CSC 555 project phase 1

Xiaotong He

# Part 1: Set up multi-node cluster (1 master, 2 workers, all 30G)

1) Create 3 nodes, 1 master and 2 workers with the same security group. Then connect to master and set up Hadoop.

2)

download Hadoop and install it:

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/hadoop-2.6.4.tar.gz>

unzip the archive:

tar xzf hadoop-2*.*6.4.tar.gz

install ant to list java process

sudo yum install ant

enter Hadoop:

cd hadoop-2.6.4/

Modify the conf/hadoop-env.sh to add to it the JAVA\_HOME configuration

nano hadoop-2.6.4/etc/hadoop/hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-0.amzn2.0.1.x86\_64/jre

modify the .bashrc file

nano ~/.bashrc

export HADOOP\_HOME=~/hadoop-2.6.4

export PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

source ~/.bashrc

Configure **core-site.xml**, adding the PrivateIP (do not use public IP) of the master

nano hadoop-2.6.4/etc/hadoop/core-site.xml

(private IPs can be found on AWS)

Configure **hdfs-site** and set replication factor to 2

nano hadoop-2.6.4/etc/hadoop/hdfs-site.xml

cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template hadoop-2.6.4/etc/hadoop/mapred-site.xml and then configure **mapred-site.xml**

cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template hadoop-2.6.4/etc/hadoop/mapred-site.xml

nano hadoop-2.6.4/etc/hadoop/mapred-site.xml

Configure **yarn-site.xml** (once again, use PrivateIP of the master)

nano hadoop-2.6.4/etc/hadoop/yarn-site.xml

edit the slaves file and list your 3 nodes (master and 2 workers) using Private IPs

nano hadoop-2.6.4/etc/hadoop/slaves

(remove localhost)

cat hadoop-2.6.4/etc/hadoop/slaves

-------- so far, three nodes set up done! ---------

--------pack up and move Hadoop to the workers ------------

All you need to do is to generate and then copy the public key to the worker nodes to achieve passwordless access across your cluster.

**On master node**:

ssh-keygen -t rsa

(it will generate private and public keys, then copy them to ~/.ssh/authorized\_keys on each woker)

cat .ssh/id\_rsa.pub

(to check the public key)

cat .ssh/id\_rsa

(to check the private key)

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

(add the public key of the master to the master)

On worker:

You now need to manually copy the .ssh/id\_rsa.pub and append it to ~/.ssh/authorized\_keys **on each worker.**

Note that the example below is NOT the master, but one of the workers (ip-172-31-5-246). The first public key is the .pem Amazon half and the 2nd public key is the master’s public key copied in as one line.

A screenshot of a cell phone

Description automatically generated

Back on master:

pack up the entire Hadoop directory into a single file for transfer.

tar cvf myHadoop.tar hadoop-2.6.4

ls -al myHadoop.tar (to verify that the .tar file had been created)

scp myHadoop.tar ip-172-31-21-14:~/myHadoopWorker.tar

(security copy to worker1 which ip is 172-31-21-14)

scp myHadoop.tar ip-172-31-25-34:~/myHadoopWorker.tar

**On worker node:** (add public key of master to woker list)

nano ~/.ssh/authorized\_keys

#ssh 172.31.30.161

tar xvf myHadoopWorker.tar

(untar the hadoop file)

**On master node:**

hadoop namenode -format

start-dfs.sh; start-yarn.sh; mr-jobhistory-daemon.sh start historyserver

jps

hadoop dfsadmin -report

open the master public ip in browser.

<http://ec2-3-87-80-11.compute-1.amazonaws.com:50070/>

datanote

A screenshot of a social media post

Description automatically generated

A screenshot of text

Description automatically generated

--------------Wordcount using bioproject.xml from Assignment 1----------------  
down file to local:

wget [http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/bioproject.xml](http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/psd7003.xml%20)

copy the file to HDFS for preprocessing:

hadoop fs -mkdir /data

hadoop fs -put bioproject.xml /data

hadoop fs -ls /data

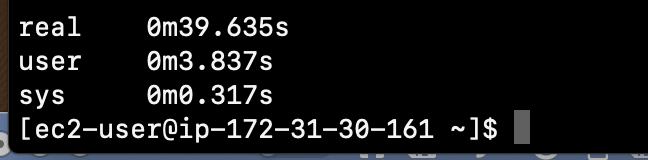
Screen of a cell phone

Description automatically generated

Run word count on the downloaded text file, using the time command to determine the total runtime of the MapReduce job.

