READ-ME

VIKRAM GAVISIDDESHWARA MOUNA GIRI PRANAV SAI DEENUMSETTI

[Year]

1. SHARED MEMORY

* RUN for configuration 1

1. Install JAVA by running the sudo apt-get install openjdk-7-jdk
2. Go to the file location [Inside the directory in which the JAVA files are located]
3. Run the below command to compile the files

Javac \*.java

1. Generate the 128GB DATA using gensort

./64/gensort -a 1374389535 unsorted

1. Then type the below command

java Sort\_Java

Then the below message is displayed.

“Please enter input file name to be sorted”

Enter the generated file name along with the path.

Then this below message is displayed

"Please enter the number of threads [1, 2, 4 or 8] "

Select the threads.

1. Time will displayed when the execution is completed. And the output file is named “output\_sorted”
2. Perform valsort evaluation with the below command

./64/valsort output\_sorted

* RUN for configuration 2

1. Install JAVA by running the sudo apt-get install openjdk-7-jdk
2. Go to the file location [Inside the directory in which the JAVA files are located]
3. Run the below command to compile the files

Javac \*.java

1. Generate the 128GB DATA using gensort

./64/gensort -a 10995116278 unsorted

1. Then type the below command

java Sort\_Java

Then the below message is displayed.

Please enter input file name to be sorted

Enter the generated file name along with the path.

Then this below message is displayed

"Please enter the number of threads [1, 2, 4 or 8] "

Select the threads.

1. Time will displayed when the execution is completed. And the output file is named “output\_sorted”
2. Perform valsort evaluation with the below command

./64/valsort output\_sorted

1. HADOOP

CONFIGURATION 1

//login as root user

$sudo -i

//format the namenode

$hdfs namenode -format

//start dfs services which will start the namenode,datanode and secondary namenode

$start-dfs.sh

//start yarn services which will start the nodemanager and resource manager

$start-yarn.sh

//create the input and ouput directories in hdfs

hdfs dfs -mkdir -p /input

hdfs dfs -mkdir -p /output

//for mounting the volume to root folder

sudo apt-get install mdadm

sudo umount -l /mnt

sudo mdadm --create --force --verbose /dev/md0 --level=0 --

name=MY\_RAID --raid-devices=3 /dev/nvne0n1 /dev/nvne1n1 /dev/xvda

sudo mkfs.ext4 -L MY\_RAID /dev/md0

sudo mkdir -p /mnt/raid

sudo mount LABEL=MY\_RAID /mnt/raid

//create 128GB unsorted data

$/64/./gensort -a 1374389535 ~/Unsorted

//move th unsorted data to hdfs

$ ./hadoop fs -put Unsorted /input/

//remove the unsorted file from local system

rm -r unsorted

//generate the jar from compiled java class files and the then run the jar file using the below command

$ ./hadoop jar HadoopSortSingleNode.jar HadoopSortSingleNode /input Sorted

//put the file back to local to run valsort

./hadoop fs -get Sorted /mnt/raid/

//run valsort

/64/./valsort part-r-00000

CONFIGURATION 2

//login as root user

$sudo -i

//format the namenode

$hdfs namenode -format

//start dfs services which will start the namenode,datanode and secondary namenode

$start-dfs.sh

//start yarn services which will start the nodemanager and resource manager

$start-yarn.sh

//create the input and ouput directories in hdfs

hdfs dfs -mkdir -p /input

hdfs dfs -mkdir -p /output

//for mounting the volume to root folder

sudo apt-get install mdadm

sudo umount -l /mnt

sudo mdadm --create --force --verbose /dev/md0 --level=0 --

name=MY\_RAID --raid-devices=3 /dev/nvne0n1 /dev/nvne1n1 /dev/xvda

sudo mkfs.ext4 -L MY\_RAID /dev/md0

sudo mkdir -p /mnt/raid

sudo mount LABEL=MY\_RAID /mnt/raid

//create 128GB unsorted data

$/64/./gensort -a 10995116278 ~/Unsorted

//move th unsorted data to hdfs

$ ./hadoop fs -put Unsorted /input/

//remove the unsorted file from local system

rm -r unsorted

//generate the jar from compiled java class files and the then run the jar file using the below command

$ ./hadoop jar HadoopSortSingleNode.jar HadoopSortSingleNode /input Sorted

//put the file back to local to run valsort

./hadoop fs -get Sorted /mnt/raid/

//run valsort

/64/./valsort part-r-00000

CONFIGURATION 3 - MULTINODE CLUSTER SETUP

Configure one datanode initially with the configuration listed

below and take a AMI in amazon Ec2 console and launch 7 more

instances and list all those in slaves. This will ease the work of setting up

the cluster.

