The two images will be then modified making one of them the master and the other as a slave.

Networking

It is to be noted that both systems are to be connected via a network. The easiest method to configure said network is to place both systems within the same hardware network with the same software conditions. A simple example of this is both systems being connected via a single hub or switch.

For simplicities sake, the following networking addresses is assigned, the master machine is configured with an IP address 192.168.0.1 and 192.168.0.2 is assigned to the slave machine.

```
1>/etc/hosts
192.168.0.1    master
192.168.0.2    slave

2>Distribute the SSH public key of hduser@master
hduser@master:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave
This command will prompt you for the login password for user hduser on slave,
then copy the public SSH key for you,
creating the correct directory and fixing the permissions as necessary.
hduser@master:~$ ssh master
The authenticity of host 'master (192.168.0.1)' can't be established.
RSA key fingerprint is 3b:21:b3:c0:21:5c:7c:54:2f:1e:2d:96:79:eb:7f:95.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master' (RSA) to the list of known hosts.
Linux master 2.6.20-16-386 #2 Thu Jun 7 20:16:13 UTC 2007 i686
...
hduser@master:~$
…and from master to slave.
hduser@master:~$ ssh slave
The authenticity of host 'slave (192.168.0.2)' can't be established.
RSA key fingerprint is 74:d7:61:86:db:86:8f:31:90:9c:68:b0:13:88:52:72.
Are you sure you want to continue connecting (yes/no)? yes
Ubuntu 10.04
...
hduser@slave:~$
```

Fig.1. Setting up the Networking

Test Cluster Overview

Within the following section, the configuration process of both the master node as well as the slave node will be expanded upon. However, in this setup the master node will display behaviors of a slave node due to constricted nature of the overall system in terms of systems.

The master node will run all cluster management nodes such as the following: The NameNode which is in charge of managing the HDFS storage layer. The JobTracker which manages nodes and load balancing within the MapReduce compute layer. As in this case, each of the machines will run with the slave layer as well. The DataNode, which manages the HDFS fragment, started on that system. The TaskTracker executing and managing MapReduce jobs given to the node

Configuration

The conf/masters file lists out all the machines where the secondary NameNode service may be run over the Hadoop cluster. Since this is a binary system, there are no secondary name nodes in the system. The machine over which the following startup commands are run bin/start-dfs.sh and bin/start-mapred.sh scripts, will become the primary NameNode and the JobTracker respectively.

Here are more details regarding the conf/masters file:

The secondary NameNode generally operates by merging the fsimage. It is also responsible for periodic edits of the log files. Since memory requirements are similar to the primary NameNode it is not advisable to run both services on the same physical hardware. The startup script “bin/start-dfs.sh“is used to startup the secondary NameNodes on the nodes specified in “conf/masters“ file.

The following is the configurations used on the conf/master:

