MASI Lab hadoop-hbase cluster setup instruction V1.0

SHUNXING BAO

Frederick Weitendorf

1. Warning 2
2. Create hadoop user 2
3. Install vim 3
4. Edit hosts file. 3
5. Setup passphraseless ssh 4
6. Setup hadoop distributed cluster 6
7. Setting up an hbase distributed cluster 11
8. Deploy and install hadoop & hbase on a new server 14
9. Setup yarn configruation 16
10. Start hadoop, hbase and yarn service 19
11. Maintain hadoop, hbase cluster 20
12. Commission and Decommission Nodes 27
13. Stop Hadoop Distributed File System 28
14. CHECKLIST 29

# Warning

## DO NOT INCLUDE kyuss, mastadon and masigate as Hadoop cluster node.

## Make sure that if a computer is being used a workstation, that you ask that person if they are ok with their workstation being added to the Hadoop cluster.

# Create hadoop user

## Create a Hadoop user account with a home directory

### Our Hadoop cluster user name is ‘hadoop’, with password ‘hadoop’.

#### Login to a new server with the admin account ‘masi’ by command ‘ssh masi@xxx, where ‘xxx’ is the new server alias (when promoted type admin’s password)

#### Ask Prof. Landman for the password if you do not have it

#### Type command ‘ sudo adduser hadoop’ (when promoted type admin’s password)

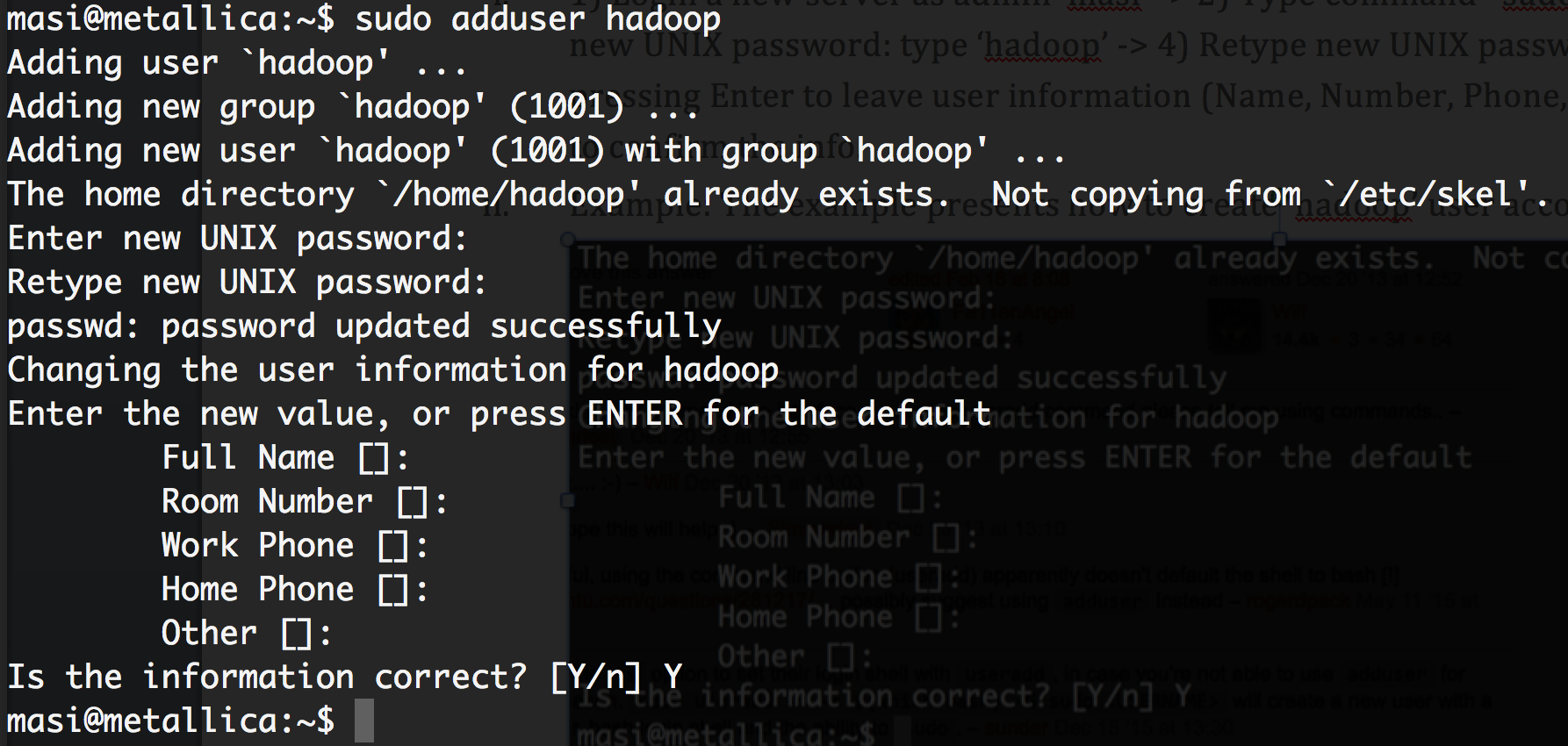
#### Enter new UNIX password: hadoop

#### Retype new UNIX password: hadoop

#### Keep pressing Enter to leave user information (Name, Number, Phone, Other) as default

#### Type ‘Y’ to confirm the info.

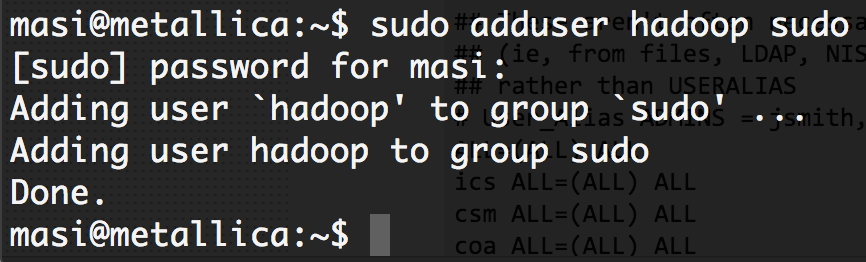
### Example: The example presents how to create ‘hadoop’ user account on server ‘metallica’.



## Grant sudo privileges to an existing user

### Type command ‘sudo adduser hadoop sudo’

### Example: The example presents how to grant ‘hadoop’ user with sudo privileges on server ‘metallica’.

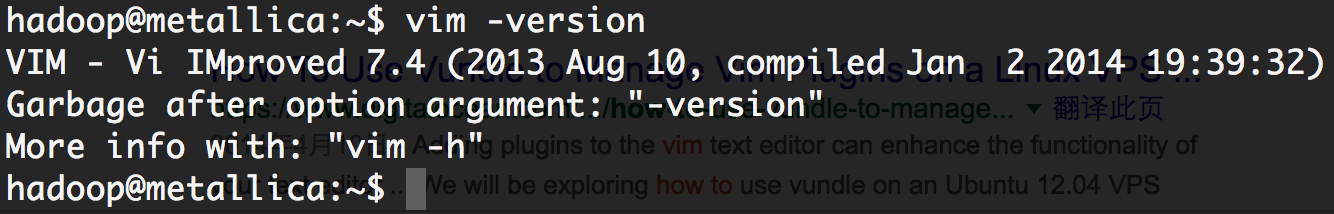


## Now you can login your designated new server using account ‘hadoop’ with password ‘hadoop’.

# Install vim

We use secure shell (ssh) to remotely login to a server. Most of the time we can only use the terminal to edit a file’s location in a remote server without a user interface. Here we recommend to use the VIM text editor to edit files remotely.

## SSH a server node with the newly created ‘hadoop’ account -> type command ‘vim -version’ to check VIM text editor version on the server. You should get version info of your VIM as follows.



## If you cannot get similar info above, you should install VIM on server with following two commands (when promoted type hadoop user’s password):

***sudo apt-get update***

***sudo apt-get install vim***

## Then retype command ‘vim -version’ to verify your install process. Now you should be able to see the VIM text editor version.

# Edit hosts file.

## The Hadoop cluster is VERY sensitive with IP addresses, especially the private host name of a machine. We should not include any private host names in our server’s host file. The private host names are defined in the Cluster’s hosts file: /etc/hosts, and this file is refreshed every 15 minutes by /etc/hosts.local. In order to make sure Hadoop cluster is not affected by private host names, two files should be edited.

## Edit /etc/hosts

### Type command ‘sudo vim /etc/hosts’ to edit the cluster’s hosts file. Since the hosts file is a system file, you have to use sudo permission to edit this file. (When promoted type hadoop user’s password)

### Press ‘i’ to enter VIM Insert mode -> move cursor to the lines that define private host names.

### Comment/remove those lines by typing ‘#’

### type ESC button to quit VIM Insert mode, type ‘:wq’ and ENTER to save your edit.

### Example: Edit /etc/hosts file on server metallica in VIM with Insert mode. Comment/remove private host names ‘127.0.1.1 metallica.ds.vanderbilt.edu metallica’ and ‘::1 metallica.ds.vanderbilt.edu metallica localhost6’ by ‘#’. Save your edit.

