Deploying Apache™ Hadoop® with Colfax and Mellanox VPI Solutions

Background

Hadoop® has become the “go-to” framework for Map-Reduce systems. Increasing in popularity, Map-Reduce brings current and future users to look for systems which can deploy their application and framework’s in a scalable manner. While more data is helpful for understanding a problem, the challenge is whether the infrastructure used to process the data provides the expected return. In this white paper we describe the build of a Hadoop cluster, using Mellanox InfiniBand FDR interconnect and highly powerful and reliable servers from Colfax International. The Hadoop version described is the Apache™ community Hadoop.

To learn more about Hadoop we encourage you to review below links:

   
2. The described configurations and settings are used in the following Apache Hadoop
   
   A. Apache Hadoop 1.0.4

Hardware

To implement and test the technology, you will need:

• At least one Colfax Master Node (NameNode, Job Tracker)
• At least three Colfax Slave Nodes (DataNode, Task Tracker)
• Four or more Mellanox ConnectX®-3 adapter cards
• One or more InfiniBand FDR switches
• Four or more cables required for the ConnectX-3 card


In this article we will review a 5 node cluster configuration. Scaling the deployment is easily done by adding more Slave Nodes to the deployment. When scaling the deployment, take into consideration the amount of RAM you have in the Master Node, as well as the disk space.
High availability features are discussed within the above Apache Hadoop framework link. We recommend deploying two Master Nodes in master and secondary name node configuration.

**Recommended Server Configuration**

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Hardware Part</th>
<th>Specification</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Node (NameNode, Job Tracker)</td>
<td>System CPUs</td>
<td>Two, Quad core or more</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>32GB or Higher</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disk Drives</td>
<td>Two or More, 1TB each</td>
<td>RAID configuration</td>
</tr>
<tr>
<td>Slave Node (DataNode, Job Tracker)</td>
<td>System CPUs</td>
<td>Two, Quad core or more</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>24GB or Higher</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disk Drives</td>
<td>Four or More, 2TB each</td>
<td>JBOD configuration</td>
</tr>
</tbody>
</table>

**Table 1. Hadoop Server Recommended Configuration**

Use any of the Colfax International Servers from the below list to build a Master Node:

<table>
<thead>
<tr>
<th>Model</th>
<th>Memory</th>
<th>Disk Bays</th>
<th>Expansion Slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>CX1265i-X5</td>
<td>24 DIMMs, DDR3</td>
<td>8 Hot-Swap 2.5&quot;</td>
<td>2x PCIe x16</td>
</tr>
<tr>
<td>CX1260i-X5</td>
<td>24 DIMMs, DDR3</td>
<td>4 Hot-Swap 3.5&quot;</td>
<td>2x PCIe x16</td>
</tr>
</tbody>
</table>

**Table 2. Colfax Hadoop Master Node Server Configuration**

Use any of the Colfax International Servers from the below list to build a Slave Node:

<table>
<thead>
<tr>
<th>Model</th>
<th>Memory</th>
<th>Disk Bays</th>
<th>Expansion Slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>CX2265i-S24-X5</td>
<td>24 DIMMs, DDR3</td>
<td>24 Hot-Swap 2.5&quot; + 2 Internals</td>
<td>2x PCIe x8 Gen3</td>
</tr>
<tr>
<td>CX2265i-S16-X5</td>
<td>24 DIMMs, DDR3</td>
<td>16 Hot-Swap 2.5&quot; + 2 Internals</td>
<td>2x PCIe x8 Gen3</td>
</tr>
<tr>
<td>CX2235i-S24-X5</td>
<td>16 DIMMs, DDR3</td>
<td>24 Hot-Swap 2.5&quot; + 2 Internals</td>
<td>8x PCIe x8 Gen3, LPF</td>
</tr>
<tr>
<td>CX2235i-S16-X5</td>
<td>16 DIMMs, DDR3</td>
<td>16 Hot-Swap 2.5&quot; + 2 Internals</td>
<td>8x PCIe x8 Gen3, LPF</td>
</tr>
</tbody>
</table>

**Table 3. Colfax Hadoop Slave Node Server Configuration**

It is highly recommended to have larger RAM size on the master node to handle the cluster’s metadata, and to minimize the spill to the disks during this operation.

The above configuration is recommended for most use cases. There are several cases in which higher RAM and disk space is required. For such deployments, it is recommended that you contact us at bigdata@mellanox.com, where you can engage with one of our regional system engineers to help deploy your Hadoop cluster.

