CSE587 : Data Intensive Computing (Spring 2015)

Project 3: Pig and Hive

Naveen Narayan

UB ID : nnarayan

UB Person # : 50134647

Aim :

The aim of this project is to use Pig and Hive to calculate volatility for given stocks. The application is run on data sets of varying size and also on varying number of nodes(on CCR) to compare and analyze performance on these factors.

Introduction :

Pig is a high-level platform for creating MapReduce programs used with Hadoop. The language for this platform is called Pig Latin. Pig Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for RDBMS systems. Pig Latin can be extended using UDF (User Defined Functions) which the user can write in Java, Python, JavaScript, Ruby or Groovy and then call directly from the language.

Pig vs SQL

In comparison to SQL, Pig

1. uses lazy evaluation,
2. uses extract, transform, load (ETL),
3. is able to store data at any point during a pipeline,
4. declares execution plans,
5. supports pipeline splits, thus allowing workflows to proceed along DAGs instead of strictly sequential pipelines.

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.

Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem. It provides an SQL-like language called HiveQL with schema on read and transparently converts queries to map/reduce, Apache Tez and in the future Spark jobs. All three execution engines can run in Hadoop YARN.

While based on SQL, HiveQL does not strictly follow the full SQL-92 standard. HiveQL offers extensions not in SQL, including multitable inserts and create table as select, but only offers basic support for indexes. Also, HiveQL lacks support for transactions and materialized views, and only limited subquery support.

Hive vs Pig

Hive because of its SQL like query language is often used as the interface to an Apache Hadoop based data warehouse. Hive is considered friendlier and more familiar to users who are used to using SQL for querying data. Pig fits in through its data flow strengths where it takes on the tasks of bringing data into Apache Hadoop and working with it to get it into the form for querying.

Implementation :

Pig –

- Stock data is provided in .csv files(<stockName.csv>). Each line in file corresponds to a day's trading information for that stock.
- All '.csv' files are loaded at once.
- Extract file name (stock's name ; for each file), Date and Adjusted Closing Price.
- Split the Date column into 2 columns, Year&month and Day.
- For each stock, extract beginning day and ending day of all months.
- For each stock, compute xi for each month.

$$xi = \frac{\text{(Month end adjusted close price – Month beginning adjusted close price)}}{\text{(Monthly beginning adjusted close price)}}$$

- Compute xbar.

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

- Count the number of months for which valid data is provided for each stock.

- Calculate volatility.

$$\text{volatility} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

- Ignore stocks with volatility 0.
- Extract the top 10 and last 10 stocks based on their volatility.

Top 10 stocks	Last 10 stocks
(LDRI.csv,5.149336582098077E-4)	(ACST.csv,9.271589761859984)
(GAINO.csv,5.650074160499658E-4)	(NETE.csv,5.396253961502244)
(VGSH.csv,0.0013014906189562881)	(XGTI.csv,4.542344311472958)
(MBSD.csv,0.0025000459104618893)	(TNXP.csv,3.2483321967818672)
(TRTLU.csv,0.003478105118698644)	(EGLE.csv,3.022206537901315)
(AGZD.csv,0.0039385938787697365)	(PTCT.csv,1.8462537015817715)
(SKOR.csv,0.003948740216629902)	(GOGO.csv,1.7793421751861378)
(AXPWW.csv,0.0044388372955839524)	(MEILW.csv,1.7188134065765308)
(VCSH.csv,0.00463776018563248)	(ROIQW.csv,1.396532083959149)
(GGAC.csv,0.004988207908071677)	(CFRXZ.csv,1.0792682449689408)

Hive –

- Stock data is provided in .csv files(<stockName.csv>). Each line in file corresponds to a day's trading information for that stock.
- All '.csv' files are loaded at once.
- Extract file name (stock's name ; for each file), Date and Adjusted Closing Price.
- For each stock, compute xi for each month.

$$xi = \frac{\text{(Month end adjusted close price – Month beginning adjusted close price)}}{\text{(Monthly beginning adjusted close price)}}$$

- Compute xbar.