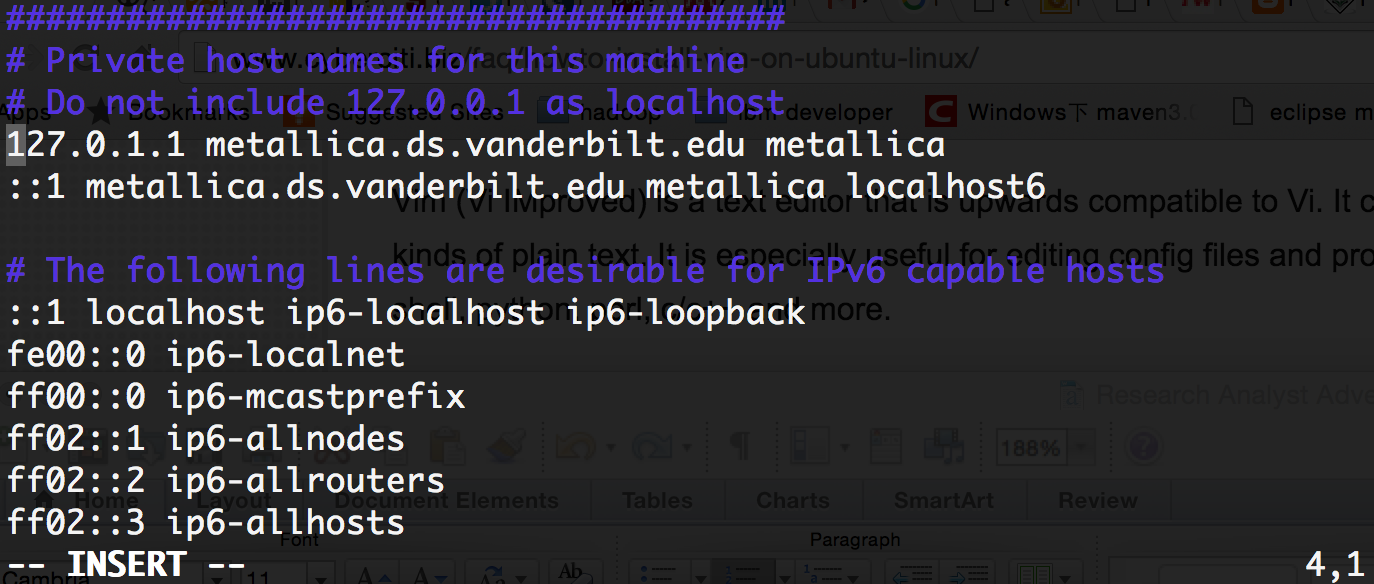


Fig-a INSERT mode

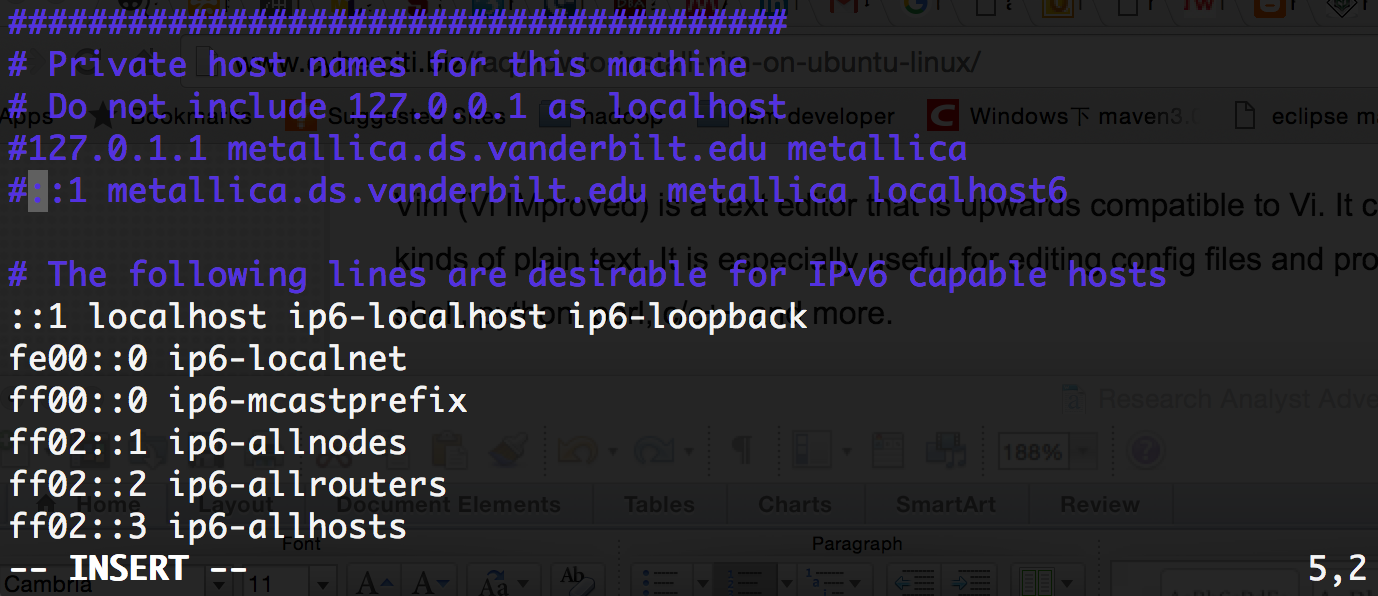


Fig-b Comment/remove private host names

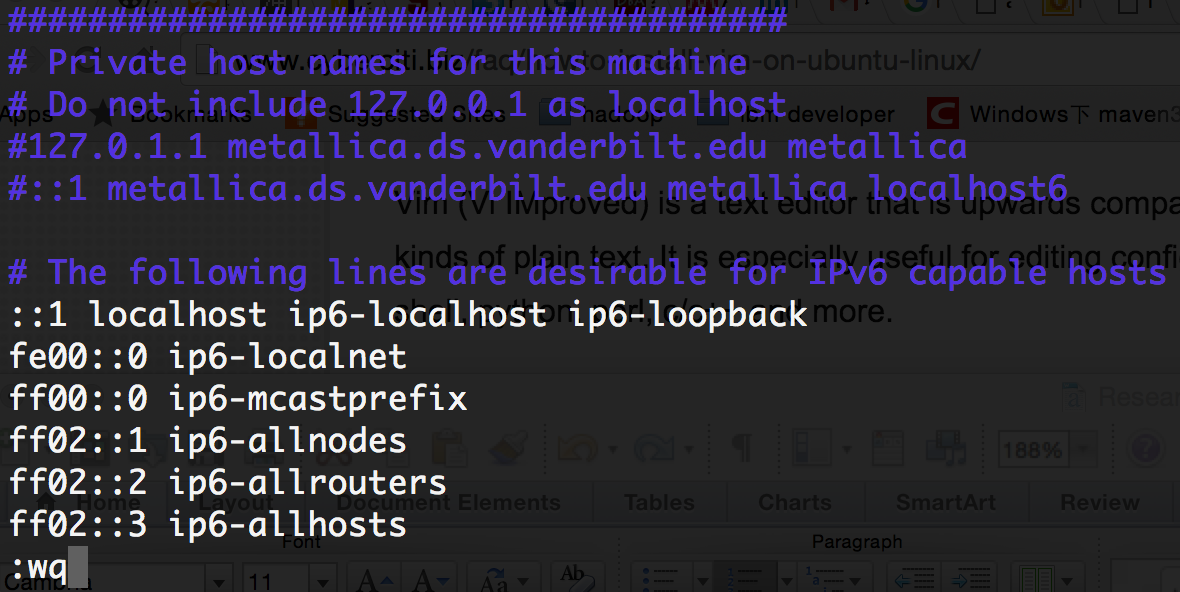


Fig-c Quit INSERT mode by press ‘ESC’-> type ‘:wq’ ->press ENTER

## Edit /etc/hosts.local

### Comment/remove the private host names in /etc/hosts.local file with same steps.

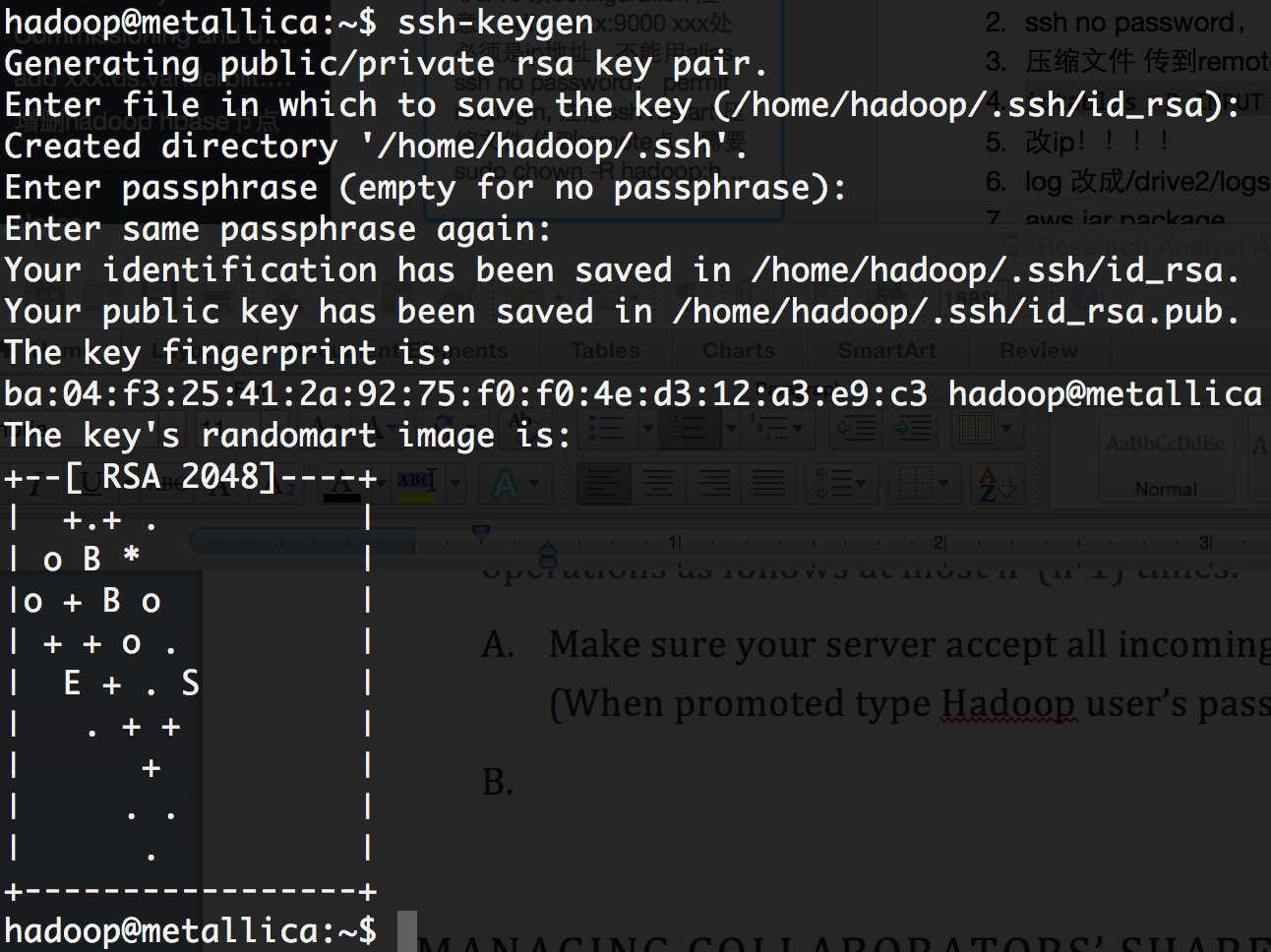
# Setup passphraseless ssh

When remote logging in or executing a command on a remote server with the same user account, you need to type your hadoop password. Once the hadoop cluster grows larger, typing the same password multiple times is annoying. Passphraseless ssh can save you a lot of time without typing same password so many times. For your convenience, the passphraseless ssh should be done pairwise. It means if your cluster size is n, then you need to this operation as follows at most n\*(n-1) times.