**Five Node using VPI Interconnect**

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Part Number</th>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>MC2207130-002</td>
<td>QSFP to QSFP cable, IB FDR, 2m</td>
<td><a href="http://www.mellanox.com/related-docs/prod_cables/DS_FDR_56Gb/s_Passive_Copper_Cables.pdf">http://www.mellanox.com/related-docs/prod_cables/DS_FDR_56Gb/s_Passive_Copper_Cables.pdf</a></td>
</tr>
</tbody>
</table>

**Table 4. VPI Hadoop Deployment Networking Bill of Materials**
Software Requirements

1. Supported OS
   a. RHEL 5.5, 5.6, 5.7, 5.8, 6.0, 6.1, 6.2, 6.3
      i. Corresponding CentOS distributions
         ii. SLES10 sp4, SLES11, SLES sp1, SLES sp2

2. Java Development Kit (JDK) version 1.6.0_25 or higher

3. Mellanox driver 1.5.3 or higher

4. The Hadoop distribution mentioned in section 1 above

Installation

The following section describes the installation of Hadoop on a Linux based machine(s). The supported Linux versions are described in the Software Requirements section.

Installing the Mellanox OpenFabrics Enterprise Distribution (OFED) driver

1. Download the Mellanox OFED driver iso from: http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=26&menu_section=34

2. Mount the iso (mount -o loop MLNX_OFED_LINUX-1.5.3-3.1.0-rhel6.3-x86_64.iso /mnt)

3. Install the missing packages
   a. For namenode (e.g. rhel/centos Software development workstation)
      i. yum install tcl tk
   a. For datanode (e.g. rhel/centos Basic Server)
      i. yum install tcl tk gcc-gfortran

4. cd /mnt
5. `./mlnxofedinstall`

6. Reboot

7. Run `connectx_port_config` (Choose the right config required InfiniBand or 40GbE Ethernet)

8. Run `service openibd restart`

9. Verify with the `ibstatus` command that you have the links active (e.g. port 1 InfiniBand, port 2 Ethernet)

   Infiniband device 'mlx4_0' port 1 status:
   default gid: fe80:0000:0000:0000:0002:c903:00fa:ced1
   base lid: 0x39
   sm lid: 0x2c
   state: 4: ACTIVE
   phys state: 5: LinkUp
   rate: 56 Gb/s (4X FDR)
   link_layer: InfiniBand

10. If you have the LinkUp, you are all set.

### Installing Hadoop

Using Mellanox interconnect provides two options of installation:

1. “Vanilla” – Installing Hadoop framework without taking advantage of the RDMA capabilities integrated within Mellanox’s end-to-end interconnect. In this mode the data flow will use the TCP/IP stack inherent with the Linux operating system in conjunction with Mellanox drivers.

2. Unstructured Data Accelerator (UDA) Based – Installing Hadoop framework and Mellanox’s UDA. In this mode the intermediate data flow will use the RDMA capabilities to accelerate the Map Reduce capabilities. Testing with large data sets (500GB and more) shows over 45% reduction in execution time. To learn more on Mellanox’s UDA please visit: http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=144&menu_section=69

### The “Vanilla” Option

Installing Apache Hadoop Distribution 1.0.4 using Colfax and Mellanox high-end servers and networks capabilities.

1. Setup the required network (In the example below we add –ib for InfiniBand). You will need to edit the portion of the `HADOOP_PREFIX/conf/hadoop-env.sh` “NODENAME” to reflect the correct hostname used for the cluster. All hostnames should have DNS setup as well.

2. Download JDK 1.6.x and install (The install location will be your `$JAVA_HOME`) on all nodes.

3. Update the `.bashrc` with `$JAVA_HOME` and change the path to include this as the first choice

4. Add line “export HADOOP_PREFIX=$HOME/hadoop-1.0.4”

5. Copy `.bashrc` to all the nodes

6. Plan on the disk that will be used for hadoop and you can use `preparedisks.sh` in `$HOME/hadoop-scripts` directory
   a. Use with caution you need edit the script for the disks you need to initialize or you may lose data on your existing disks

7. Create a simple `hduser` login on all nodes

8. Untar the hadoop-scripts on the home directory of `hduser`

9. Download http://download.nextag.com/apache/hadoop/common/hadoop-1.0.4/hadoop-1.0.4.tar.gz (You can use `wget`)

10. `cd hadoop-scripts`

11. run the `crsshkeys.sh` script to generate a passwordless ssh login on all nodes (ex: `./crsshkeys.sh`
hydra001 thru 5). This script creates authorized keys in the .ssh directory

