|  |
| --- |
| Hadoop Essential Commands or Parameters |
|  |

|  |
| --- |
|  |



© Brain4ce Education Solutions Pvt. Ltd.

# Hadoop Essential Commands or Parameters

Commands:

To have SSH access across machines

1. Generate ssh keys using

ssh-keygen -t rsa -P ""

1. To enable SSH access to your local machine with this newly created key

cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys

1. Distributing ssh public keys to other machines

To add the xxxx@master’s public SSH key (which should be in $HOME/.ssh/id\_rsa.pub)

to the authorized\_keys file of xxxx@slave(in this user’s $HOME/.ssh/authorized\_keys)

xxxx$machine1:ssh-copy-id -i $HOME/.ssh/id\_rsa.pub hduser@machine2

To untar and create dir and to change owner of directory

$ sudo tar -xvf hadoop-1.0.3.tar.gz

$ sudo mv hadoop-1.0.3 hadoop

$ sudo chown -R hduser:hadoop hadoop

Creating Link

$ln -s hadoop-1.0.3 hadoopx

======================================================

Editing config files to setup cluster

config files:(you can use ports 8020 for core and 8021 for mapred also)

core-site.xml

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</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>

mapred-site.xml

<property>

<name>mapred.job.tracker</name>

<value>localhost:9001</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>

hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>1</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>

======================================================

To have namenode or datanode data in specific directories in hdfs-site.xml

<property>

<name>dfs.name.dir</name>

<value>/data/namenode</value>

<final>true</final>

</property>

<property>

<name>dfs.data.dir</name>

<value>/data/datanode</value>

<final>true</final>

</property>

======================================================

For rack awareness:

Update parameter in hdfs-site.xml

<property>

<name>topology.script.file.name</name>

<value>/location/topology.sh</value>

</property>

Create a file in home directory topology.data :

ip address 1(dn1) /rack1

ip address 2(dn2) /rack2

and similarly...or I can give more information like

ip address 1(dn1) /sw1/rack1

ip address 2(dn2) /sw2/rack2

Create a topology script in conf directory

topology.sh :

#!/bin/bash

while [ $# -gt 0 ] ; do

nodeArg=$1

exec< path/topology.data

result=""

while read line ; do

ar=( $line )

if [ "${ar[0]}" = "$nodeArg" ] ; then

result="${ar[1]}"

fi

done

shift

if [ -z "$result" ]; then

echo -n "/default"

else

echo -n "$result"

fi

done

Steps to perform upgrade

Hadoop upgrade:

To save status of hadoop 1 cluster

1. hadoop fsck /

2. hadoop fsck / - files - blocks -locations

3. backing up for comparision later

hadoop fsck / - files - blocks - locations > fsck.bck

4.Backing up list of files recursively

hadoop fs -lsr / > filename.bck

5.Backing up datanodes information to compare later if everything is consistent

hadoop dfsadmin -report > layout.bck

6.Checking if there is any pending upgrade from last change

hadoop dfsadmin -upgradeProgress status

NO uncommitted change from last upgrade, make sure it shows "there is no upgrade in progress"

Also check in metadata and data path for existence of 'previous' directory

if so,then first finalize upgrade and then proceed..

7.Stop daemons.

8.Download new version hadoop package

9.Untar it.

10.Copy all config files from old version to new version directory

cp /usr/local/hadoop/conf/xx /usr/local/hadoopnew/conf/

11.Check if symlink exists,if not create one using

ln -s hadoop hadaj

if exits, unlink it using

unlink hadoop

then create new symlink pointing to new directory

ln -s hadoopnew hadaj

This avoids editing.bash\_profile/.bash\_rc etc

12. Note\*\* no fomatting namenode

13.Start daemon namenode

hadoop-daemon.sh start namenode -upgrade

14. CHECK UPGRADE STATUS

15.START DAEMON DATANODE

16. Finally check for dfsadmin report and safemode status.

17. The cluster can be left in same state or the upgrade has to be finalized.( which removes

all previous enteries)

18. if there is any case of roll back,

hadoop-daemon.sh start namenode -rollback

19. if no roll back,finalize the upgrade hadoop dfsadmin -finalizeUpgrade

after this step, no roll back possible.