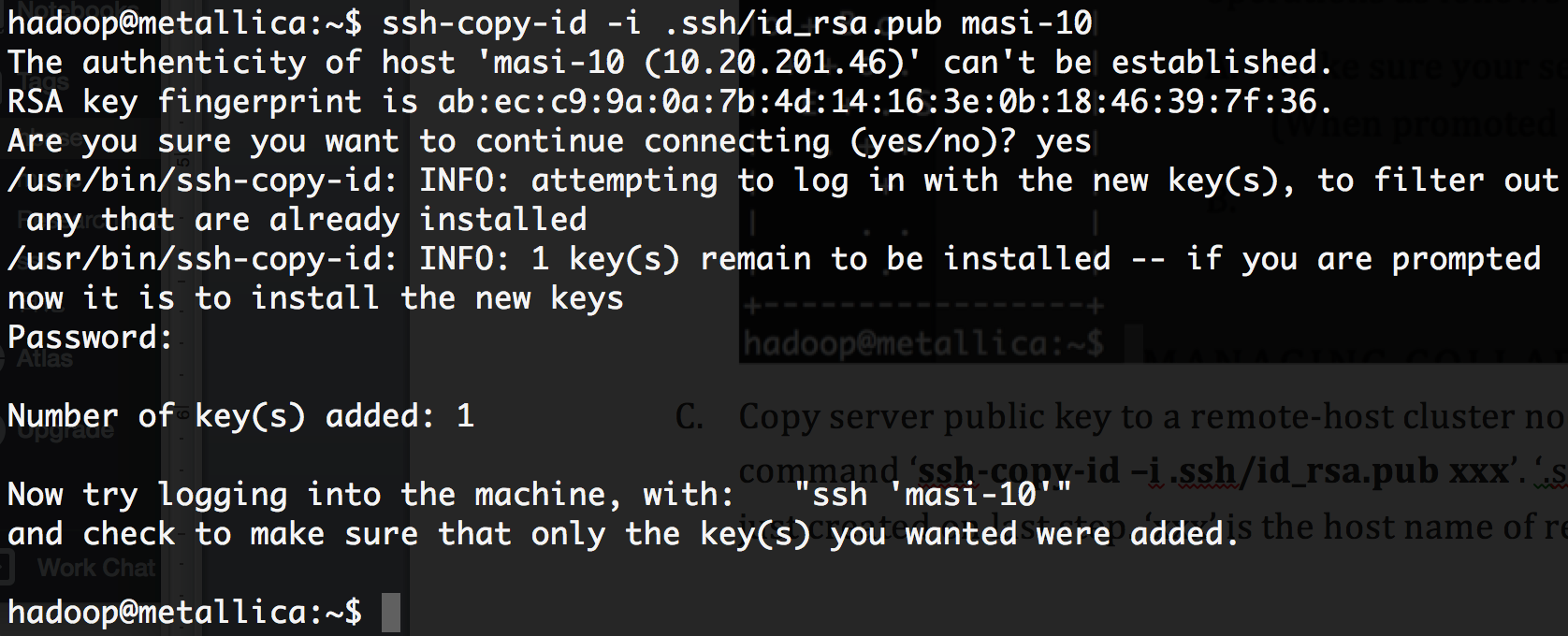
However, this can also be very time consuming to do as the cluster grows in size. It is most important that, if you do not set up passphraseless ssh with all the nodes, that you at least set send ssh keys between the namenode and the new node.

## Make sure your server accepts all incoming request via the command ‘sudo iptables –P INPUT ACCEPT’ (When promoted type Hadoop user’s password)

## Generate default public/private rsa key pair by command ‘ssh-keygen’, simply keep pressing Enter. **DO** **NOT** generate a new key on nodes that have already been added, as you will then need to start over setting up passphraseless ssh for that node and all other nodes it had been previously configured for.



## Copy the servers public key to a remote-host cluster node you want to login without password with the command ‘ssh-copy-id -i ~/.ssh/id\_rsa.pub xxx’. ‘.ssh/id\_rsa.pub’ is the default public key you’ve just created on last step, where ‘xxx’ is the host name of remote machine. (Type ‘yes’ when first ssh-ing a remote host. When prompted type the Hadoop user’s password)



## Fig. ‘masi-10’ is the remote host name

## Now you can login to a remote machine without a password. i.e. ‘ssh xxx’ can log in server ‘xxx’ with ‘hadoop’ account. If you still need to type a password, please try the following approach:

### Log in remote host with prompted Hadoop password

### Type command: chmod o-w ~/

### Type command: chmod 700 ~/.ssh

### Type command: chmod 600 ~/.ssh/authorized\_keys

### Log in to your previous machine and try to log in remote host to verify passphraseless ssh

## Double check if “hadoop” account is allowed to remote access.

## Go to /etc/security/access.conf

## Check if there is a line:

## + : masi hadoop : ALL

## If not, please add it.

# Setup hadoop distributed cluster

This section aims to give you an overview of configuration settings that should be noticed and have been tuned by the authors. All nodes use the same version of Hadoop, so we can configure Hadoop on the Master node first with the vim text editor (‘masi-10’ is the current master node in MASI Lab), and then sending the compressed Hadoop package with well-edit configuration to slave nodes.

The default home directory for our Hadoop cluster is installed in /usr/local/hadoop. The configuration files of hadoop are saved in /usr/local/hadoop/etc/hadoop/. Now lets go through the configuration file we should edit in /usr/local/hadoop/etc/hadoop/. The screenshot in this section are all from file’s vim text editor.

## [hadoop-env.sh](http://hadoop-env.sh/)

### JAVA\_HOME

#### As the description says, the only required environment variable for the Hadoop cluster is JAVA\_HOME. All of MASI Lab uses the same version of Java, so you can simply use the setting as follows.

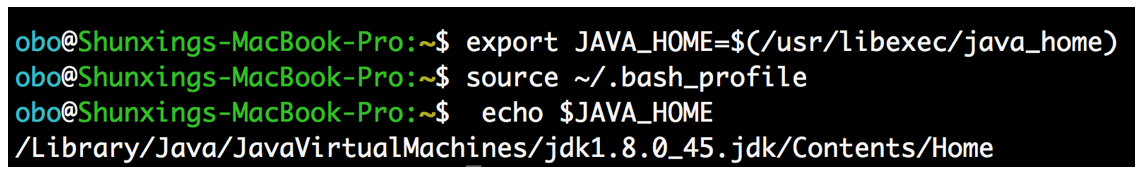
### ../Desktop/JAVA%20HOME.png

### Generally, for a Linux user, you can find it via



~~Here /usr/lib/jvm/java-7-openjdk-amd64 is your JAVA\_HOME path.~~ ( Since our cluster has been upgraded to 16.04, our current JAVA\_HOME path is /usr/lib/jvm/java-1.8.0-openjdk-adm64)

For a Mac user,



Here /Library/Java/JavaVirtualMachines/jdk1.8.0\_45.jdk/Contents/Home is your JAVA\_HOME path.

### JAVA runtime option

Hadoop does not support IPv6 currently. Please make sure you Hadoop-env.sh disables IPv6 by setting the Java runtime option correctly as follows.

### ../Desktop/ipv4.png

## slaves

## This file defines slave nodes information.

## ***<Note>*** *MASI Lab provides same standard (share standard DNS) for representing each machine. Fig-x presents a part of /etc/hosts file where hostnames are mapped to IP address. The first column is the IP address of a remote server. The second column lists a fully qualified domain name. And the third column is the hostname/alias of a server.*

## ../Desktop/6%20server%20DNS.png

## Fig-x.

## In the slaves file, you can use either IP address, hostname/alias, or a fully qualified domain name to specify a slave. For easier recognition, the author recommends using hostname/alias when referring to a remote server node.

## ../Desktop/7%20slaves.png

## When including a new server as a slave node, input the hostname of server to file slaves.

## Multiple XML configuration files.

When adding/editing an xml file, please acquaint yourself with the XML format first:

***<configuration>***

***<property>***

***<name>*** *Property Name* ***</name>***

***<value>*** *Property Value* ***</value>***

***<description>*** *Property description (Can be NULL).* ***</description>***

***</property>***

***</configuration>***

***note:*** *new properties that start/end with <property></property> block should not be embedded into other propertys’ blocks, and property blocks must be written within the block of <configuration></configuration>*

### core-site.xml

Site-specific configuration for a given Hadoop installation. This file is a global configuration, meaning all nodes in a Hadoop cluster use the same setting in core-site.xml. Currently two properties have been set in this file.

#### hadoop.tmp.dir

#### The place to stores **ALL** temporary data. The default location is a system’s /tmp directory. Once a machine reboots, this data will be lost, so we need to define an another place to store data. Each MASI Lab server should have a 2 TB hard drive for Hadoop usage, with the mounted directory being /drive2. See section 8.4 if there is no mounted /drive2 for installation and troubleshooting information.

#### ../Desktop/Screen%20Shot%202016-04-13%20at%206.35.23%20PM.p

#### When extending Hadoop nodes, you do not need to create a new folder of /drive2/hadoop/tmp hierarchically by yourself. Hadoop will create them automatically.