12. chmod g-w ~/.ssh/authorized_keys
13. scp $HOME/.ssh/.authorized_keys hduser@hydra002 (run the same for all the nodes)
14. Test ssh works without password (ssh hydra002)
15. Modify the $HOME/hadoop-scripts/runcmdall.sh script to your cluster name and needs
16. Use the runcmdall.sh script to untar the hadoop-1.0.4.tar.gz on all nodes
17. Check if the $JAVA_HOME is set and java – version does report the JAVA version you have installed (java –version)
   a. [hduser@hydra001-ib ~]$ java -version
   b. java version “1.6.0_33”
   c. Java(TM) SE Runtime Environment (build 1.6.0_33-b04)
   d. Java HotSpot(TM) 64-Bit Server VM (build 20.8-b03, mixed mode)
18. Login from the namenode to all the other nodes to add the host id’s or disable the key checking
19. mv $HOME/ hadoop-1.0.4/conf $HOME/ hadoop-1.0.4/conf.empty
20. Copy the conf files to $HOME/ hadoop-1.0.4/conf
21. Modify the files masters, slaves, core-site.xml, hdfs-site.xml, mapred-site.xml, hadoop-env.sh files to suit your environment
22. scp –r $HOME/ hadoop-1.0.4/conf hduser@<not other nodes>:/$HOME/ hadoop-1.0.4/conf
23. $HOME/hadoop-scripts/runcmdall.sh “mkdir -p /data01/hduser/dfs/nn /data02/hduser/dfs/nn”
24. $HOME/hadoop-scripts /runcmdall.sh “mkdir -p /data01/hduser/dfs/dn /data02/hduser/dfs/dn”
25. $HOME/hadoop-scripts /runcmdall.sh “mkdir -p /data01/hduser/mapred/local /data02/hduser/ mapred/local”
26. $HOME/hadoop-scripts/runcmdall.sh “chmod go-w /data01/hduser/dfs/dn /data02/hduser/ dfs/dn ” – Verify the permissions on the datanode slices
   a. It should be: drwxr-xr-x 6 hduser hduser 4096 Feb 28 11:23 /data01/hduser/dfs/dn
27. $HADOOP_PREFIX/bin /hadoop namenode –format –Answer “Y”
28. Start HDFS service
   a. $HADOOP_PREFIX/bin/start-dfs.sh
29. Verify using the jps command if the namenode, secondarynamenode and datanodes in other
   nodes working.
   a. Namenode should show
   b. [hduser@hydra001-ib hadoop-1.0.4]$ jps
   c. 4731 Jps
   d. 3607 NameNode
   e. 3993 SecondaryNameNode
   f. [hduser@hydra001-ib hadoop-1.0.4]$ jps
   g. Datanode will show “DataNode”
30. Create required tmp HDFS directories
   a. $HADOOP_PREFIX/bin/hadoop fs -mkdir /tmp
   b. $HADOOP_PREFIX/bin/hadoop fs -chmod -R 1777 /tmp
31. Verify all nodes are up and storage is being shown correctly
   a. $HADOOP_PREFIX/bin/hadoop dfsadmin –report –Answer “Y”
32. Start mapreduce services
   a. $HADOOP_PREFIX/bin/start-mapred.sh
33. Verify using the jps command if the namenode, secondarynamenode and datanodes in other
   nodes working.
   a. Namenode should show
   b. [hduser@hydra001-ib hadoop-1.0.4]$ jps
c. 4731 Jps
d. 3607 NameNode
e. 3993 SecondaryNameNode
f. 4125 JobTracker
g. [hduser@hydra001-ib hadoop-1.0.4]$  
h. Datanodes (all other nodes) should show “DataNode” & “TaskTracker”

34. Run the terasort to verify the cluster is working fine
   a. $HOME/hadoop-scripts/runterasort.sh
   b. Check the namenode ip ex: http://hydra001:50030 – You should see the Job Tracker page with the jobs running

35. If you see the Terasort job completed on the JT page, You are all set!!

Adding the UDA Package on top of Vanilla.

Make sure the Mellanox ConnectX®-3 cards are properly installed on your Name Node and Data Nodes before starting the UDA installation.

To install UDA, you should first follow the Hadoop installation in the “Vanilla Option” section.