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

- Count the number of months for which valid data is provided for each stock.
- Calculate volatility.

$$\text{volatility} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

- Ignore stocks with volatility 0.
- Extract the top 10 and last 10 stocks based on their volatility.

```
hdfs://localhost:9000/user/hive/warehouse/stockdata/LDRI.csv    5.1493366E-4
hdfs://localhost:9000/user/hive/warehouse/stockdata/GAINO.csv   5.650074E-4
hdfs://localhost:9000/user/hive/warehouse/stockdata/VGSH.csv    0.0013014906
hdfs://localhost:9000/user/hive/warehouse/stockdata/MBSD.csv    0.0025000458
hdfs://localhost:9000/user/hive/warehouse/stockdata/TRTLU.csv   0.0034781052
hdfs://localhost:9000/user/hive/warehouse/stockdata/AGZD.csv    0.003938594
hdfs://localhost:9000/user/hive/warehouse/stockdata/SKOR.csv    0.00394874
hdfs://localhost:9000/user/hive/warehouse/stockdata/CADT.csv    0.004156636
hdfs://localhost:9000/user/hive/warehouse/stockdata/AXPWW.csv   0.0044388375
hdfs://localhost:9000/user/hive/warehouse/stockdata/VCSH.csv    0.00463776
```

Top 10 stocks

```
hdfs://localhost:9000/user/hive/warehouse/stockdata/ACST.csv    9.27159
hdfs://localhost:9000/user/hive/warehouse/stockdata/NETE.csv    5.396254
hdfs://localhost:9000/user/hive/warehouse/stockdata/XGTI.csv    4.542344
hdfs://localhost:9000/user/hive/warehouse/stockdata/TNXP.csv    3.2483323
hdfs://localhost:9000/user/hive/warehouse/stockdata/EGLE.csv    3.0222065
hdfs://localhost:9000/user/hive/warehouse/stockdata/PTCT.csv    1.8462538
hdfs://localhost:9000/user/hive/warehouse/stockdata/GOGO.csv    1.7793422
hdfs://localhost:9000/user/hive/warehouse/stockdata/MEILW.csv   1.7188134
hdfs://localhost:9000/user/hive/warehouse/stockdata/ROIQW.csv   1.396532
hdfs://localhost:9000/user/hive/warehouse/stockdata/CFRXZ.csv   1.0792682
```

Last 10 stocks

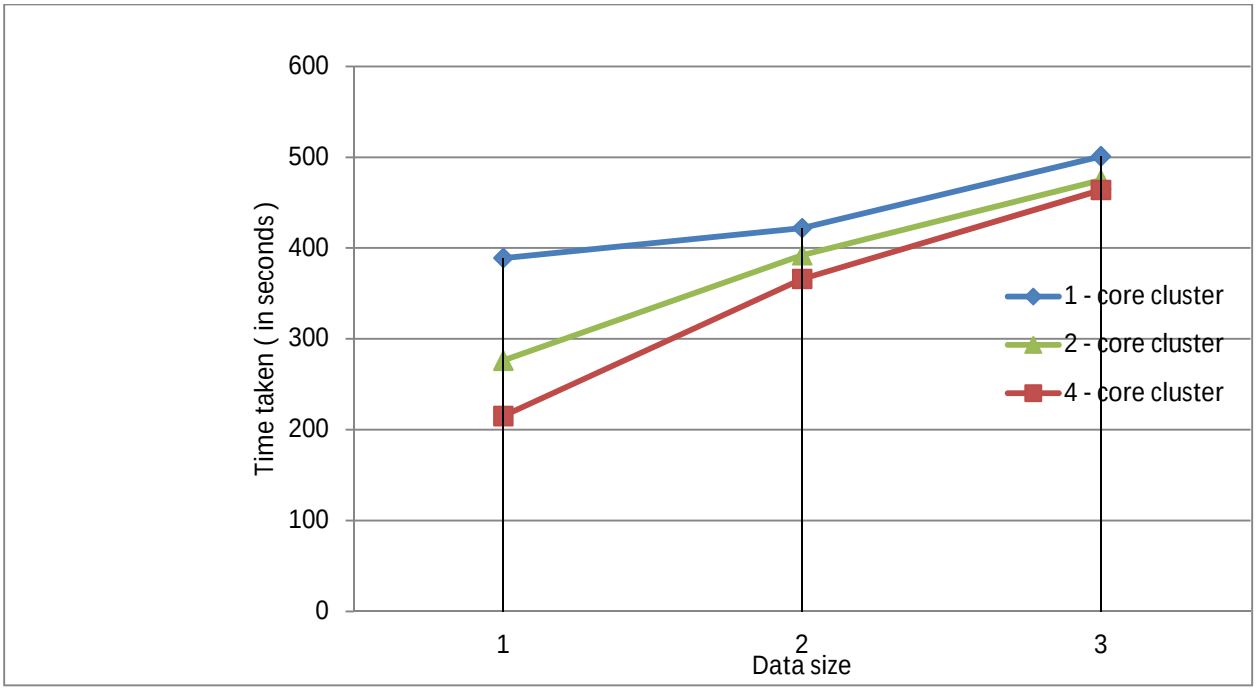
Observations :