#### fs.defaultFS

*The name of the default Hadoop distributed file system[reference-1* *https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/core-default.xml]. 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. e.g. The default hdfs port is 9000, 10.20.201.46 is the IP address of master node (masi-10). The value of fs.defaultFS* ***MUST*** *start with ‘hdfs://’.*

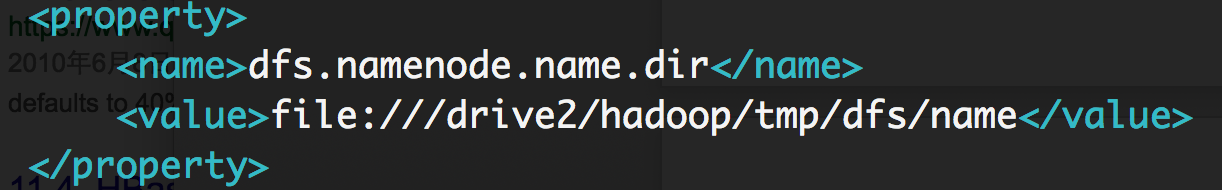
#### ../Desktop/Screen%20Shot%202016-04-13%20at%206.35.31%20PM.p

### hdfs-site.xml

### Namenode, Datanode, SecondaryNameNode and Replication information are defined in this file [reference https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml].

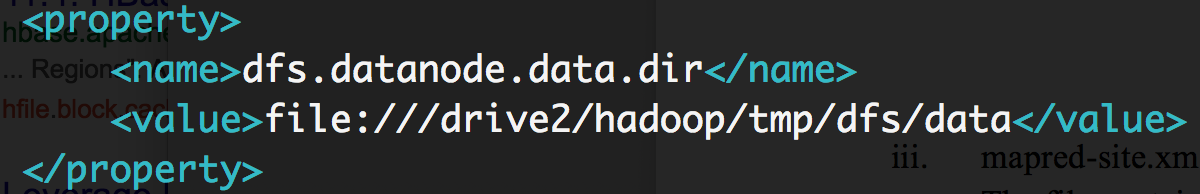
#### dfs.namenode.name.dir

*Determines where on the local filesystem the DFS namenode should store the name table(fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.*



#### dfs.datanode.data.dir

*Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices. The directories should be tagged with corresponding storage types ([SSD]/[DISK]/[ARCHIVE]/[RAM\_DISK]) for HDFS storage policies. The default storage type will be DISK if the directory does not have a storage type tagged explicitly. Directories that do not exist will be created if the local filesystem allows.*

**

#### dfs.namenode.secondary.http-address

*The secondary namenode http server address and port.*

#### ../Desktop/Screen%20Shot%202016-04-13%20at%206.38.06%20PM.p

#### dfs.replication

*Default block replication. The actual number of replications can be specified when a file is created. The default,which is 3 replications, is used if replication is not specified in create time.*

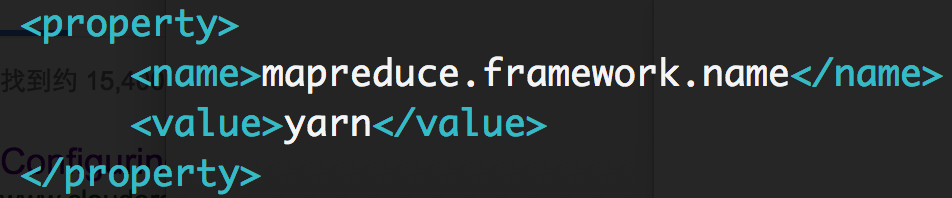


### mapred-site.xml

### The file contains configuration information that overrides the default values for MapReduce’s parameters. When first Hadoop is first downloaded, there is no such file. You need to copy mapred-site.xml.template and save as mapred-site.xml with further edit. This file is customized, and you need to tune the parameter for your own purpose. As follows is just an example that authors use for their experiment setup [reference https://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml].

#### mapreduce.framework.name

*The runtime framework for executing MapReduce jobs. Can be one of local, classic or yarn. For Hadoop v2.0+, we use ‘yarn’ for MapReduce job scheduling and resource management. The yarn configuration will be introduced in Section 8.*



#### mapreduce.map.memory.mb

*The amount of memory to request from the scheduler for each map task. Here we set 2GB for each map.*

### ../Desktop/Screen%20Shot%202016-04-13%20at%206.40.13%20PM.p

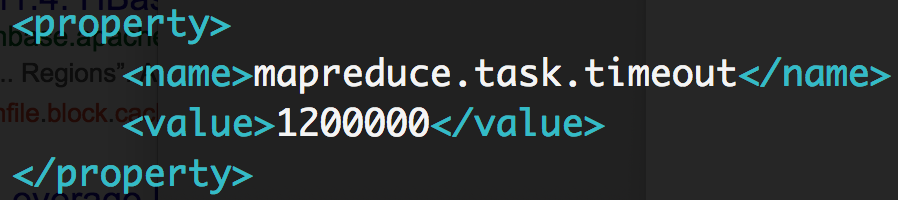
#### mapreduce.map.java.opts

*Larger heap-size for child jvms of maps.*

#### ../Desktop/Screen%20Shot%202016-04-13%20at%206.40.48%20PM.p

#### mapreduce.task.timeout

*The number of milliseconds before a task will be terminated if it neither reads an input, writes an output, nor updates its status string (like a heartbeat). A value of 0 disables the timeout. The default value is 600000 milliseconds (10 minutes). The setting here is an empirical value based on authors’ experiment.*

**

#### mapreduce.jobhistory.address

*MapReduce JobHistory Server IPC host:port. The default port is 10020. This setting is only need to set in Master node.*

#### ../Desktop/Screen%20Shot%202016-04-13%20at%206.42.47%20PM.p

# Setting up an hbase distributed cluster

The MASI Lab HBase cluster is a distributed directory built upon the lab’s hdfs. HMaster is located on the same node of Hadoop NameNode. Each machine has a RegionServer and Datanode for data locality. The cluster uses the default Zookeeper that is provided by HBase (please check setting 6.A.ii).

Similarly with our Hadoop cluster setup, the default home directory for our HBase cluster is installed in /usr/local/hbase. The configuration files of HBase are saved in /usr/local/hadoop/conf/. Now let’s go through the configuration file we should edit in /usr/local/hadoop/conf/.

## hbase-env.sh

Set HBase environment variables in this file. Examples include options to pass to the JVM on starting an HBase daemon such as heap size and garbage collector configs. You can also set configurations for HBase configuration, log directories, “niceness”, ssh options, where to locate process pid files, etc [reference http://archive.cloudera.com/cdh/3/hbase/hbase.env.sh.html].

### JAVA\_HOME

### Same with Hadoop, you need to set your JAVA\_HOME in hbase-env.sh

### ../Desktop/Screen%20Shot%202016-04-13%20at%206.34.01%20PM.p

### HBase default Zookeeper

### HBase manages Zookeeper with the setting as follows.

### ../Desktop/Screen%20Shot%202016-04-13%20at%206.34.28%20PM.p

## hbase-site.xml

## HBase specific site customization configuration [reference <https://hbase.apache.org/book.html>]. The file is in XML format, so follow the same edit strategy as before with the hadoop xml configuration files.

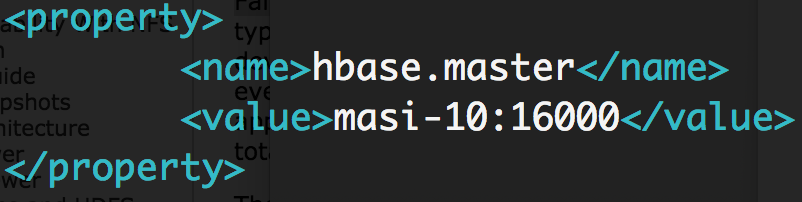
### hbase.rootdir

The directory shared by region servers and into which HBase persists. The HBase stores data in /tmp by default. Many servers are configured to delete the contents of /tmp upon reboot, so you should store the data elsewhere. The URL should specify the HDFS directory '/hbase' where the HDFS instance’s namenode is running at namenode.example.org on port 9000, set this value to: hdfs://namenode.example.org:9000/hbase. The value of hbase.rootdir MUST start with ‘hdfs://’.



### hbase.master

### Defines the hostname/IP address and port that HMaster is to start with.



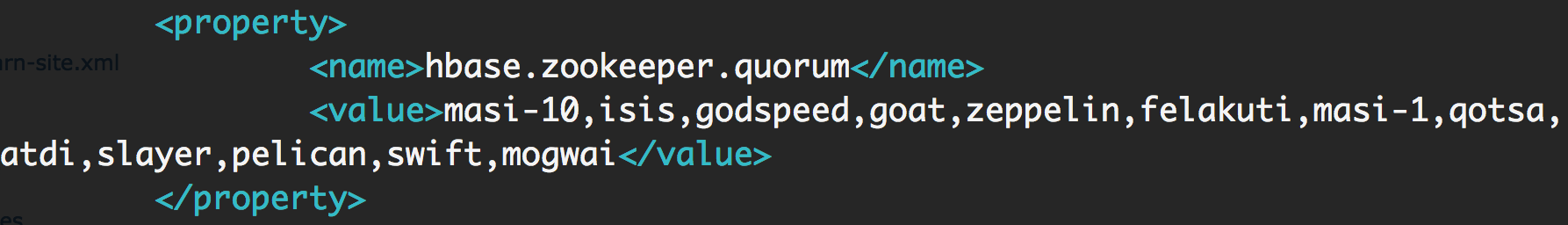
### hbase.cluster.distributed

The mode the cluster will be in. Possible values are false for standalone mode and true for distributed mode. Here we set the value as true for distributed mode.

### ../Desktop/Screen%20Shot%202016-04-13%20at%206.28.53%20PM.p

### hbase.zookeeper.quorum

Comma separated list of servers in the ZooKeeper ensemble. For a fully-distributed setup, this should be set to a full list of ZooKeeper ensemble servers. If HBASE\_MANAGES\_ZK is set in hbase-env.sh this is the list of servers which hbase will start/stop ZooKeeper on as part of cluster start/stop. We need to input all slave and master nodes. The total number of quorums should be an ODD number, since when Zookeeper handles distributed coordination and monitors the health stat in HBase cluster, there must be one leader elected. As follows presents 12 slaves nodes’ hostname including 1 master node which form a 13 node-quorum.



### hbase.zookeeper.property.dataDir

Property from ZooKeeper’s config zoo.cfg (Since we do not use customized Zookeeper). Tis is the directory where a snapshot is stored.