After successfully installing the “vanilla” Hadoop version, follow these next steps:

Set the ulimit to unlimited:

```
ulimit -l unlimited
```

Increase the maximum number of memory translation table segments per HCA

Check for the following settings in: /etc/modprobe.d/ib_ipoib.conf

```
“options mlx4core log_num_mtt=XX”
```

If present, check the value of mtt and based on your memory footprint, this value needs to be adjusted (Ex: 64Gb of memory, you can set this to 24). More information on this can be obtained here: http://www.open-mpi.org/faq/?category=openfabrics#ib-low-reg-mem).

If not present, create a modified.conf with the setting:

```
echo “options mlx4_core log_num_mtt=24” > /etc/modprobe.d/mofed.conf
```

Reboot the server for the settings to take effect.

UDA Integration (To be executed for all nodes)

Patch the plugin (describe below is the CDH3u4 and Hadoop 0.20.2 patch)

```
cd ../<hadoop dir> (ex: cd ../$HADOOP_HOME , )  
ls –ld hadoop-0.20.2-cdh3u4  
drwxr-xr-x. 17 root root 4096 Sep  4 04:58 hadoop-0.20.2-cdh3u4  
patch -p0 < cdh3u4.patch  
cd <hadoop dir> (ex: cd /usr/lib/hadoop-0.20.2-cdh3u4)  
run ant
```

Copy the jar files from the build directory again to $HADOOP_HOME

Install the UDA RPM

```
rpm –ivh libuda-3.0.1-4453.el6.x86_64.rpm
```

Verify the rpm install:

```
# rpm -ql libuda  
/usr/lib64/uda/libuda.so  
/usr/lib64/uda/set_hadoop_slave_property.sh  
/usr/lib64/uda/uda-CDH3u4.jar  
/usr/lib64/uda/uda-hadoop-0.20.2.jar  
/usr/lib64/uda/uda-hadoop-1.x.jar  
/usr/share/doc/libuda-3.0.1/LICENSE.txt  
/usr/share/doc/libuda-3.0.1/README
```
Add UDA jar to classpath of hadoop-env.sh:

```
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH":/usr/lib64/uda/uda-CIH3u4.jar
```

The Jar file would be different if you using a different distribution

**UDA Configuration**

Add the following properties in the files mentioned. For more information on these properties, please refer to the "Mellanox Unstructured Data Accelerator Quick start guide".

1. File hdfs-site.xml
   ```xml
   <property>
     <name>dfs.datanode.dns.interface</name>
     <value>ib0</value>
   </property>
   ```

2. File mapred-site.xml
   ```xml
   <property>
     <name>mapred.rdma.setting</name>
     <value>1</value>
   </property>

   <property>
     <name>mapred.rdma.buf.size</name>
     <value>1024</value>
   </property>

   <property>
     <name>mapred.map.tasks.speculative.execution</name>
     <value>false</value>
   </property>

   <property>
     <name>mapred.reduce.tasks.speculative.execution</name>
     <value>false</value>
   </property>

   <property>
     <name>mapred.rdma.cma.port</name>
     <value>1</value>
   </property>

   <property>
     <name>mapred.rdma.cma.port</name>
     <value>9011</value>
   </property>

   <property>
     <name>mapred.reduce.slowstart.completed.maps</name>
     <value>0.95</value>
   </property>

   <property>
     <name>mapred.rdma.wqe.per.conn</name>
     <value>1024</value>
   </property>

   <property>
     <name>mapred.tasktracker.shuffle.provider.plugin</name>
     <value>com.mellanox.hadoop.mapred.UdaShuffleProviderPlugin</value>
   </property>
   ```
<name>mapred.reducetask.shuffle.consumer.plugin</name>
     <value>com.mellanox.hadoop.mapred.UdaShuffleConsumerPlugin</value>
</property>

Testing UDA functionality

Execute a Terasort test.

For example: Execute a 300GB Tergen and Terasort job.

\[
\text{hadoop jar }$\text{HADOOP_HOME/hadoop-examples-*.jar teragen 3000000000 /users/hadoop/teragen-input}
\]

\[
\text{hadoop jar /usr/lib/hadoop-0.20/hadoop-examples-*.jar terasort /users/hadoop/terasort-input /users/hadoop/terasort-output}
\]

UDA troubleshooting

1. Verify the plugin ability patch inside the hadoop jar
   \[
   \text{jar -tf /usr/lib/hadoop/hadoop-core-*.jar | grep ShuffleConsumerPlugin.class}
   \]

