Data Intensive Computing Assignment 4 CSE 487/587

-*Utkarsh Srivastava*

Introduction :

Stock Volatility is a measure of how stable a stock is in real time performance.It is calculated by calculating the rate and amount of changes its price undergoes in a given period of time and is dependent on the sstandard deviation.
In the activity, We are given the following paramteres from which volatility V needs to be computed :

Date represents the date of the stock ;
Open represents the open price in that day of stock ;
High represents the highest price in that day of stock ;
Low represents the lowest price in that day of stock ;
Adj Close represents the close price in that day of stock ;
Volume represents the volume in that day of stock ;
We compute the volatility of 2970 stocks of various companies , analyze and compare them and finally display the 10 most and least volatile stocks. The performance and running time on 12,24 and 48 core nodes is also discussed .

Procedure :

The procedure to compute the stock prices can be divided into 3 phases of map and reduce. We instantiate three jobs and set their input paths and intermittent output file. We transfer the progress within each phase of map and reduce by using a context object to write the arguments key & value

First Phase :

In the first phase , As part of mapping, all parameters are extracted from the value argument. The date and closing balance are then filted and stored in keys and value pairs in the format : stockname_yyyy_mon and day_stockprice.During the first phase of reduce , We inherit the stock price with day under one entires and thus can compute x_last day with x_small day . We find xi by the following formula :
xi = Monthly Rate of Return = (Month end adjusted close price – Month beginning adjusted close price) / (Monthly beginning adjusted close price)

Second phase :

In the Mapping part of second phase, We extract the stock name and the prices from the values and ignore the mm and the yyyy. We thus map the stock name with the xi.
During reduction, we compute the volatility for each phase. Invalid values are discarded here , like 0 and N=1

Third Phase:

This is the final phase. Extract the values of volatility and the stock price and send it to the Final reducer , which will compute the 10 highest and lowest volatile stocks (through a hash map ,sorted by its values) and print the final output

The primary motive behind using Hbase is random and realtime read/write access to Big Data. It hosts very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware. Apache HBase is an open-source, distributed, versioned, non-relational database modeled after Google's Big Table. Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS.
Here We can use the standalone mode for Hbase locally as it is not an appropriate configuration for a production instance of HBase, but will allow you to experiment with Hbase. The version ised in 0.98.
The general program flow is as follows :
We first

Performance Improvement in Hbase:

If we compare the results with those of hbase, We realise that Hbase is lacking considerably compared to pig and hive. The most probable reason for the same is that hbase was not designed to perform row based operations and instead concentrates upon the field based operations . Therefore, we rely severely to perform aggregate functions on map-reduce program from java. Also, it takes almost the entire aspect of hbase performance on reading and writing values in its table. Therefore we can govern our thoroughput primarily from the speed ot storage and retrieval.