### ../Desktop/Screen%20Shot%202016-04-13%20at%206.30.22%20PM.p

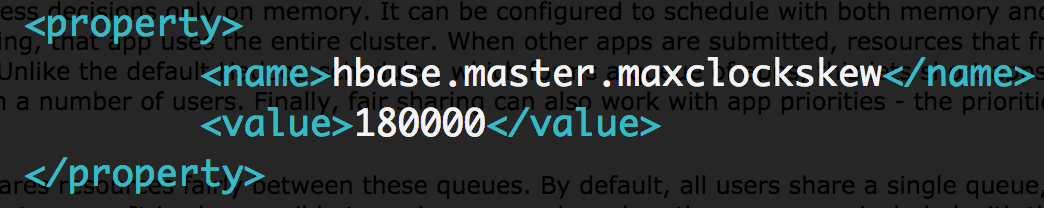
### zookeeper.session.timeout

ZooKeeper session timeout in milliseconds. It is used in two different ways. First, this value is used in the ZK client that HBase uses to connect to the ensemble. It is also used by HBase when it starts a ZK server and it is passed as the 'maxSessionTimeout'. See <http://hadoop.apache.org/zookeeper/docs/current/zookeeperProgrammers.html#ch_zkSessions>. For example, if an HBase region server connects to a ZK ensemble that’s also managed by HBase, then the session timeout will be the one specified by this configuration. But, a region server that connects to an ensemble managed with a different configuration will be subjected to that ensemble’s maxSessionTimeout. So, even though HBase might propose using 90 seconds, the ensemble can have a max timeout lower than this and it will take precedence. The current default that ZK ships with is 40 seconds, which is lower than HBase’s.

### ../Desktop/Screen%20Shot%202016-04-13%20at%206.30.54%20PM.p

### hbase.master.maxclockskew

When a Regionserver's time is ahead of Master's time and the difference is more than hbase.master.maxclockskew value, the region server startup does not fail with ClockOutOfSyncException. It is because of the system time difference among each server. We set this value for endurance time difference.



### hbase.hregion.max.filesize

Maximum HFile size (MB). If the sum of the sizes of a region’s HFiles has grown to exceed this value, the region is split in two. The default size is 1024 MB. We set the value as 512 MB for our experiment with RegionSplitPolicy usage.

### ../Desktop/Screen%20Shot%202016-04-13%20at%206.32.10%20PM.p

### hbase.master.loadbalance.bytable

### Make HMaster balance the regions for each table.

### ../Desktop/Screen%20Shot%202016-04-13%20at%206.32.32%20PM.p

## regionserves

## Similar with Hadoop slaves file, regionserves records HBase regionserver (slave in HBase) nodes’ IP address, hostname/alias, or fully qualified domain name.

# Deploy and install hadoop & hbase on a new server

## Prepare package

### Compress and send a Hadoop package to a new slave server

### Commands:

cd /usr/local

sudo rm -r ./hadoop/logs/\* #For saving space, logs are huge

tar –zcf ~/hadoop.master.tar.gz ./hadoop

cd ~

scp ./hadoop/master.tar.gz Slave\_Hostname:/home/hadoop

### Compress and send HBase package to a new slave server

### Commands:

cd /usr/local

sudo rm r ./hbase/logs/\* #For saving space, logs are huge

tar –zcf ~/hbase.master.tar.gz ./hbase

cd ~

scp ./hbase/master.tar.gz Slave\_Hostname:/home/hadoop

## Setup Hadoop & HBase on new slave server

### Extract Hadoop package to /usr/local

## Commands:

sudo rm –r /usr/local/hadoop # In case there is an old version Hadoop

sudo tar -zxf ~/hadoop.master.tar.gz -C /usr/local

### Extract HBase package to /usr/local

### Commands:

sudo rm –r /usr/local/hbase # In case there is an old version HBase

sudo tar -zxf ~/hbase.master.tar.gz -C /usr/local

### Create folder for Zookeeper

The directory of Zookeeper data is set in Section 6.B.v. We need to manually create the folders hierarchically. Commands:

cd /usr/local

sudo mkdir zookeeper

cd zookeeper

sudo mkdir data

## Ensure that there is an available second drive

## Hopefully there is already a second drive mounted as /drive2

## Run df -h to get a list of mounted filesystems. If there is a device mounted as /drive2, no work needs to be done

## If there is no device mounted as /drive2, note the name of the device mounted as /. Then, run ls /dev/sd\* which lists all hard drives currently accessible.

## If there is a second hard drive visible (note: the disk mounted as / could be e.g. /dev/sda but there could be listed here /dev/sda1 in addition. This is not a second hard drive but rather a partition of the same drive mapped to /.), we can try to mount it as /drive2. First run ls /dev/sd\* AGAIN to make sure this disk is not already being used. Let’s say this unused second disk is listed as /dev/sdb. If you are completely sure this drive is not being used and contains no necessary information on it, the following command would reformat it: sudo mkfs.ext4 /dev/sdb . Now we can run sudo mount /dev/sdb /drive2 and we are done.

## If there is no second hard drive visible, you will need to open up the computer’s case to physically check if there is a hard drive in the computer.

## If no second hard drive is found, insert an unused hard drive and follow the instructions from 2a. In the case that there is still no second hard drive detected, you will need to open the BIOS of the computer and make the drive visible.

## If a second hard drive is found, try enabling it in the BIOS. If it is still not detected by ls /dev/sd\* (or you can’t find it in the BIOS), there is probably a broken hardware component between the disk (most likely), the SATA cable, and the SATA port. Try replacing the disk first.

## Grant permission for user ‘hadoop’ with related folders

According to Section 5.C.i.1, the new slave server uses /drive2 as temporary data storage directory. If the new server does not have /drive2, please create a new directory /drive2 by command:

sudo mkdir /drive2.

Then you need to grant the ‘hadoop’ account permission to edit folders as follows :

#### 

sudo chown hadoop:hadoop /drive2

sudo chown hadoop:hadoop /usr/local/hadoop

sudo chown hadoop:hadoop /usr/local/hbase

sudo chown hadoop:hadoop /usr/local/zookeeper

# Setup yarn configruation

As Section 5.C.iii.1 says, Hadoop v2.0+ uses yarn for MapReduce job scheduling and resource management. The yarn configuration is set individually. Here we present how to edit /usr/local/hadoop/etc/hadoop/yarn-site.xml on a newly created slave server node after we login to the new server with the account ‘hadoop’ [reference <http://www.cloudera.com/documentation/enterprise/5-3-x/topics/cdh_ig_yarn_cluster_deploy.html>, https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-common/yarn-default.xml].

## yarn-site.xml

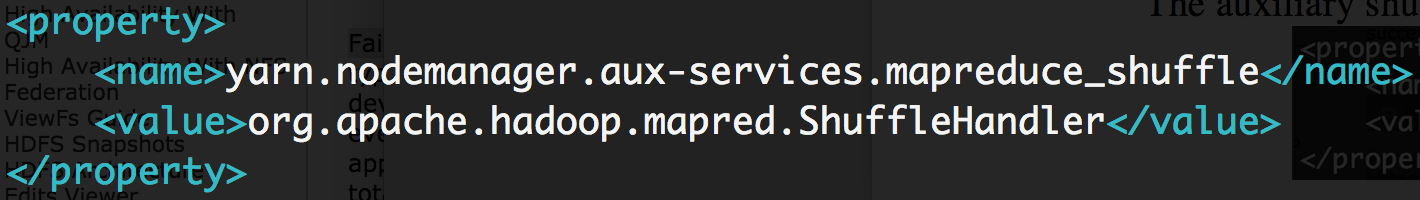
### yarn.nodemanager.aux-services

Shuffle service that needs to be set for Map Reduce applications. A comma separated list of services where service name should only contain a-z, A-Z, 0-9, or \_ and can not start with numbers.

## ../Desktop/Screen%20Shot%202016-04-13%20at%206.27.04%20PM.p

### yarn.nodemanager.aux-services.mapreduce\_shuffule

The auxiliary shuffle service class to use.



### yarn.resourcemanager.resource-tracker.address

## The resource manager resource tracker address, the tracker is the Master node. e.g. ‘masi-10’.

## ../Desktop/Screen%20Shot%202016-04-13%20at%206.22.55%20PM.p

### yarn.resourcemanager.scheduler.address

The address of the scheduler interface. It is located in the Master node. e.g. ‘masi-10’.

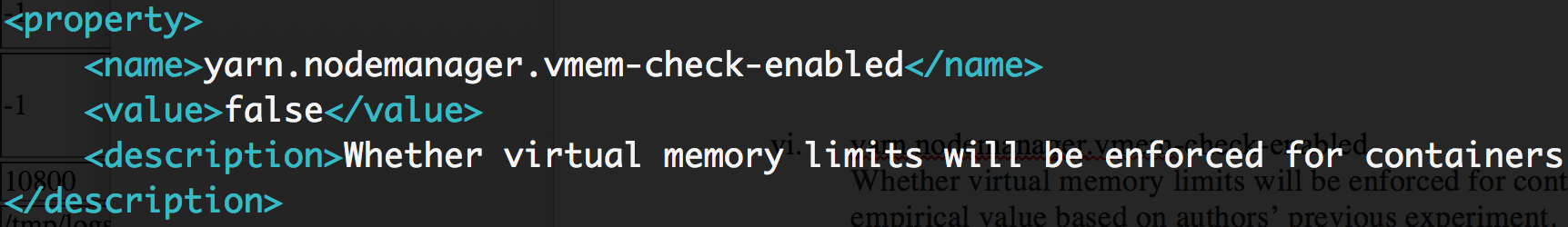


### yarn.resoucemanager.address The address of the applications manager interface in the resource manager.

## ../Desktop/Screen%20Shot%202016-04-13%20at%206.21.51%20PM.p

### yarn.nodemanager.vmem-check-enabled

Determines whether virtual memory limits will be enforced for containers. The value ‘false’ is an empirical value based on the authors’ previous experiments.



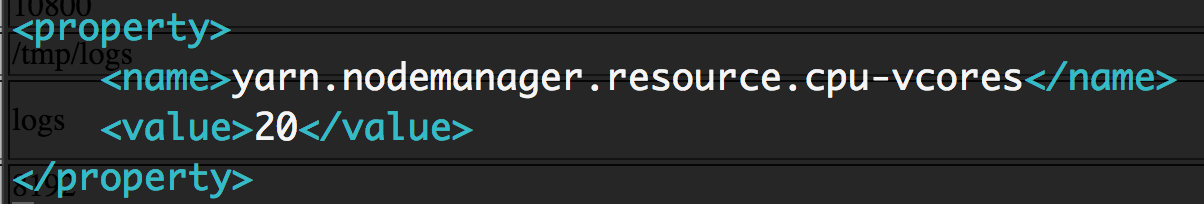
### yarn.nodemanager.vmem-pmem-ratio

Ratio between virtual memory to physical memory when setting memory limits for containers. Container allocations are expressed in terms of physical memory, and virtual memory usage is allowed to exceed this allocation by this ratio. The value ‘4’ is an empirical value based on the authors’ previous experiments.

