CSE 587
DATA INTENSIVE COMPUTING
SPRING 2015
PROGRAMMING ASSIGNMENT #3 – REPORT

PRATIK PRAMOD CHAVAN
UBIT NAME: PCHAVAN2
UB PERSON #:- 5013-4114
CONTACT NO:- (716)-400-4194
EMAIL ID: pchavan2@buffalo.edu

1. RATIONALE FOR PIG AND HIVE COMPUTATION:

PIG:

- Initially, the local input directory is passed as an argument when submitting the job. The path to this directory can be accessed by $1 in the SLURM script. I fetched the *.csv files from this directory and copied them to hdfs using -put command.
- Then, I loaded files from this directory (/pigdata) to relation stock. In this, I used first column to store the name of the file. I used parameter –tagFile in PigStorage() to extracting the filename.
- Then, I put columns which are required for actual computation viz., filename, date and Adj. close in new relation stock2.
- After this, I created a new relation stock5 in which I split the date which was initially stored as a single string into date, month and year columns. Initially STRSPLIT() output was a single tuple. So, I used flatten to put date, month and year into separate columns in same relation stock5.
- Then, I copied stock5 to stock18. In relation stock121, I basically typecasted date, month and year columns from stock5 to int as they would letter be needed in that format so as to perform aggregate functions on them.
- In stock19 and stock20, I basically computed first and last date of each month.
- Relation stock22 contained all the dates for each file and Adj_close value.
- Relations stock24 and stock26 contained filename along with first and last date for each month and each year respectively.
- Performing an inner join on stock22, stock24 and stock stock22, stock26 resulted in 2 new relations stock25begin and stock25end. These relations basically contained only the filename first and last date of every month along with month, year and corresponding Adj Close price.
- In relations begin and end, I basically combined filename, month and year to one tuple into a single column so to make further group by operations simpler.
- In relation monthly, I computed monthly Rate of Return for each month for each file.
- Then, I performed group by operation using filename so at to compute volatility for individual file.
- Then, I used aggregate functions AVG(), SUM(), COUNT() to calculate xi, summation of (x-xi)*(x-xi) and dividing it by number of months. And then, I used SQRT() to compute volatility. Relations month and mt were used in the intermediate processing.
- Then, I used DISTINCT to filter out duplicates from monthly.
- After that, I ordered monthly by volatility to compute maximum and minimum values of volatilities. For maximum values, I used order by desc.
- Then, I used Rank to compute each column number as we want to have top 10 results as final result.
- Finally, I put maximum 10 results in biggest directory in hw3_out and minimum 10 results in smallest directory in hw3_out.
- These directories are at the end copied into the local file system at argument given ($2).

HIVE:

- Initially, I loaded the data in table stockdata in a similar way as in Pig (Copy from local to hdfs and then using location on hdfs).
- Then, I stored filename, date, month, year and Adj_close in table s. I stored date, month and year as int for future processing. I extracted filename using INPUT__FILE__NAME.
- In s1 and s2, I calculated minimum and maximum date for each month using froup by filename, month and year.
- In s3 and s4, I performed inner join on s2,s and s1, s on columns filename, month and year to form tables having Adj_close of first and last date of every month.
- Then, tables s5, s6, s7 and s8 are used to for computing intermediate values. Aggregate functions like AVG(), SUM(), COUNT() were used. Finally, suing SQRT(), I stored volatilities for each file in s8.
- Then, I ordered it by volatility to sort it ascending and descending. I stored these tables in s9 and s10.
- In s11 and s13, I computed rank() of each function to get column number.
- I used s14 and s12 to computed, top 10 maximum and minimum results.
- Finally, I used select on s12 and s14 to return required results which are stored in log file.

2. EXPERIMENTS:

We are required to find the computing times for the 3 types of data sets (small, medium and large) each on 1,2 and 4 nodes (each with 12 cores) at CCR at University at Buffalo.

The objective here is to check the performance of the program and check its data scalability and node scalability.

The following table shows the execution times which I got for all the cases using each of Pig, Hive and MapReduce jobs. The results for MapReduce jobs are taken from the results I obtained in Programming Assignment 1.

PIG:

Problem Size	Execution Time: 1 node (12 cores)	Execution Time: 2 nodes (24 cores)	Execution Time: 4 nodes (48 cores)
Small	1713 (28.55 min)	1267 (21.11 min)	1251 (20.85 min)
Medium	1863 (31.05 min)	1499 (24.98 min)	1486 (24.77 min)
Large	2910 (48.50 min)	2105 (35.08 min)	2100 (35 min)

HIVE:

Problem Size	Execution Time: 1 node (12 cores)	Execution Time: 2 nodes (24 cores)	Execution Time: 4 nodes (48 cores)
Small	1027 (17.11 min)	993 (16.55 min)	966 (16.1 min)
Medium	1397 (23.28 min)	1369 (22.81 min)	1335 (22.25 min)
Large			