time hadoop jar hadoop-2.6.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.4.jar wordcount /data/bioproject.xml /data/wordcount1

( jar command to create .jar file, wordcount is inbuilt, input data is /data/bioproject.xml output file is /data/wordcount1)



hadoop fs -du /data/wordcount1/

A picture containing text, bottle

Description automatically generated

hadoop fs -cat /data/wordcount1/part-r-00000 | grep arctic

(check output)

Answer: Compared to the output of hw1: 1m45.280s, running on 3 node(0m39.635s) is faster than running on 1 node.

A black sign with white text

Description automatically generated

-------------------------------------Part 1 done--------------------------------------------------

# Part 2: Hive

-----------------download dataset-------------------------------

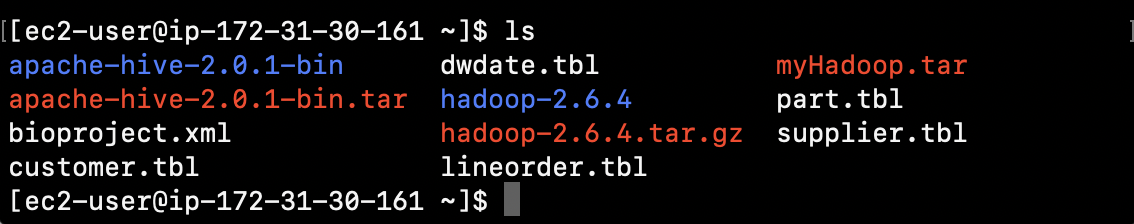
wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/dwdate.tbl>

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/lineorder.tbl>

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/part.tbl>

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/supplier.tbl>

wget <http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/customer.tbl>



----------------------------Install hive-------------------------

wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/apache-hive-2.0.1-bin.tar.gz

gunzip apache-hive-2.0.1-bin.tar.gz

tar xvf apache-hive-2.0.1-bin.tar

set the environment variables:

nano ~/.bashrc

source ~/.bashrc

export HIVE\_HOME=/home/ec2-user/apache-hive-2.0.1-bin

export PATH=$HIVE\_HOME/bin:$PATH

export HADOOP\_HOME=~/hadoop-2.6.4

export PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

$HADOOP\_HOME/bin/hadoop fs -mkdir /tmp

$HADOOP\_HOME/bin/hadoop fs -mkdir /user/hive/warehouse

(if you get an error here, it means that /user/hive does not exist yet. Fix that by running $HADOOP\_HOME/bin/hadoop fs -mkdir -p /user/hive/warehouse instead**)**

$HADOOP\_HOME/bin/hadoop fs -chmod g+w /tmp

$HADOOP\_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

Cd $HIVE\_HOME

rm -rf metastore\_db/

$HIVE\_HOME/bin/schematool -initSchema -dbType derby

bin/hive

create table:

create table lineorder (

lo\_orderkey int,

lo\_linenumber int,

lo\_custkey int,

lo\_partkey int,

lo\_suppkey int,

lo\_orderdate int,

lo\_orderpriority varchar(15),

lo\_shippriority varchar(1),

lo\_quantity int,

lo\_extendedprice int,

lo\_ordertotalprice int,

lo\_discount int,

lo\_revenue int,

lo\_supplycost int,

lo\_tax int,

lo\_commitdate int,

lo\_shipmode varchar(10)

)

Row format delimited fields

Terminated by ‘|’;

create table dwdate (

d\_datekey int,

d\_date varchar(19),

d\_dayofweek varchar(10),

d\_month varchar(10),

d\_year int,

d\_yearmonthnum int,

d\_yearmonth varchar(8),

d\_daynuminweek int,

d\_daynuminmonth int,

d\_daynuminyear int,

d\_monthnuminyear int,

d\_weeknuminyear int,

d\_sellingseason varchar(13),

d\_lastdayinweekfl varchar(1),

d\_lastdayinmonthfl varchar(1),

d\_holidayfl varchar(1),

d\_weekdayfl varchar(1)