### ../Desktop/37%20vmem%20ratio.png

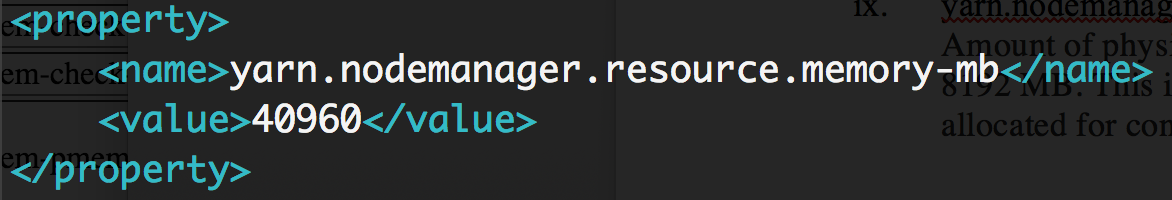
### yarn.nodemanager.resource.cpu-vcores

Number of v(irtual) cores that can be allocated for containers. This is used by the Resource Manager scheduler when allocating resources for containers. This is not used to limit the number of physical cores used by YARN containers. This is a **CUSTOMIZED** value based on the available cpu of a server that can be allocated for containers. The default number is 8.



### yarn.nodemanager.resource.memory-mb

Amount of physical memory, in MB, that can be allocated for containers. The default value is 8192 MB. This is a **CUSTOMIZED** value based on available memory of a server that can be allocated for containers.

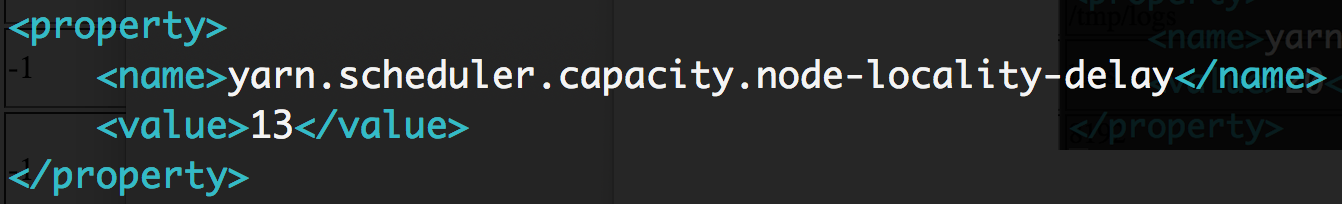


***note:*** *yarn* ***CANNOT*** *automatically get the real cpu numbers and total memory of a machine. As a result the user needs to set it up manually, or else yarn will use the default value of 8 cpu per 8192 MB of memory. Please make sure the previous ratio is suitable, or else YARN could allocate more or less map jobs relative to the maximum threshold of vcores. The number of map jobs is not defined by vcores’s value.*

### yarn.scheduler.capacity.node-locality-delay

There are two default schedulers that are provided by yarn. The default one is CapacityScheduler [ reference http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html]. The CapacityScheduler is designed to run Hadoop applications as a shared, multi-tenant cluster in an operator-friendly manner while maximizing the throughput and the utilization of the cluster.

The value here is the number of missed scheduling opportunities after which the CapacityScheduler attempts to schedule rack-local containers. Typically, this should be set to the number of nodes in the cluster. By default, it is approximately the number of nodes in one rack, which is 40. A positive integer value is expected. We set this value to allow for a higher ratio of data-local map tasks.



### YARN FairScheduler

Another default scheduler that is provided by yarn is FairScheduler. Fair scheduling is a method of assigning resources to applications such that all apps get, on average, an equal share of resources over time [reference https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/FairScheduler.html]. The authors have commented/removed the related properties with FairScheduler temporarily for further research use.



## yarn-env.sh

### Hadoop does not support IPv6, although YARN does not have such a limitation. If we do not disable IPv6 in YARN, in traditional MapReduce jobs (such as Word Count), the jobs are dispatched based on a node’s short hostname, i.e. swift, pelican, slayer etc. However, for HBase in conjunction with MapReduce, it needs to cooperate with RegionServer where data is located. In short, YARN communicates with RegionServers with fully qualified domain name, i.e. swift.ds.vanderbilt.edu, pelican.ds.vanderbilt.edu, slayer.ds.vanderbilt.edu etc. Thus, because of unmatched values between the fully qualified domain names and short hostname aliases, yarn fails to dispatch as many data-local maps ( dispatch computation code in mapper phase to the nodes where needed data locates) as it should. Altogether, we need to disable YARN’s IPv6 option in yarn-env.sh in order to improve MapReduce efficiency.

../Desktop/36%20yarn-env.png

# Start hadoop, hbase and yarn service

## Start Hadoop Distributed File System

ssh to master node as ‘hadoop’ user -> cd $HADOOP\_HOME-> sbin/start-dfs.sh

## Start Distributed HBase cluster

ssh to master node as ‘hadoop’ user -> cd $HBASE\_HOME-> sbin/start-hbase.sh

## Start YARN

ssh to master node as ‘hadoop’ user -> cd $HADOOP\_HOME-> sbin/start-yarn.sh

## Check if the service starts/stops

Use the command ‘jps’ to check if the Hadoop, HBase and YARN services are running

### For a Slave node, 4 jps should exist:

#### DataNode (HDFS data store, act as Hadoop slave should only exist on slave )

#### HRegionServer (HBase regionserver, act as slave in HBase cluster),

#### HQurorumPeer (Zookeeper cluster member),

#### NodeManager (YARN slave),.

### For a master node, 5 jps should exist:

#### NameNode (Hadoop master)

#### SecondaryNameNode (Hadoop master backup)

#### HMaster (HBase master)

#### HQurorumPeer (Zookeeper cluster member)

#### ResourceManager (YARN master)

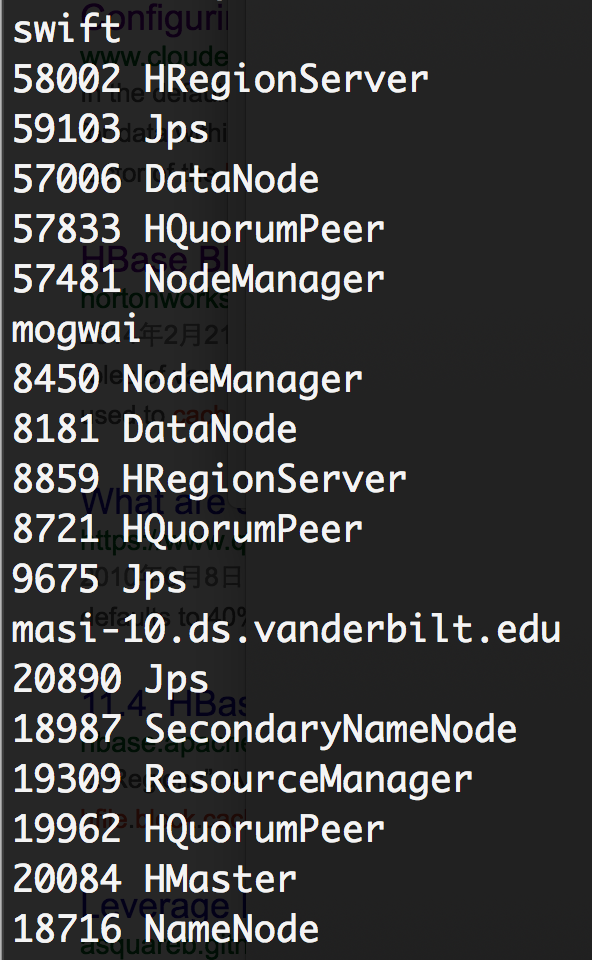


Fig. swift and mogwai are two sample slaves in the cluster (Hadoop, HBase). masi-10 is master node.

# Maintain hadoop, hbase cluster

## Web UI

### For Hadoop & HBase Master instances which are running on Linux with a user interface, you can use a Web browser to observe and monitor the status.

### Hadoop HDFS

### You can check the hdfs status in a web browser with MASTER\_IP:50070, i.e. 10.20.201.46 is masi-10’s IP address.

#### Hadoop Cluster Overview with ‘Overview’ tab

#### screenshot/overview.png

#### Datanode status with ‘Datanodes’ tab

#### screenshot/datanode.png

#### Browse HDFS directory with ‘Utilities’ tab -> ‘Browse the file system’ option

#### screenshot/browse%20Dirctory.png

### HBase cluster

### For HBase, the default port for the web UI is 160010, so you can use the link MASTER\_IP:160010/master-status to monitor the HBase cluster.

#### HBase Master overview

#### Including total regionserver status, tables in current HBase with their design scheme and region split policy, HBase version attributes etc.

#### screenshot/master.png

#### Fig. 11 live regionservers with number of Regions on each server

#### screenshot/tables.png

#### Fig. Two tables exist in current cluster

#### Table detail

#### Upon clicking the link of each table, you can observe all regions of a table, which regionserver serves the table, the start/end row key of a region, table locality (1.0 means region’s data are colocated with the datanode and saved locally) etc.

#### screenshot/newCQS.png

### YARN

Port 8088 is delegated to the YARN service. You can use MASTER\_IP:8088/cluster to observe your YARN nodes’ liveliness, and monitor your MapReduce applications’ statuses (accepted, running, finished, failed and killed etc).

#### screenshot/yarn.png

## Bash command

## If you prefer using the command line to monitor and observe the cluster, no worries. Here are some interesting and useful commands that can help you do this.