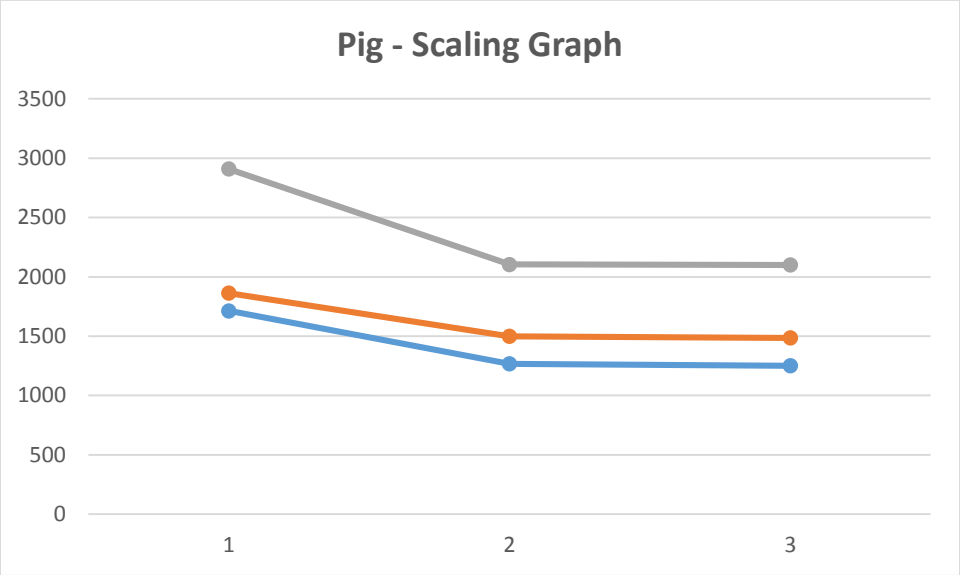
MAPREDUCE:

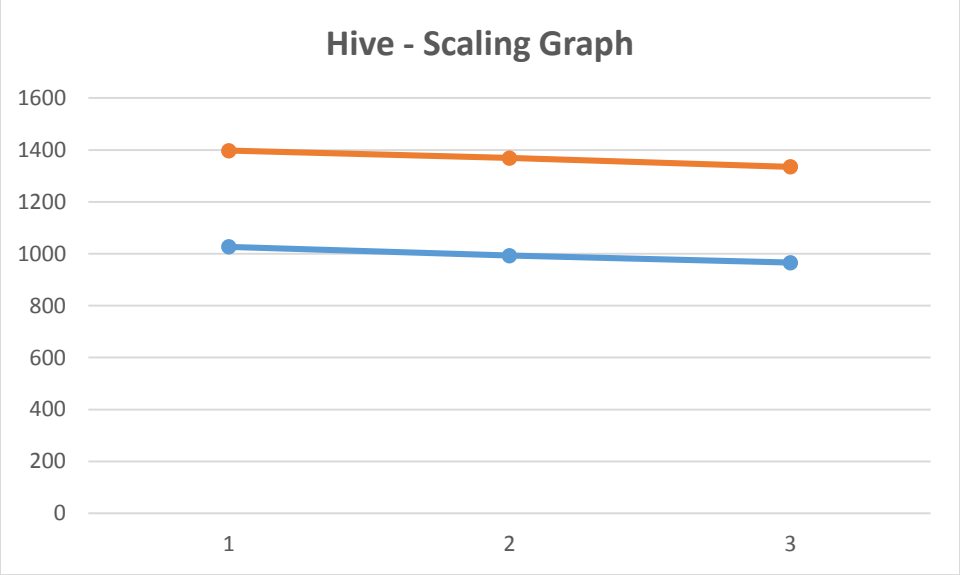
Problem Size	Execution Time: 1 node (12 cores)	Execution Time: 2 nodes (24 cores)	Execution Time: 4 nodes (48 cores)
Small	2400 (40 min)	1085 (18.08 min)	508 (8.47 min)
Medium	7168 (119.47 min)	3245 (54.08 min)	1492 (24.87 min)
Large	23667 (394.45 min)	10661 (177.68 min)	5017 (83.61 min)

3. GRAPHS:

The graphs below are the scaling graph which tell us how the scaling against number of nodes and the scaling against size of dataset is taking place.

X- axis has 3 values which represent number of nodes which are 1, 2 and 4. Blue line represents the small dataset, orange line represents medium dataset and purple line represents large dataset. Y-axis represents time in seconds taken to execute the job.





4. DISCUSSION:

The results I obtained for MapReduce were scaled perfectly for all the cases. However, for Pig and Hive, even though there was some amount scaling as it can be seen from the graphs, it was not similar to MapReduce.

For Pig, execution for all the data sets took almost the same amount of time with 2 and 4 nodes. With 1 node however, it took considerably larger time.

For Hive, the execution time for a particular data set was almost same for 1, 2 and 4 number of cores. There was a minute difference between values. Execution with 1 core took slightly more time than execution with 2 cores. Execution with 2 core took slightly more time than execution with 4 cores. I could not find results for large data set with Hive because of Java Heap Space Error.

COMPARISON OF PIG, HIVE AND MAPREDUCE IN SAME SETTINGS:

Let us now compare the results obtained for all 3 data sets in Pig, Hive and MapReduce by taking average of execution times on 1 core, 2 cores and 4 cores.

Problem Size	Pig	Hive	MapReduce
Small	**1410 (23.50 min)**	**995 (16.58 min)**	**1331 (22.18 min)**
Medium	**1616 (26.93 min)**	**1367 (22.78 min)**	**3968 (66.13 min)**
Large	**2372 (39.52 min)**		**13115 (218.58 min)**

From the table above, we can see that, for small data set Hive performed the best.

Performance of Pig and MapReduce was also similar but slightly worse than Hive. For this data set, we can conclude that performance of all 3 was similar.

For medium data set, performance of Pig and Hive was almost similar. But, Hive yielded better results between the two. Performance of MapReduce however was significantly worse in this case.

For Large data set, I did not have results of Hive. But, if they were to be similar with the hive performance for other 2 data sets, we can conclude that they were much better than MapReduce and similar to those of Pig.

Hive, In general performed slightly better than Pig and both Pig and Hive performed way much better than MapReduce for medium and large data set.

So, we can conclude that it is feasible to use MapReduce for smaller data sets but since performance of Pig and Hive is not directly proportional to data size, it is better to use Pig and Hive for significantly larger data sets.