2. Verify the UDA rpm installation exist
   \[
   \text{rpm -qa | grep -i uda}
   \]

3. Verify the UDA configuration parameters are set
   \[
   \text{grep -i uda <hadoop configuration directory>}
   \]

4. Examine tasktracker log files for any memory errors
   \[
   \text{Ex: "MSG=Cannot allocate memory (errno=12)" – This error shows that the mtt value + number of reducers are not able to allocate memory. Reduce the number of reducers or decrease the mtt value based on the guideline provided. More information is provided in the tuning section of the quick start guide.}
   \]

Scaling the Cluster Size

Adding nodes or building a cluster with more nodes than a single rack can contain, is a common practice. The installation of servers and the network should adhere to the target application performance requirements. Additional nodes provides additional storage space and compute power.

Scaling beyond the single switch requires the installer to take into consideration the needed throughput of the single server and the rack.

In an “All-to-All” setting, we’ve found that at least 10Gb of true bandwidth is required in order to scale effectively.

For example we can consider a 120 nodes cluster. The diagram below shows the suggested configuration:

\[Figure 4: \text{Example: Scaling to 120 nodes}\]
In the above example, where nodes are connected with an InfiniBand FDR 56Gb/s fabric, the All-to-All available bandwidth will be 18.6Gb/s.

Scaling to larger clusters is done in the same fashion. Connection ToR switches with enough bandwidth to satisfy nodes throughputs.

**High Availability**

When considering High Availability (HA) features, one should take advantage of the framework capabilities. For the interconnect consideration, there are several options to consider:

The first option would be doubling the number of switches and cables by using a dual rail configuration. Dual rail configuration is enabled by using a second port on the server’s adapter card connected to a second switch. In this configuration, the node is connected to two fabrics in parallel, eliminating any single point of failure, in terms of connectivity from the server to its adjacent nodes.

The second option would be adding a secondary networking card to the servers and using it as the failover point, in the event the primary card fails or “hangs off”. In such a configuration, the number of switch ports required is doubled.

The last option would be combining the first two options and doubling both the adapter cards installed and the number of switches in the configuration.

---

**Appendix: Setup Scripts**

File: checkconfig.sh

```bash
echo "Check Hadoop Home"
echo $HADOOP_HOME
echo "Hadoop Config Dir"
echo $HADOOP_CONF_DIR
echo "Current Active config"
ls -ld /etc/hadoop/conf
echo "Current active binary config"
ls -ld /usr/lib/hadoop*
echo "Checking the conf directory on the HADOOP_HOME"
ls -ld /usr/lib/hadoop-0.20/conf
```

File: checkdns.sh

```bash
nslookup `hostname`
ping -c 1 `hostname`
```

File: cleanlogs.sh

```bash
rm -rf /var/log/hadoop/*.out* /var/log/hadoop/*.log* /var/log/hadoop/metrics/*.log
/var/log/hadoop/SecurityAuth.audit /var/log/hadoop/job*.xml /var/log/hadoop/userlogs/*
touch /var/log/hadoop/metrics/dfsmetrics.log
touch /var/log/hadoop/metrics/jvmmetrics.log
touch /var/log/hadoop/metrics/mrmetrics.log
/chown hdfs:hdfs /var/log/hadoop/metrics/SecurityAuth.audit
/chown hdfs:hadoop /var/log/hadoop/metrics/jvmmetrics.log
/chown mapred:mapred /var/log/hadoop/metrics/mrmetrics.log
/chown hdfs:hadoop /var/log/hadoop/metrics
/chown hdfs:hadoop /var/log/hadoop/metric
/chmod g+rwx /var/log/hadoop/metrics/dfsmetrics.log
/chmod g+rwx /var/log/hadoop/metrics/jvmmetrics.log
/chmod g+rwx /var/log/hadoop/metrics/mrmetrics.log
/chmod g+rwx /var/log/hadoop/metrics/SecurityAuth.audit
```

©2013 Mellanox Technologies. All rights reserved.
chmod g+rw /var/log/hadoop
chmod g+rw /var/log/hadoop/metrics