)

Row format delimited fields

Terminated by ‘|’;

create table part (

p\_partkey int,

p\_name varchar(22),

p\_mfgr varchar(6),

p\_category varchar(7),

p\_brand1 varchar(9),

p\_color varchar(11),

p\_type varchar(25),

p\_size int,

p\_container varchar(10)

)

Row format delimited fields

Terminated by ‘|’;

create table supplier (

s\_suppkey int,

s\_name varchar(25),

s\_address varchar(25),

s\_city varchar(10),

s\_nation varchar(15),

s\_region varchar(12),

s\_phone varchar(15)

)

Row format delimited fields

Terminated by ‘|’;

create table customer (

c\_custkey int,

c\_name varchar(25),

c\_address varchar(25),

c\_city varchar(10),

c\_nation varchar(15),

c\_region varchar(12),

c\_phone varchar(15),

c\_mktsegment varchar(10)

)

Row format delimited fields

Terminated by ‘|’;

load the data from local FS:

**LOAD DATA LOCAL INPATH '/home/ec2-user/lineorder.tbl' OVERWRITE INTO TABLE lineorder;**

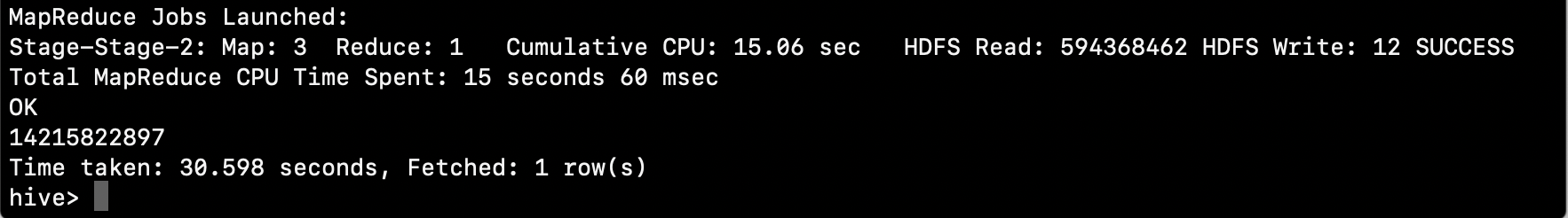
**LOAD DATA LOCAL INPATH '/home/ec2-user/dwdate.tbl' OVERWRITE INTO TABLE dwdate;**

**LOAD DATA LOCAL INPATH '/home/ec2-user/customer.tbl' OVERWRITE INTO TABLE customer;**

**LOAD DATA LOCAL INPATH '/home/ec2-user/part.tbl' OVERWRITE INTO TABLE part;**

**LOAD DATA LOCAL INPATH '/home/ec2-user/supplier.tbl' OVERWRITE INTO TABLE supplier;**

1.2



1.3



2.1



Test python file:

cat customer.tbl |python city\_trans.py

Python code:

import sys

for line in sys.stdin:

line = line.strip()

vals = line.split('\t')

custkey= vals[0]

address=vals[2]

if len(address) >6:

address=address[:6]

city=vals[3][:-1]+ " #" + vals[3][-1]

new=[custkey,address,city]

print '\t'.join(new)

in hive:

hive> ADD FILE /home/ec2-user/city.py;

create table customerswap:

create table customerswap (

c\_custkey int,

c\_address varchar(25),

c\_city varchar(10)

)

Row format delimited fields

Terminated by ‘\t’;

INSERT OVERWRITE TABLE customerswap

SELECT TRANSFORM (c\_custkey,c\_address,c\_city) USING ‘python city.py’ AS (c\_custkey,c\_address,c\_city) FROM customer;

INSERT OVERWRITE TABLE customerswap

SELECT TRANSFORM (c\_custkey,c\_name,c\_address,c\_city,c\_nation,c\_region,c\_phone,c\_mktsegment) USING ‘python city.py’ AS (c\_custkey,c\_address,c\_city) FROM customer;

A close up of a logo

Description automatically generated

---------------------------------Part 2 done-----------------------------------------

# Part 3: Pig

For PIG, you need to upload the data on HDFS first.