### Hadoop HDFS

### On the command line, login to the master node and type the command

### ***hdfs dfsadmin -report***

### to check the Hadoop cluster status. If the ‘hdfs’ command is not in your system environment variables, you can either add it or go into the directory /usr/local/hadoop (where Hadoop is installed) and run the command

### ***bin/hdfs dfsadmin -report***

## ../Desktop/Screen%20Shot%202016-04-13%20at%207.04.02%20PM.p

## Fig. Total Hadoop cluster status

## ../Desktop/Screen%20Shot%202016-04-13%20at%207.03.28%20PM.p

## Fig. One of slave Datanode status

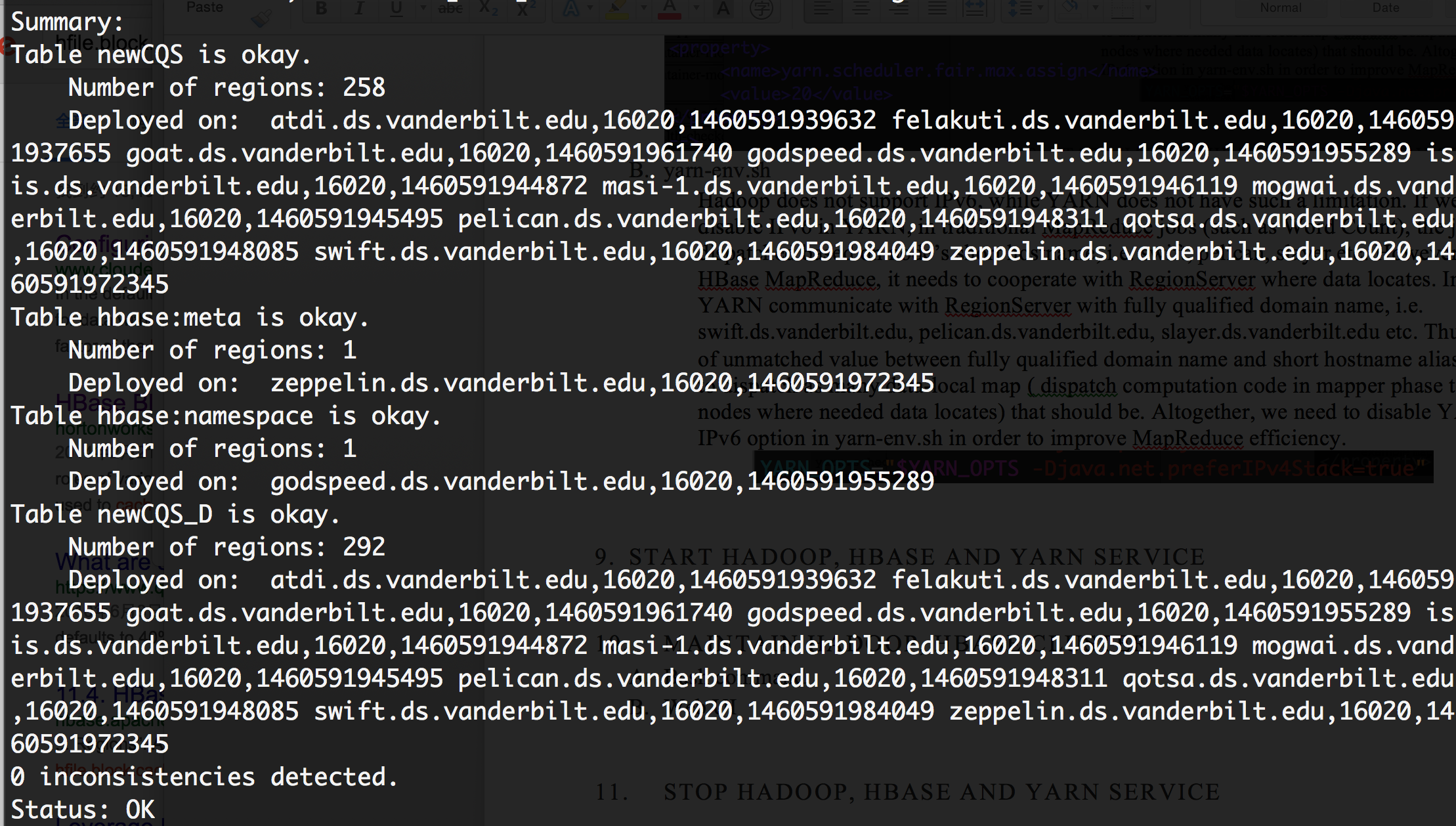
### HBase cluster

#### Cluster regions consistency

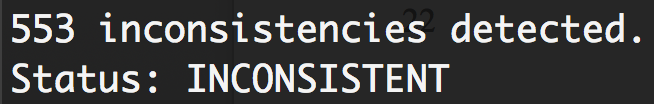
### HBase table regions move within the cluster before being localized. So checking the consistency of region is very important in order to let the user retrieve and write data to the right place. To start, go into the directory /usr/local/hbase and run

### ***bin/hbase hbck***

to check HBase cluster consistency status. The figure as follows shows 4 tables that are stored in HBase. Two of them (hbase:meta and hbase:namespace) are system tables, and the other two (newCQS and newCQS\_D) are custom table. We can see the total number of regions of each table, and check the table’s distribution. ‘0 inconsistencies detected’ indicates that the cluster status is ‘OK’.

******

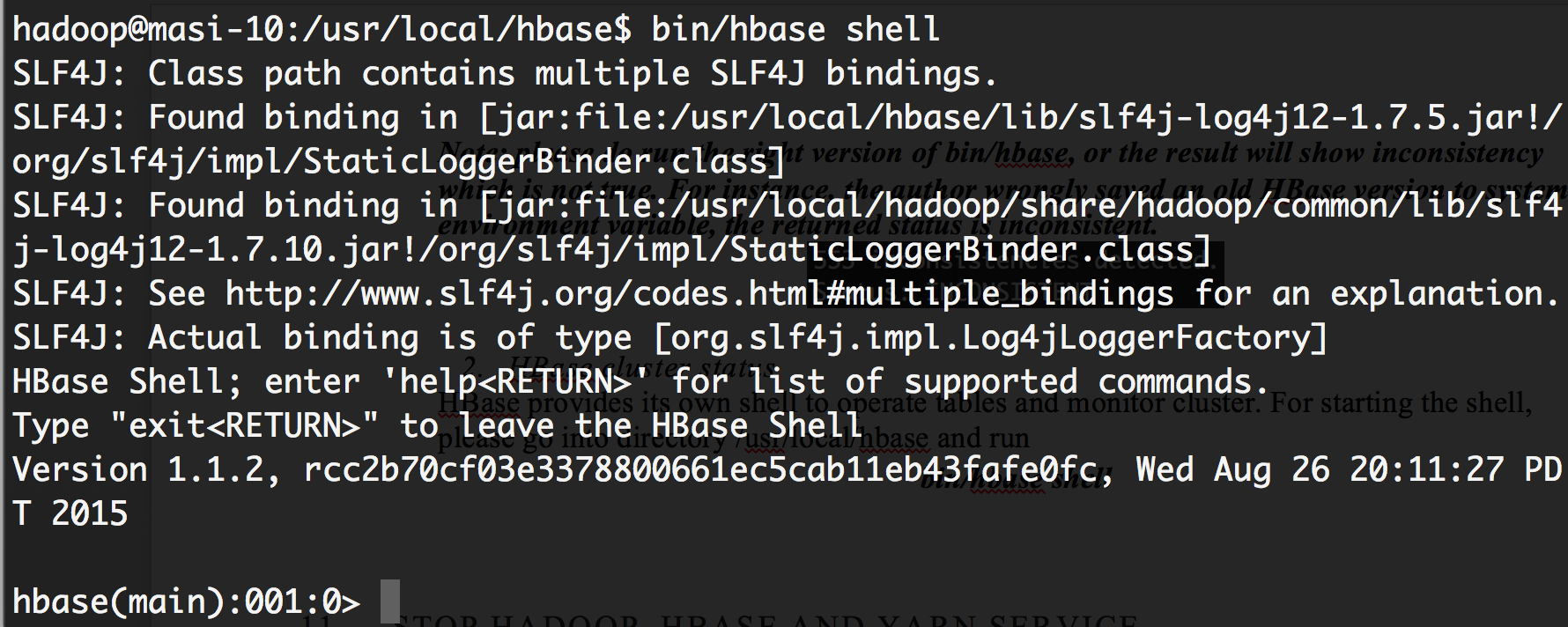
***Note: make sure you run the right version of bin/hbase, or the result yield false inconsistencies. For instance, here the author wrongly saved an old HBase version to the system environment variable, so the returned status is inconsistent.***

******

#### HBase cluster status

HBase provides its own shell to operate tables and monitor the cluster. For starting the shell, go into the directory /usr/local/hbase and run

### ***bin/hbase shell***



Three status checking commands exist, with different usages

##### status

##### This command returns RegionServer status and the average load of the HBase cluster

##### ../Desktop/Screen%20Shot%202016-04-13%20at%207.05.47%20PM.p

##### status ‘simple’

##### This command returns detailed information for each RegionServer. The figure as follows show the servers isis and mogwai’s statuses.

##### ../Desktop/Screen%20Shot%202016-04-13%20at%207.06.06%20PM.p

##### status ‘detailed’

##### Returns details of each table region’s information. The figure as follows list two regions of the table ‘newCQS\_D’.

##### ../Desktop/Screen%20Shot%202016-04-13%20at%207.06.33%20PM.p

### YARN

Login to the master node as ‘hadoop’ account, and go into directory /usr/local/hadoop.