File: create-hadoop-sysusers.sh

groupadd -r hdfs

groupadd -r mapred

groupadd hadoop

useradd -r -g hdfs -G hadoop -c 'Hadoop HDFS' -d /usr/lib/hadoop-0.20  hdfs

useradd -r -g mapred -G hadoop,hdfs - c 'Hadoop MapReduce' -d /usr/lib/hadoop-0.20 mapred

useradd -g hadoop - G hdfs -m -c 'Hadoop User' -d /home/hadoop/hadoop

File: cdhdfsdirs.sh

# This script creates all required HDFS directories for the
# cluster including the user of the cluster hadoop

cd $HADOOP_HOME

sudo -u hdfs bin/hadoop fs -chown -R hdfs:hadoop /

sudo -u hdfs bin/hadoop fs -chmod go+rx /

sudo -u hdfs bin/hadoop fs -chown go-w /

sudo -u hdfs bin/hadoop fs -mkdir /tmp

sudo -u hdfs bin/hadoop fs -chmod -R 777 /tmp

sudo -u hdfs bin/hadoop fs -chown mapred: hadoop /mapred/system

sudo -u hdfs bin/hadoop fs -mkdir /user/hadoop

sudo -u hdfs bin/hadoop fs -chown -R hadoop: hadoop /user/hadoop

sudo -u hdfs bin/hadoop fs -chmod go-rwx /mapred/system

sudo -u hdfs bin/hadoop fs -ls /

sudo -u hdfs bin/hadoop fs -ls /mapred/system

File: crsshkeys.sh

ssh-keygen -t rsa - P "" -f ~/.ssh/id_rsa

ssh-keygen -t rsa - P "" -f ~/.ssh/id_rsa

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

ssh root@hydra002-ib " ssh-keygen -t rsa - P "" -f ~/.ssh/id_rsa"

ssh root@hydra002- ib cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

ssh root@hydra003-ib " ssh-keygen -t rsa - P "" -f ~/.ssh/id_rsa"

ssh root@hydra003-ib cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

scp ~/.ssh/authorized_keys  root@hydra002-ib:/root

scp ~/.ssh/authorized_keys root@hydra003-ib:/root

File: initialize-cluster.sh

rm -rf /data1/dfs/nn/* /data1/dfs/dn/* /data1/mapred/local/*

rm -rf /data2/dfs/nn/* /data2/dfs/dn/* /data2/mapred/local/*


rm -rf /data5/dfs/nn/* /data5/dfs/dn/* /data5/mapred/local/*

File: newslice-fixperm.sh

# Create the /data?? directories and initialize with the
# directories for namenode, datanode & mapred

mkdir -p /data01/dfs/nn

mkdir -p /data01/dfs/dn
mkdir -p /data01/mapred/local
chown -R hdfs:hadoop /data01
chown -R hdfs:hadoop /data01/dfs
chmod -R 700 /data01/dfs
chown -R mapred:hadoop /data01/mapred
chmod -R 755 /data01/mapred

…. For all data nodes

#Create the metrics and log directories

mkdir -p /var/log/hadoop/metrics
mkdir -p /var/log/hadoop/userlogs
chown -R hdfs:hadoop /var/log/hadoop
chown -R mapred:mapred /var/log/hadoop/userlogs

#Create the directory for hadoop pid's

mkdir -p /var/hadoop
chown hdfs:hadoop /var/hadoop
chmod g+rwX /var/Hadoop

File: prepareddisks.sh

# ***Use this script with caution*** It can wipe the entire disk
# clean ** this script shows an example of 3 disks
# sdb,sdc & sdd.

parted /dev/sdb mkpart primary ext4 0% 100%
mkfs.ext4 /dev/sdb1

parted /dev/sdc mkpart primary ext4 0% 100%
mkfs.ext4 /dev/sdc1

parted /dev/sdd mkpart primary ext4 0% 100%
mkfs.ext4 /dev/sdd1

File: runcmdall.sh

# Use this script to run commands on all clusters or scripts from # the same directory
# ex: ./runcmdall “ls -l /etc/hadoop/conf” shows all files in the # conf diretory

echo “Running on Hydra-2”
ssh root@hydra002 $1
echo “Running on Hydra-3”
ssh root@hydra003 $1
echo “Running on Hydra-4”
ssh root@hydra004 $1
echo “Running on Hydra-5”
ssh root@hydra005 $1
echo “Running on Hydra-1”
ssh root@hydra001 $1
File: testdfsio.sh

```bash
cd $HADOOP_HOME
sudo -u hdfs bin/hadoop jar $HADOOP_HOME/hadoop-test-*.jar TestDFSIO -write -nrFiles 10 -fileSize 1000
sudo -u hdfs bin/hadoop jar $HADOOP_HOME/hadoop-test-*.jar TestDFSIO -read -nrFiles 10 -fileSize 1000
sudo -u hdfs bin/hadoop jar $HADOOP_HOME/hadoop-test-*.jar TestDFSIO -clean
```