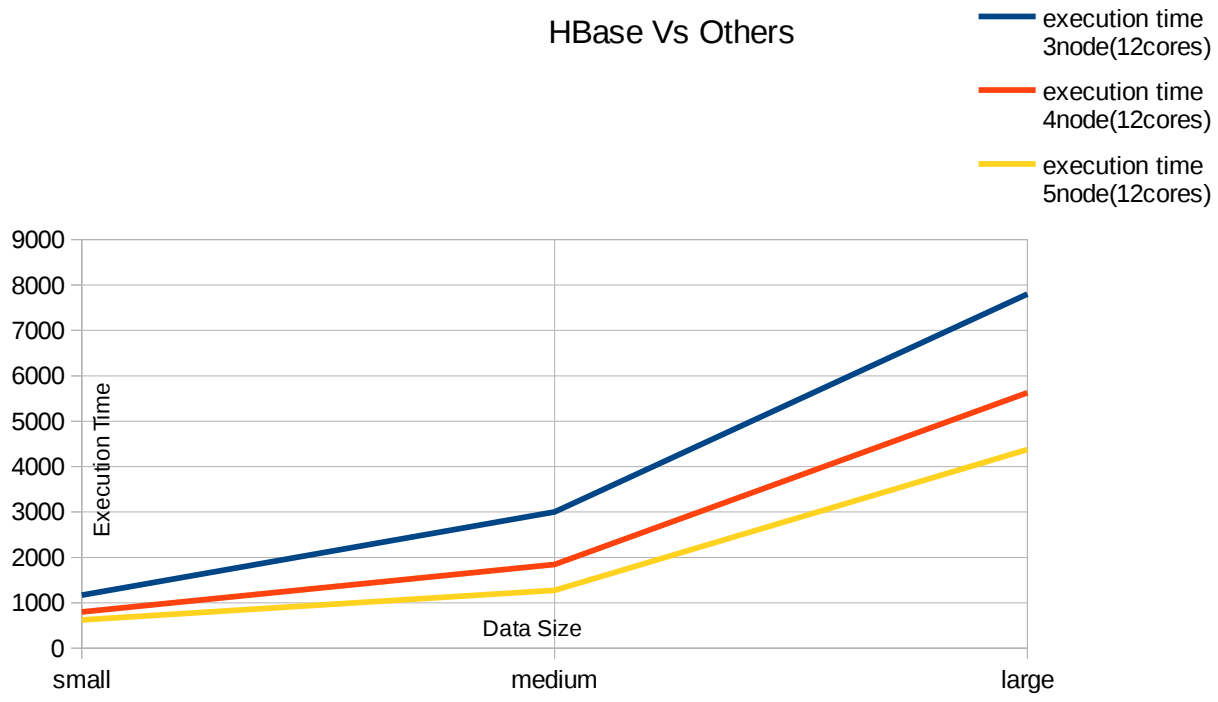
Performance Tables and Graphs:-

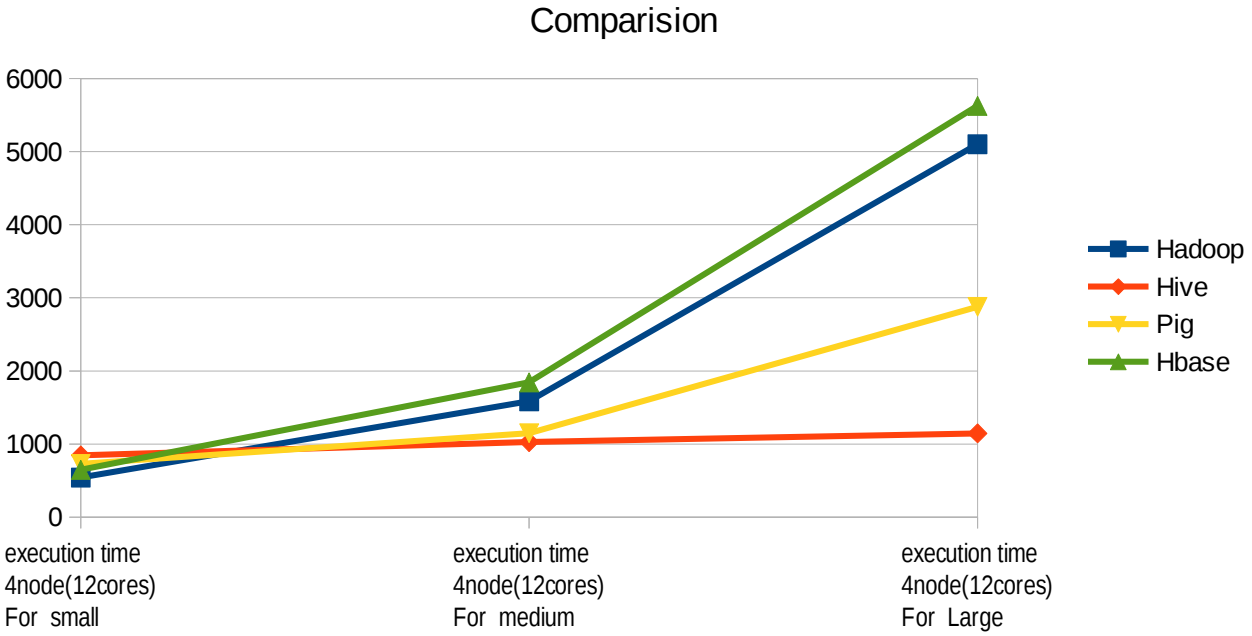
Hbase

Problem Size	execution time 3node(12cores)	execution time 4node(12cores)	execution time 5node(12cores)
small	1217	851	613
medium	2979	1721	1376
large	7934	5228	4176

Comparison

Problem Size	(12cores) Small	(24cores) medium	(48cores) Large
Hadoop	533	1483	5034
Hive	845	1027	1099
Pig	728	1148	2888
Hbase	647	1844	5628


HBase Vs Others
execution time 3node(12cores)
execution time 4node(12cores)
execution time 5node(12cores)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Execution Time
Data Size
small
medium
large


Comparision
6000
5000
4000
3000
2000
1000
0
Hadoop
Hive
Pig
Hbase
execution time 4node(12cores) For small
execution time 4node(12cores) For medium
execution time 4node(12cores) For Large

Observations And Results Comparing Pig, Hive and Hbase :

-It was found that the performance increases exponentially for larger amounts of data . The same logic when implemented in map-reduce required a significant amount of coding overhead, complexity in terms of understanding the configuration, context and Mapper and Reducer objects.

Although , The cost of performing logical operations is higher .

-Hive, takes much more memory to do the sort and risks running Memory. But it is simpler to code and understand especially, if knowledge of basic SQL commands and HiveQL structire is known in advance. It was realized that using simpler queries rather than complex joins helps ease the GC error.

To avoid it we can set the hive.map.aggr is false

-If we compare the results with those of hbase, We realise that Hbase is lacking considerably compared to pig and hive. The most probable reason for the same is that hbase was not designed to perform row based operations and instead concentrates upon the field based operations . Therefore, we rely severely to perform aggregate functions on java map-reduce. Also, it takes almost the entire aspect of hbase performance on reading and writing values in its table. Therefore we can govern our thoroughput primarily from the speed of storage and retrieval. Had the data been million rows thousand columns and involved basic colum operations, Hbase would have shown its true potential.