//generating a passwordless authentication

$ssh-keygen

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

//place the authorized key of a node in every other node including namenode so that all the nodes have ssh passwordless connectivity

$ cat ~/.ssh/id\_rsa.pub | ssh datanode1 'cat >> ~/.ssh/authorized\_keys'

namenode$ cat ~/.ssh/id\_rsa.pub | ssh datanode2 'cat >>~/.ssh/authorized\_keys'

configure the config files as below

Namenode config for multicluster setup

core-site.xml

<property>

<name>fs.defaultFS</name>

<value>hdfs://172.31.34.244:8020</value>

</property>

hdfs-site.xml

<property>

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

<value>file:/root/hadoop/file/namenode</value>

</property>

<property>

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

<value>file:/root/hadoop/file/datanode</value>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

mapred-site.xml

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

yarn-site.xml

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>172.31.34.244:8025</value>

/property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>172.31.34.244:8030</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>172.31.34.244:8050</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

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

</property>

<property>

<name>yarn.nodemanager.disk-health-checker.min-healthy-disks</name>

<value>0</value>

</property>

++++++++++++++++++++++++++++++=

hadoop multinode nodein datanodes

core-site.xml

<property>

<name>fs.defaultFS</name>

<value>hdfs://172.31.34.244:8020</value>

</property>

mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

hdfs-site.xml

<property>

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

<value>file:///root/hadoop/file/namenode</value>

</property>

<property>

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

<value>file:///root/hadoop/file/datanode</value>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

yarn-site.xml

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>172.31.34.244:8025</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>172.31.34.244:8030</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>172.31.34.244:8050</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

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

</property>

<property>

<name>yarn.nodemanager.disk-health-checker.min-healthy-disks</name>

<value>0</value>

</property>

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>

14000

</value>

</property>

<property>

<name>yarn.nodemanager.resource.cpu-vcores</name>

<value>1</value>

</property>

+++++++++++++++++++++++++++++++++++++++

place the ip address of each nodes in each others /etc/hosts folder with their names

vi /etc/hosts

172.31.37.128 datanode7

172.31.41.109 datanode6

172.31.46.154 datanode5

172.31.46.96 datanode4

172.31.43.126 datanode3

172.31.38.150 datanode2

172.31.45.197 datanode1

172.31.34.244 namenode

in namemode add the ipaddress of all datanodes under /hadoop/etc/hadoop/slaves file

Starting the cluster

login to namanode

start-dfs.sh

start-yarn.sh

in namenode

/jps

8176 SecondaryNameNode

8623 Jps

7910 NameNode

8334 ResourceManager

in datanodes

$ jps

1580 NodeManager

1464 DataNode

1724 Jps

//create the input and ouput directories in hdfs

hdfs dfs -mkdir -p /input

hdfs dfs -mkdir -p /output

//for mounting the volume to root folder

sudo apt-get install mdadm

sudo umount -l /mnt

sudo mdadm --create --force --verbose /dev/md0 --level=0 --

name=MY\_RAID --raid-devices=3 /dev/nvne0n1 /dev/nvne1n1 /dev/xvda

sudo mkfs.ext4 -L MY\_RAID /dev/md0

sudo mkdir -p /mnt/raid

sudo mount LABEL=MY\_RAID /mnt/raid

//create 1GB unsorted data

$/64/./gensort -a 1099511627800 ~/Unsorted

//move th unsorted data to hdfs

$ ./hadoop fs -put Unsorted /input/

//remove the unsorted file from local system

rm -r unsorted

//generate the jar from compiled java class files and the then run the jar file using the below command

$ ./hadoop jar HadoopSortSingleNode.jar HadoopSortSingleNode /input Sorted

//put the file back to local to run valsort

./hadoop fs -get Sorted /mnt/raid/

//run valsort

/64/./valsort part-r-00000

1. SPARK

Launch amazon EC2 instance.

Install spark on the instance (Steps are described in the following sections)

Login into the spark cluster

Copy source code to master instance

create input file using gensort using ./gensort -a <lineNumbers> <filePath>

move the file to HDFS using ./hadoop dfs -copyFromLocal <input file path> /input

Run the program,./bin/spark-submit --master spark://MASTER IP:7077 /terasort/sparkterasort.py

Copy file to local from HDFS using ./hadoop dfs -copyToLocal /output <local file path>

Validate output using valsort ./valsort <file name>