Robert Kaszubski

CSC 555 Project Phase 1

Midterm

**Part 1)**

I set up my cluster:

A screenshot of a computer

Description automatically generated with medium confidence

Running wordcount:

Text

Description automatically generated

Time:

Text

Description automatically generated

Successful output:

A screenshot of a computer

Description automatically generated with medium confidence

Running Grep Arctic as in Assignment 2:

Text

Description automatically generated

Looking back at assignment 2, running wordcount using the single node originally finished in 1 minute and 13.386 seconds. Running it now using the cluster setup, resulted in a run time of 47.3 seconds. So, it was about 26 seconds faster than before. I would have expected it to be even faster given that it was running on a three-node cluster rather than a single node, yet it wasn’t even twice as fast. However, we do have to consider that there were more things that may have slowed down the process a bit such as the way the blocks were spread out of distributed among the nodes. It is also likely that the speed of the network and connecting to different nodes played a part. However, the overall speed is still much faster than it was before.

**Part 2**

**1)**

Building Tables:

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 '|' STORED AS TEXTFILE;

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 '|' STORED AS TEXTFILE;

Import data:

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

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

A screenshot of a computer

Description automatically generated with medium confidence

Running Query:

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

The query took 42.512 seconds to run

**2)**

Python Code:

Text

Description automatically generated

ADD FILE /home/ec2-user/part2b.py

New Table Schema:

create table dwdatenew (

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\_monthnuminyear int,

d\_sellingseason varchar(13),

d\_lastdayinweekfl varchar(1),

d\_holidayfl varchar(1),

d\_weekdayfl varchar(1),

d\_daynuminweekmonthyear varchar(10)

) ROW FORMAT DELIMITED FIELDS

TERMINATED BY '\t' STORED AS TEXTFILE;

COMMAND:

INSERT OVERWRITE TABLE dwdatenew SELECT TRANSFORM (d\_datekey, d\_date, d\_dayofweek, d\_month, d\_year, d\_yearmonthnum, d\_yearmonth, d\_daynuminweek, d\_daynuminmonth, d\_daynuminyear, d\_monthnuminyear, d\_weeknuminyear, d\_sellingseason, d\_lastdayinweekfl, d\_lastdayinmonthfl, d\_holidayfl, d\_weekdayfl) USING 'python part2b.py' AS (d\_datekey, d\_date, d\_dayofweek, d\_month, d\_year, d\_yearmonthnum, d\_yearmonth, d\_monthnuminyear, d\_sellingseason, d\_lastdayinweekfl, d\_holidayfl, d\_weekdayfl, d\_daynuminweekmonthyear) FROM dwdate;

Text

Description automatically generated

Took 14.284 seconds to run

Sample results: new column is added to the end, other columns removed – 13 total columns now vs 17 before

New column is called d\_daynuminweekmonthyear and takes the format of day num in week/month/year.

A picture containing graphical user interface

Description automatically generated

**Part 3**

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);

Testing:

lineorderG = GROUP lineorder ALL;

Count = FOREACH lineorderG GENERATE COUNT(lineorder);

DUMP Count;

Data was loaded and pig works!

A screenshot of a computer screen

Description automatically generated with medium confidence

**Query 1:**

groupDiscount = GROUP lineorder BY lo\_discount;

groupAvg = FOREACH groupDiscount GENERATE group as lo\_discount, AVG(lineorder.lo\_extendedprice);

STORE groupAvg INTO 'Query1' using PigStorage(', ');

A screenshot of a computer

Description automatically generated with medium confidence

Output:

Text

Description automatically generated

Running as Script:

Text

Description automatically generated

Runtime for **Query 1 is 59 seconds and 65 milliseconds**

**Query 2:**

filterDiscount = FILTER lineorder BY lo\_discount > 8;

filterQuantity = FILTER filterDiscount BY lo\_quantity > 33;

groupQuantity = GROUP filterQuantity BY lo\_quantity;

sumRevenue = FOREACH groupQuantity GENERATE group as lo\_quantity, SUM(filterQuantity.lo\_revenue) as revenue;

STORE sumRevenue INTO 'Query2' using PigStorage(',');

A computer screen capture

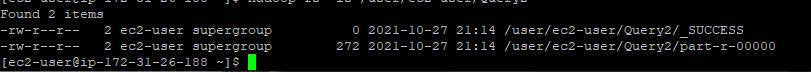
Description automatically generated with medium confidence

Output:

Text

Description automatically generated

File Size:



File size is **272 bytes**

Running as script:

Text

Description automatically generated

Query 2 took **1 minute, 3 seconds, and 834 milliseconds** to run

**Part 4**

SUBQUERY:

We are first trying to implement the subquery:

SELECT lo\_revenue, MAX(lo\_quantity) as lo\_quantity,

MAX(lo\_discount) as lo\_discount

FROM lineorder

WHERE lo\_orderpriority LIKE '%URGENT'

GROUP BY lo\_revenue)

MyMapper1.py:

Text

Description automatically generated

Sample Output of MyMapper1 (cat lineorder.tbl | python MyMapper1)

Text

Description automatically generated

Key is lo\_revenue, Value is lo\_quantity \_ lo\_discount

MyReducer1.py:

Text

Description automatically generated

Sample Output of MyReducer1 (cat lineorder.tbl | python MyMapper1.py | sort -n | python MyReducer1.py):

A picture containing text

Description automatically generated

Key is lo\_revenue, Value is max(lo\_quantity)\_max(lo\_discount)

Now with the subquery complete we can take this data and begin our second MapReduce operation:

Main Query:

MyMapper2.py:

Text

Description automatically generated

Sample Output of MyMapper2 (cat lineorder.tbl | python MyMapper1.py | sort -n | python MyReducer1.py | python MyMapper2.py) :

A picture containing text

Description automatically generated

Key is now Quantity, Value is Revenue

MyReducer2.py:

Text

Description automatically generated

Output: cat lineorder.tbl | python MyMapper1.py | sort -n | python MyReducer1.py | python MyMapper2.py | sort -n | python MyReducer2.py

FINAL OUTPUT:

A picture containing background pattern

Description automatically generated

Key is Quantity, Value is revenue

**Using Hadoop Streaming:**

FIRST MAP REDUCE JOB (ran in cd $HADOOP\_HOME)

hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/lineorder.tbl -output /data/outputSubQuery2 -mapper MyMapper1.py -reducer MyReducer1.py -file ../MyReducer1.py -file ../MyMapper1.py

Text

Description automatically generated

Text

Description automatically generated

First map reduce ran successfully and wrote output to /data/outputSubQuery2

Now using that output to run the second map reduce job

SECOND MAP REDUCE:

hadoop jar hadoop-streaming-2.6.4.jar -input /data/outputSubQuery2 -output /data/outputMainQuery2 -mapper MyMapper2.py -reducer MyReducer2.py -file ../MyReducer2.py -file ../MyMapper2.py

Text

Description automatically generated

It ran successfully and stored the final output to /data/outputMainQuery2



Output:

Text

Description automatically generated

I got the same output as when I tested it using pipes, so the two map reduce jobs ran successfully!