#### YARN node liveliness

#### **bin/yarn node -list**

#### ../../../Desktop/Screen%20Shot%202016-04-13%20at%206.59.43%

#### MapReduce Job status

***bin/mapred job -list***

##### ../../../Desktop/Screen%20Shot%202016-04-14%20at%2011.29.31%

# Commission and Decommission Nodes

* 1. After Hadoop and HBase have been installed, adding and removing a node from the cluster is quite simple.
  2. Commissioning Nodes
     1. In /usr/local/hadoop/etc/hadoop on the namenode, add the hostnames of the nodes to be added to the file “includes”
     2. Ensure hdfs-site.xml contains the following property

<property>  
<name>dfs.hosts.include</name>  
<value>/home/hadoop/includes</value>  
<final>true</final>  
</property>

* + 1. Execute the following command on the namenode: hadoop dfsadmin -refreshNodes
    2. Add the hostname of the nodes to be added to the slaves file
    3. ssh to the newly added slave(s) and execute: /usr/local/hadoop/sbin/hadoop-daemon.sh start datanode
    4. If the new datanodes(s) are shown as active on the web UI, the operation was successful
  1. Decommissioning Nodes
     1. In /usr/local/hadoop/etc/hadoop on the namenode, add the hostname of the nodes to be removed to the file “excludes”
     2. Ensure hdfs-site.xml contains the following property

<property>  
<name>dfs.hosts.exclude</name>  
<value>/home/hadoop/excludes</value>  
<final>true</final>  
</property>

* + 1. Execute the following command on the namenode: hadoop dfsadmin -refreshNodes
    2. Remove the hostnames of the nodes to be removed to the slaves file
    3. If the datanodes(s) are shown as decommissioned on the web UI, the operation was successful

# Stop Hadoop Distributed File System

* 1. ssh to master node as ‘hadoop’ user -> cd $HADOOP\_HOME-> sbin/stop-dfs.sh

## Stop Distributed HBase cluster

ssh to master node as ‘hadoop’ user -> cd $HBASE\_HOME-> sbin/stop-hbase.sh

## Stop YARN

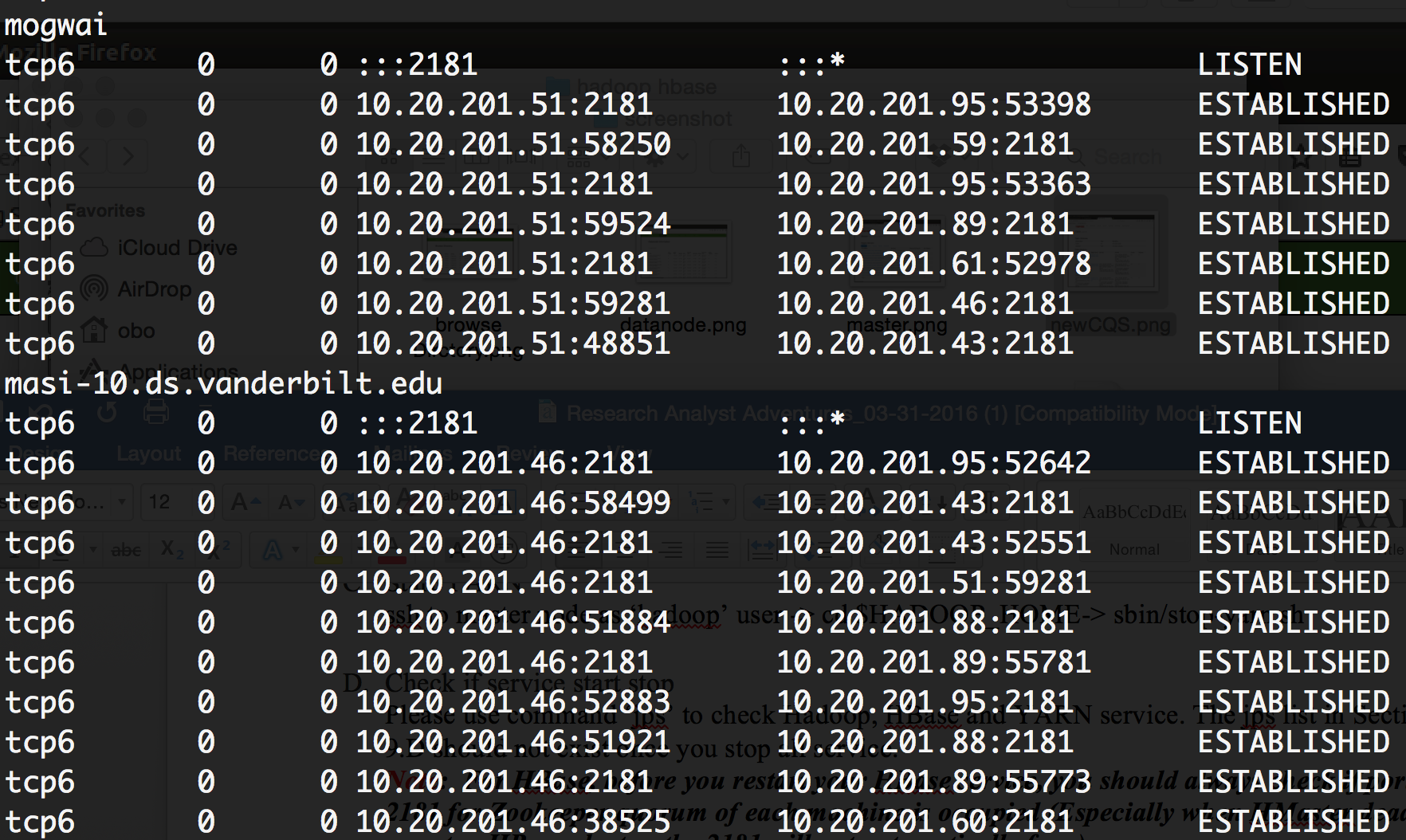
ssh to master node as ‘hadoop’ user -> cd $HADOOP\_HOME-> sbin/stop-yarn.sh

## Check if services have stopped

Use the command ‘jps’ to check to see if Hadoop, HBase and YARN services are active. The jps list in Section 9.D should not exist once you stop all services.

***Note: For HBase, before you restart your HBase service, you should always check if port 2181 for Zookeeper quorum of each machine is occupied (Especially when HMaster is dead and you have stopped the HBase cluster, port 2181 will not automatically be free, as the connection will still be established). Use the following command to check on active ports:***

***netstat –an | grep 2181***



***If results appear as shown above (i.e. port 2181 is still established), you can manually free established connections via the commands***

***sudo fuser –k 2181/tcp***

***sudo fuser –k –n tcp 2181***

***Continue to check if each machine’s port 2181 has been freed. Usually it will take 1-2 minutes.***

# CHECKLIST

## Warning

* DO NOT INCLUDE kyuss, mastadon and masigate as Hadoop cluster node.

## Stop all running hadoop, hbase and yarn service

* Stop Hadoop Distributed File System

*ssh to master node as ‘hadoop’ user -> cd $HADOOP\_HOME-> sbin/stop-dfs.sh*

* Stop Distributed HBase cluster

*ssh to master node as ‘hadoop’ user -> cd $HBASE\_HOME-> sbin/stop-hbase.sh*

* Stop YARN

*ssh to master node as ‘hadoop’ user -> cd $HADOOP\_HOME-> sbin/stop-yarn.sh*

* Check if services have stopped on all slaves
* Check if services have stopped on master
* Check if port 2181 is occupied for any machine. If not, manually free the port as shown in ‘MASI Lab Hadoop HBase cluster setup’ section 12.D

## Create hadoop user

* Login to a new server node as admin ‘masi’, create a new user ‘hadoop’ with password ‘hadoop’
* Grant sudo privileges to user ‘hadoop’
* Logout of the new server as ‘masi’, and login the new server as user ‘hadoop’

## Install vim

* Check if new server has the VIM text editor installed or not
* If not, install VIM

## Edit hosts file.

* Edit /etc/hosts, comment the two lines of private host names for new server

## Setup passphraseless ssh

* Accept all incoming requests via the command ‘sudo iptables –P INPUT ACCEPT’
* Generate a default public/private rsa key pair via the command ‘ssh-keygen’, simply keep pressing Enter throughout the prompts
* Send the public key of the new server to all previously existing cluster nodes.
* Similarly, login to all existing cluster nodes and copy their existing public key to the new server.
* Test if you can ssh as user ‘hadoop’ without password onto the new server, if not, please refer to Section 4.D of documentation ‘MASI Lab Hadoop HBase cluster setup’

## Setup hadoop distributed cluster

* Login to the master node ‘masi-10’ with the ‘hadoop’ account, go into directory /usr/local/hadoop/etc/hadoop
* Edit the slaves file , add a new line and input the new server’s hostname/alias.
* Edit the yarn-site.xml file. Change the value of ‘yarn.scheduler.capacity.node-locality-delay’ to the total number (masters+slaves) of the new cluster

## Setup hbase distributed cluster

* Login to the master node ‘masi-10’ with the ‘hadoop’ account, go into directory /usr/local/hbase/conf
* Edit the hbase-site.xml file. Edit the value of ‘hbase.zookeeper.quorum’ and add the new server’s hostname/alias.
* Edit the regionservers file and add a new line, input the new server’s hostname/alias.

## Deploy and install hadoop & hbase on a new server

* Login to the master node ‘masi-10’ and prepare the compressed Hadoop, HBase package.
* Secure copy (scp) the Hadoop and HBase package to the new server to be added
* Login to the new server with the account ‘hadoop’, extract Hadoop and HBase package to /usr/local
* Create folder for Zookeeper via mkdir ‘/usr/local/zookeeper/’ and mkdir ‘/usr/local/zookeeper/data’
* Check if folder ‘/drive2’ exists, if not create it.
* Grant ‘hadoop’ account permission to edit related folders.

#### 

***sudo chown hadoop:hadoop /drive2***

***sudo chown hadoop:hadoop /usr/local/hadoop***

***sudo chown hadoop:hadoop /usr/local/hbase***

***sudo chown hadoop:hadoop /usr/local/zookeeper***

## Setup yarn configruation

* Login to new server as account ‘hadoop’, go into directory /usr/local/hadoop/etc/hadoop
* Edit the yarn-site.xml file. Edit the value of ‘yarn.nodemanager.resource.cpu-vcores’ to match the new server’s total cpu cores.
* Edit file yarn-site.xml. Edit value of ‘yarn.nodemanager.resource.memory-mb’ according to new server’s total memory can be allocated to YARN.
* Make sure your value setting meets for equation, where you can find value of ‘mapreduce.map.memroy.mb’ in file ‘mapred-site.xml’

## Start hadoop, hbase and yarn service

* Start Hadoop Distributed File System

ssh to master node as ‘hadoop’ user -> cd $HADOOP\_HOME-> sbin/start-dfs.sh

* Start Distributed HBase cluster

ssh to master node as ‘hadoop’ user -> cd $HBASE\_HOME-> sbin/start-hbase.sh

* Start YARN

ssh to master node as ‘hadoop’ user -> cd $HADOOP\_HOME-> sbin/start-yarn.sh

* Check if service start correctly on each slaves.
* Check if service start correctly on each masters.

## Maintain hadoop, hbase cluster

* Login to master node, using bash to check Hadoop HDFS
* Using bash to check HBase cluster
* Using bash to check YARN nodes