Nodes ======== Problem size	1 - core	2 - cores	4 - cores
Small	2917	1317	879
Medium	10,378	4325	1789
Large	31,366	14,123	6134

Fig 1 : Map Reduce - performance on varying number of nodes and data size

Nodes ======== Problem size	1 - core	2 - cores	4 - cores
Small	389	276	215
Medium	422	392	366
Large	501	475	464

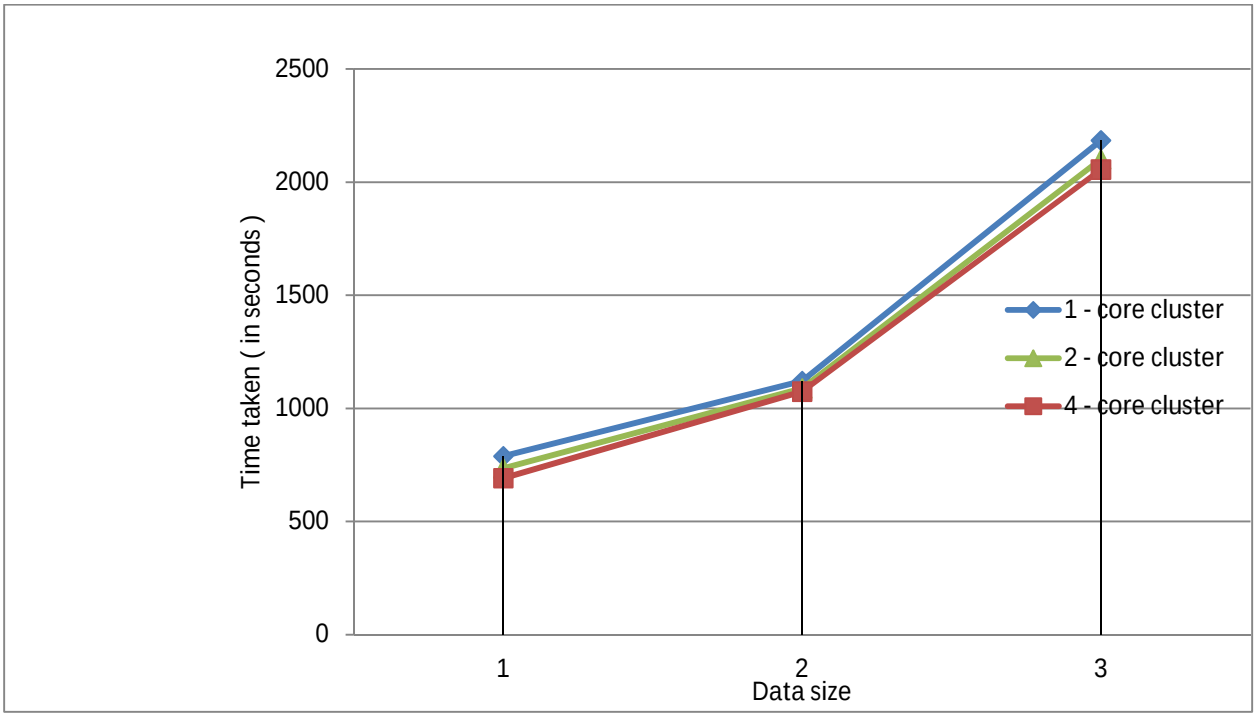
Fig 2 : Pig - performance on varying number of nodes and data size



Graph plot 1 – Pig - Performance based on data size and number of nodes

Nodes ======== Problem size	1 - core	2 - cores	4 - cores
Small	789	735	692
Medium	1,122	1089	1075
Large	2,185	2,102	2056

Fig 3 : Hive - performance on varying number of nodes and data size



Graph plot 2 – Hive - Performance based on data size and number of nodes

- Pig performs better than Hive and Map-Reduce while Hive performs better than Map-Reduce.
- Scalability (number of nodes, as well as dataset size) is more evident with Map-Reduce.

Softwares used :

Hadoop 2.5.2

Hive 0.13.1

Pig 0.14.0

References :

http://en.wikipedia.org/wiki/Pig_%28programming_tool%29

http://en.wikipedia.org/wiki/Apache_Hive

http://hortonworks.com/hadoop-tutorial/how-to-process-data-with-apache-hive/