======================================================

Checking contents of FSIMAGE

hdfs oiv -i /data/namenode/current/fsiamge - o fsimage.txt

using this viewer,i can view the metadata structure

cat fsiamge.txt

======================================================

To use fairscheduler add to mapred-site.xml

<property>

<name>mapred.jobtracker.taskScheduler</name>

<value>org.apache.hadoop.mapred.FairScheduler</value>

</property>

<property>

<name>mapred.fairscheduler.allocation.file</name>

<value>/usr/local/hadoop/conf/fair-scheduler.xml</value>

</property>

and update in fair-scheduler.xml

<allocations>

<pool name = "adv">

<minMaps>10</minMaps>

<minReduces>5</minReduces>

</pool>

<pool name = "adm">

<minMaps>10</minMaps>

<minReduces>5</minReduces>

<maxRunningJobs>3</maxRunningJobs>

</pool>

<user name = "hduser">

<maxRunningJobs>1</maxRunningJobs>

</user>

</allocations>

to use pool

bin/hadoop jar hadoop-examples-1.2.1.jar wordcount -Dpool.name=fin /books/pg20417.txt /aj/output99

when using fair-scheduler

======================================================

To have namenode data in specific directories and more locations in hdfs-site.xml

<property>

<name>dfs.name.dir</name>

<value>/data/namenode,/nfs/share(or any other directory location</value>

<final>true</final>

</property>

Here second location can be on different disk or nfs or on same disk

=====================================================

To use checkpointing

add

<property>

<name>dfs.name.dir</name>

<value>/data/namenode</value>

<final>true</final>

</property>

<property>

<name>fs.checkpoint.dir</name>

<value>/nfs/share</value>

</property>

======================================================

To commision and decommision nodes add these to hdfs-site.xml

<property>

<name>dfs.hosts.include</name>

<value>/usr/local/hadoop/include</value>

<final>true</final>

</property>

<property>

<name>dfs.hosts.exclude</name>

<value>/usr/local/hadoop/exclude</value>

<final>true</final>

</property>

similarly add this paramter to mapred-site

<property>

<name>mapred.hosts.exclude</name>

<value>/usr/local/hadoop/exclude</value>

<final>true</final>

</property>

<property>

<name>mapred.hosts.include</name>

<value>/usr/local/hadoop/include</value>

<final>true</final>

</property>

and then add respective IPs in include and exclude.

hadoop dfsadmin -refreshNodes

hadoop mradmin -refreshNodes

also update the slaves file accordingly run balancer command to mv HDFS blocks to remaining datanodes.

Start daemons

======================================================

To introduce trash

add in your core-site.xml in namenode

<property>

<name>fs.trash.interval</name>

<value>40</value>

</property>

======================================================

quota

$ hadoop fs -count -q /user/jsmith

none inf 209715200 209715200 5 1 0

The output columns for fs -count -q are: QUOTA, REMAINING\_QUOTA, SPACE\_QUOTA, REMAINING\_SPACE\_QUOTA,

DIR\_COUNT, FILE\_COUNT, CONTENT\_SIZE, FILE\_NAME.

Hadoop checks space quotas during space allocation. This means that HDFS block size (here: 128MB) and

the replication factor of the file

(here: 3, i.e. the default value in the cluster set by the dfs.replication property) play an important role.

In my case, this is what seems

to have happened: When I tried to copy the local file to HDFS,

required\_number\_of\_HDFS blocks \* HDFS\_block\_size \* replication\_count

= 1 \* 128MB \* 3 = 384MB > 200MB.

hadoop fs -D dfs.replication=1 -copyFromLocal small-file.txt /user/jsmith

Now keep in mind that the Hadoop space quota always counts against the raw HDFS disk space consumed. So if you

have a quota of 10MB, you can store only a single 1MB

file if you set its replication to 10. Or you can store up to three 1MB files if their replication is set to 3.

The reason why Hadoop’s quotas work like that is because the replication count of an HDFS file is a

user-configurable setting.

Though Hadoop ships with a default value of 3 it is up to the users to

decide whether they want to keep this value or change it.

And because Hadoop can’t anticipate how

users might be playing around with the replication setting for their files,

it was decided that the Hadoop quotas always operate on the raw HDFS disk space consumed.

Hadoop figured it would require a single block of 128MB size to store the small file.

With replication factored in, the total space would be 3 \* 128MB = 384 MB.

And this would violate the space quota of 200MB