Figure 2: Auto Configuration of master.conf

Figure 3:Auto Configuration of slave nodes

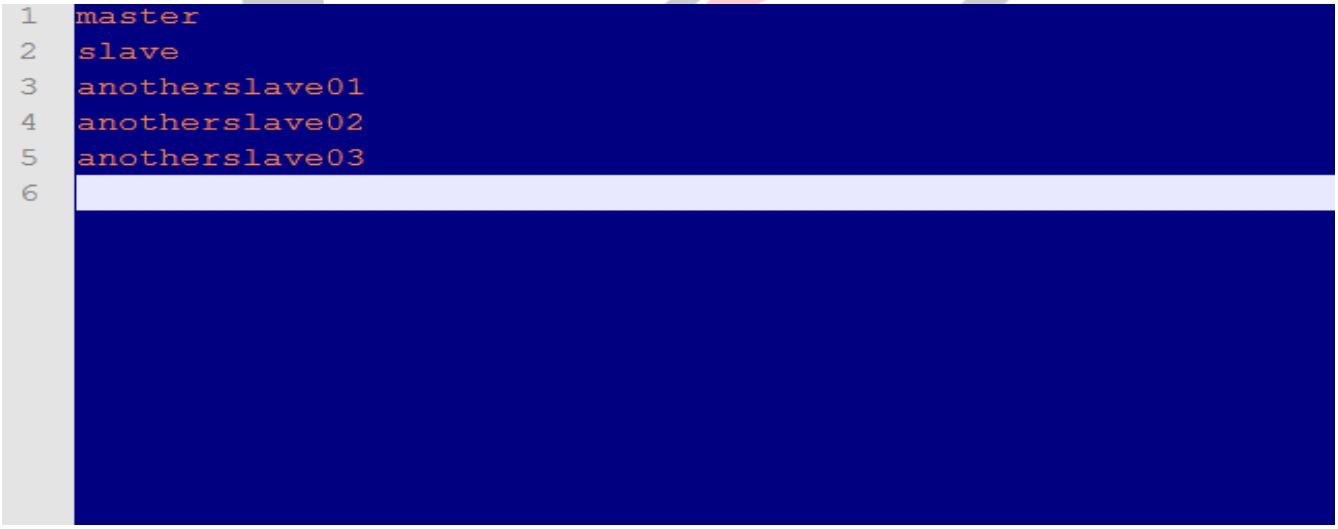

Figure 4: Configuration of slave nodes with additional slave nodes

The following changes to the conf/hdfs-site.xml, conf/mapred-site.xml and conf/core-site.xml, must be done on all nodes.

```
<property>
  <name>fs.default.name</name>
  <value>hdfs://master:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
```

Figure 5: Configuration of conf/hdfs-site.xml in master

```
<property>
  <name>mapred.job.tracker</name>
  <value>master:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
</property>
```

Figure 6: Configuration of conf/mapred-site.xml in slave nodes

```
<property>
  <name>dfs.replication</name>
  <value>2</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>
```

Figure 7: Configuration of conf/core-site.xml in both slaves and masters

In file conf/mapred-site.xml:

- mapred.local.dir - Determines where temporary data is to be stored for MapReduce jobs
- mapred.map.tasks - It is number of map jobs that can be assigned to a given number of TaskTrackers.
- mapred.reduce.tasks - A It is number of reduce jobs that can be assigned to a given number of TaskTrackers.

```
hduser@master:/usr/local/hadoop$ bin/hadoop namenode -format
... INFO dfs.Storage: Storage directory /app/hadoop/tmp/dfs/name
has been successfully formatted.
hduser@master:/usr/local/hadoop$
```

Figure 8: Formatting the HDFS filesystem

Background: The NameNode stores the HDFS fragment table, in our case the master node, in the directory specified by configuration option dfs.name.dir.

Starting the cluster is performed in two steps. First the HDFS daemons are started, namely the NameNode which is run on the master, and DataNodes which are run on the slaves. After this the MapReduce services are started, namely the JobTracker service on the master and the TaskTracker service which is run on all of the slaves.

```
hduser@master:/usr/local/hadoop$ bin/start-dfs.sh
starting namenode, logging to /usr/local/hadoop/bin/../logs/hadoo
p-hduser-namenode-master.out
slave: Ubuntu 13.04
slave: starting datanode, logging to /usr/local/hadoop/bin/../log
s/hadoop-hduser-datanode-slave.out
master: starting datanode, logging to /usr/local/hadoop/bin/../log
s/hadoop-hduser-datanode-master.out
master: starting secondarynamenode, logging to /usr/local/hadoop/b
in/../logs/hadoop-hduser-secondarynamenode-master.out
hduser@master:/usr/local/hadoop$
```

Figure 9: Service starting the HDFS layer

```
... INFO org.apache.hadoop.dfs.Storage: Storage directory /app/hadoop/tmp/dfs/data is not formatted.
... INFO org.apache.hadoop.dfs.Storage: Formatting ...
... INFO org.apache.hadoop.dfs.DataNode: Opened server at 50010
... INFO org.mortbay.util.Credential: Checking Resource aliases
... INFO org.mortbay.http.HttpServer: Version Jetty/5.1.4
... INFO org.mortbay.util.Container: Started org.mortbay.jetty.servlet.WebApplicationHandler@17a8a02
... INFO org.mortbay.util.Container: Started WebApplicationContext[/,/]
... INFO org.mortbay.util.Container: Started HttpContext[/logs,/logs]
... INFO org.mortbay.util.Container: Started HttpContext[/static,/static]
... INFO org.mortbay.http.SocketListener: Started SocketListener on 0.0.0.0:50075
... INFO org.mortbay.util.Container: Started org.mortbay.jetty.Server@56a499
... INFO org.apache.hadoop.dfs.DataNode: Starting DataNode in: FSDataset{dirpath='/app/hadoop/tmp/dfs/data/current'}
... INFO org.apache.hadoop.dfs.DataNode: using BLOCKREPORT_INTERVAL of 3538203msec
```

Figure 10: Example output on slave on job

```
hduser@master:/usr/local/hadoop$ jps
14799 NameNode
15314 Jps
14880 DataNode
14977 SecondaryNameNode
hduser@master:/usr/local/hadoop$
```

The following Java processes were noted on the master

Figure 11: Java Process started on the master by the service

The following Java process were noted on the slave

```
hduser@slave:/usr/local/hadoop$ jps
15183 DataNode
15616 Jps
hduser@slave:/usr/local/hadoop$
MapReduce daemons
```