Install pig:

cd

wget http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/pig-0.15.0.tar.gz

gunzip pig-0.15.0.tar.gz

tar xvf pig-0.15.0.tar

set environment:

nano ~/.bashrc

source ~/.bashrc

export PIG\_HOME=/home/ec2-user/pig-0.15.0

export PATH=$PATH:$PIG\_HOME/bin

put lineorder.tbl to HDFS

hadoop fs -put /home/ec2-user/lineorder.tbl

Load data into pig from local:

**lineorder= LOAD '/user/ec2-user/lineorder.tbl' USING PigStorage('|')**

**AS (**

**lo\_orderkey :int,**

**lo\_linenumber :int,**

**lo\_custkey :int,**

**lo\_partkey :int,**

**lo\_suppkey :int,**

**lo\_orderdate :int,**

**lo\_orderpriority :chararray,**

**lo\_shippriority :chararray,**

**lo\_quantity :int,**

**lo\_extendedprice :int,**

**lo\_ordertotalprice :int,**

**lo\_discount :int,**

**lo\_revenue :int,**

**lo\_supplycost :int,**

**lo\_tax :int,**

**lo\_commitdate :int,**

**lo\_shipmode :chararray**

**);**

Check table description:

**DESCRIBE lineorder;**

--Q0.1 Added simple test query

SELECT AVG(lo\_revenue)

FROM lineorder;

**data = GROUP lineorder ALL;**

**avg = FOREACH data GENERATE AVG(lineorder.lo\_revenue);**

**DUMP avg;**

A screenshot of a cell phone

Description automatically generated

--Q0.2 Added simple test query

SELECT lo\_discount, COUNT(lo\_extendedprice)

FROM lineorder

GROUP BY lo\_discount;

**data = GROUP lineorder BY lo\_discount;**

**count\_num = FOREACH data GENERATE lineorder.lo\_discount, COUNT(lineorder.lo\_extendedprice);**

**DUMP count\_num;**

--Q0.3 Added simple test query

SELECT lo\_quantity, SUM(lo\_revenue)

FROM lineorder

WHERE lo\_discount < 3

GROUP BY lo\_quantity;

**Filter = FILTER lineorder BY lo\_discount < 3;**

**DiscountGroup = GROUP Filter BY lo\_quantity;**

**sum\_re = FOREACH DiscountGroup GENERATE Discount.lo\_quantity, SUM(Discount.lo\_revenue);**

**DUMP sum\_re;**

I put all my pig commands into a text file using:

nano pig\_s.pig

then run at $PIG\_HOME:

bin/pig -f pig\_script.pig

A screenshot of a cell phone

Description automatically generated

------------------------part 3 done------------------------------

------------------------part 4-----------------------------------------

# Part 4: Hadoop Streaming

myMapper.py code:

#!/usr/bin/python

import sys

for line in sys.stdin:

line = line.strip()

vals = line.split("|")

lo\_quantity=int(vals[8])

lo\_revenue=int(vals[12])

lo\_discount=int(vals[11])

print "%s\t%s\t%s" % (lo\_quantity, lo\_revenue, lo\_discount)

myReducer.py code:

#!/usr/bin/python

import sys

curr\_group=None

curr\_sum=0

for line in sys.stdin:

line = line.strip()

# parse the line and split it by '\t'

vals = line.split('\t')

lo\_quantity=int(vals[0])

lo\_revenue=int(vals[1])

lo\_discount=int(vals[2])

if curr\_group==lo\_quantity:

if lo\_discount>=3 and lo\_discount<=5:

curr\_sum += lo\_revenue

else:

if curr\_group:

print '%s\t%d' % (curr\_group, curr\_sum)

curr\_group= lo\_quantity

curr\_sum= lo\_revenue

if curr\_group == curr\_group:

print '%s\t%d' % (curr\_group, curr\_sum)

put myMapper.py, myReducer.py, lineorder.tbl to HDFS

hadoop fs -put /home/ec2-user/myMapper.py

hadoop fs -put /home/ec2-user/myReducer.py

hadoop fs -put /home/ec2-user/lineorder.tbl /data/lineorder

hadoop jar hadoop-streaming-2.6.4.jar -input /data/lineorder -output /data/output1 -mapper myMapper.py -file myMapper.py -reducer myReducer.py -file myReducer.py

A screenshot of a cell phone

Description automatically generated