Figure 12: Configuration of slave nodes

```
hduser@master:/usr/local/hadoop$ bin/start-mapred.sh
starting jobtracker, logging to /usr/local/hadoop/bin
/../logs/hadoop-hduser-jobtracker-master.out
slave: Ubuntu 13.04
slave: starting tasktracker, logging to /usr/local/hado
op/bin/../logs/hadoop-hduser-tasktracker-slave.out
master: starting tasktracker, logging to /usr/local/hado
op/bin/../logs/hadoop-hduser-tasktracker-master.out
hduser@master:/usr/local/hadoop$
```

Figure 13: Service starting the MapReduce layer

The following Java processes were noted on master after starting the MapReduce service

Figure 14: MapReduce Java process started by service in master

```
hduser@master:/usr/local/hadoop$ jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode
hduser@master:/usr/local/hadoop$
```

```
hduser@slave:/usr/local/hadoop$ jps
15183 DataNode
15897 TaskTracker
16284 Jps
hduser@slave:/usr/local/hadoop$
```

Figure 15: MapReduce Java Process started by service in slave

To stop the multi-node cluster .Run the command bin/stop-mapred.sh on the JobTracker machine. This will shut down the MapReduce cluster by stopping the JobTracker daemon running on the machine you ran the previous command on, and TaskTrackers on the machines listed in the conf/slaves file.

```
hduser@master:/usr/local/hadoop$ bin/stop-mapred.sh
hduser@master:/usr/local/hadoop$ jps
14799 NameNode
18386 Jps
14880 DataNode
14977 SecondaryNameNode
hduser@master:/usr/local/hadoop$
```

Figure 16: Stopping the MapReduce layer

Java process running on master as noted are after halting the MapReduce service:

Figure 17: Java process running on master

Java processes on slave after stopping MapReduce daemons

```
hduser@slave:/usr/local/hadoop$ jps
15183 DataNode
18636 Jps
hduser@slave:/usr/local/hadoop$
HDFS daemons
```

Figure 18: Java processes on slave

Stopping the HDFS layer

```
hduser@master:/usr/local/hadoop$ bin/stop-dfs.sh
stopping namenode
slave: Ubuntu 10.04
slave: stopping datanode
master: stopping datanode
master: stopping secondarynamenode
hduser@master:/usr/local/hadoop$
```

Figure 19: Output while halting a HDFS node on master

The following Java process were noted to be running on the master after stopping HDFSs services

```
hduser@master:/usr/local/hadoop$ jps
18670 Jps
hduser@master:/usr/local/hadoop$
```

Figure 20: Remaining services on master

The following Java process were noted to be running on the slave after stopping HDFSs services

```
hduser@slave:/usr/local/hadoop$ jps
18894 Jps
hduser@slave:/usr/local/hadoop$
```

Figure 21: Remaining services on slave

IV. Conclusion And Future Scope

Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the MapReduce computing paradigm. Since Hadoop is statically setup, a failure of the master node causes the entire cluster to be brought down. For Hadoop a multi node cluster of two nodes was successfully setup to overcome the above mentioned weakness thus providing a dynamic set up. There is exciting development in the field of auto scaling Hadoop networks and it is feasible to develop a software stack on which Hadoop clusters can automatically consume and authenticate new nodes.

References

[1] M. Fowler, E. Stipidis and F.H. Ali , "Practical Verification of an Embedded Beowulf Architecture using Standard Cluster Benchmarks," in *The Third International Conference on Software Engineering Advances, ICSEA '08.* , 2008.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. of OSDI'04*, Dec. 2004.

[3] T. A. S. Foundation, "Hadoop Map-Reduce tutorial," [Online]. Available: http://hadoop.apache.org/docs/current/index.html.

[4] H. T. Kung, Chit-Kwan Lin, Dario Vlah and Giovanni Berlanda Scorza, "Speculative pipelining for compute cloud programming," in *The 2010 Military Communications Conference*.

[5] G. Xu, Feng Xu and Hongxu Ma, "Deploying and researching Hadoop in Virtual Machines," in *Proceeding of the IEEE International Conference on Automation and Logistics*, Zhengzhou, China, August 2012.

[6] N. Nurain, Hasan Sarwar and Md.Pervez Sajjad, "AN In-Depth Study of Map Reduce in Cloud Environments," in *International Conference on Advanced Computer Science Applications and Technologies*, 2012.

[7] "Hardware Recommendations For Apache Hadoop," [Online]. Available: http://docs.hortonworks.com/HDPDocuments/HDP1/HDP-1.3.0/bk_cluster-planning-guide/content/typical-hadoop-cluster-hardware.html.