Big Data Solution for Stock Prices and Tweet Collections

Paul Adams, paula@smu.edu, Rikel Djoko, rdjoko@smu.edu, and Stuart Miller, stuart@smu.edu

***Abstract*—Purpose-This dissertation aims to discover an optimal way to manage and process financial markets data to drive multiple algorithmic models for securities trading using the Hadoop ecosystem. This includes an applied, in-depth analysis of parallel processing using the Elastic MapReduce package on Amazon Web Services and the application of data warehousing using Apache Hive. Additionally, Apache NiFi is used to direct workflow automation scripting to migrate data into the warehouse. Finally, a complete assessment of the combinations of levels of the various Hadoop ecosystem applications used in this study will be provided in the context of statistical analysis for inference.**

Design, Methodology, and Approach-One pertinent, underlying hypothesis within this study is to prove that there are differences in processing speeds between S3 and HDFS using normalized and optimized schemas across multiple MapReduce configurations. This analysis is performed using 100 samples of the same volume of data in a repeated measures analysis using a Hotelling-T statistic. The two highest-performaning configurations of S3 and HDFS are then assessed. (We need to get this information and report it) for the next section.

Findings-Applying S3 with an optimized schema using 10 reduces, map memory allocation of 2,048 megabytes and a reduce memory allocation of 4,096 megabytes is optimal for a more expensive S3 approach. Using HDFS from local storage with a configuration of 20 reduces, map memory allocation of 8,096 megabytes and reduce memory allocation of 10,020 megabytes is ideal for batch-level migrations and querying for large-volume processing. Across n repeated measures, using a one-tailed alpha, our p-value is significant at Pr ¡ x.xxxx (confidence interval (x1, x2)), indicating local storage from HDFS outperforms S3 when using the selected configurations. However, local data storage capacity does not scale well for HDFS compared to cloud-based S3.

I. Introduction

DATA "starter file" in the twenty-first century is exploding in volumes at an exponential rate; every additional source of data that can act as a medium for data communication can obtain useful information, which, with modern technology, can be structured and stored, accessible to any who have the skills and need to make use of it. This information is increasingly profitable. As such, the scalability of storing and accessing this data must increase with it.

A. Apache Hadoop Ecosystem

Motivated through the opportunity within "big data" and parallel computing, which enables massive amounts of data to be rapidly accessed for complex analysis and distributed across a scalable, cost-effective distribution of servers, we aligned our project with a modern database application, Hadoop, which provides a data lake "ecosystem" supporting both task- and data-parallelism across the many software applications within the Apache suite in addition to software that can be managed and accessed within a network of servers, called a cluster.

B. Hadoop MapReduce

The cluster of server nodes enables users to read data from the same source, simultaneously, as the data at that source is partitioned and processed across multiple servers – a master and at least one slave – through leveraging a processing framework called MapReduce to assign nodes for "mapping" and "reducing" by applying various configurations, such as related to memory allocation to and volumes of mappers and reducers. As data is mapped, it is reprocessed into a derivative data set, split into tuples that are then processed across the cluster of server nodes, in parallel, and reassembled in the reduce process from which it is delivered to the end-user, whether it be Enterprise Resource Planning (ERP) or a personal user running a SQL "SELECT" statement.

C. Application Integration and Data Ingestion

This ability of Hadoop is very important to our problem, which requires rapid storing and accessing of data. Our data, which is stock market data obtained via API from a market data vendor every 15 minutes (primarily New York Stock Exchange and NASDAQ data) in addition to Twitter feeds updated every 5 minutes requires the ability to both store and enable applications to both connect to the data source as well as operate within it. R will be used via ODBC to access and build proof-of-concept models using the data. Once developed, Python will be deployed within the data lake to process and derive data from machine learning decisions.

D. Big Data System Implementation

In order to do some things with this stuff, we need a place to put it. Not just any place, but a special place that scales well, is quick to give and slow to take, and appreciates large volumes of data and potentially large volumes of simultaneous users. Therefore, we must usher in a new era of discussion; the dawning of the dueling between HDFS and S3. To make the battle interesting, we will root HDFS to local storage and S3 to the ever-scalable cloud. Winner takes all, battle royale. HDFS, meet local personal computer. S3, meet the Amazon

TABLE I
DATA FEATURES

Source	Features
Stock Daily	Prices: High, Low, Open, Close
Stock Intraday	Prices: High, Low, Open, Close
	High Bollinger bands
	Mid Bollinger bands
	Low Bollinger bands
	Nominal moving average
	Historical moving average
	Signal moving average
	Exponential moving average
	Stochastic 5-day oscillators K, D
Tweets	Text, URLs, Hashtags, Mentions, Users

Web Services' Elastic MapReduce package.
Furthermore, we need a data warehouse, so naturally, we opted for Apache Hive since it does a pretty good job with SQL-based HQL being easily communicated and stored within various applications and operated by traditional SQL users. This is all brought to you in part by Cloudera Hue. Cloudera Hue, thank you - without you we wouldn't actually be able to see any SQL at all. Not only are you a great face, but you're a great interface as well.

E. Database Schema and Selection

Flustered was the frivolous foe who quantified the schema not yet, ho. Finally, close the introduction out with some more explaining about normalized vs. optimized data, why that matters, and how we'll analyze it.

F. Performance Metrics and Evaluation

Performance is based on speed taken to process our 3 gigabytes of data - once processed into their respective storage systems - into the Hive data warehouse, which includes table creation and loading. We will use a repeated measures analysis and a one-tailed hypothesis test to determine optimal performance among S3 and HDFS groups, which is then used to compare and assess benchmarks between the combinations of MapReduce settings and database schemas among the best-performing S3 and HDFS configuration. This bench will be illuminated across endless samples engulfed in the event horizon - a poorly managed lake - endless light, yet still, perilous darkness.

II. DATA

This study investigates data warehouse models for housing stock price data and semi-structured alternative data. The stock price data was. Twitter was chosen as the primary alternative data source because of ease-of-access the Twitter API and the large volume of available data.

A. Stock Data

The stock price data was collected through an API provided by Alpha Vantage. This API provided access intraday stock prices, intraday price features, and daily prices. Intraday stock data contained 35 features sampled at 15 minute intervals. The intraday features were from the following categories Bollinger bands, stochastic oscillators, moving averages, and exponential moving averages. Approximately 1 millions rows of data were collected, which occurred from Oct. 04, 2019 to Oct. 24, 2019. During the collection process, the data were recorded in files organized by day and category. At the end of the collection process, the files for each category were combined and pushed to the HDFS datalake.

B. Twitter Data

sssss

III. DATA WAREHOUSE DEVELOPMENT

This stock data is natively structured in a tabular form and naturally keyed by the combination of timestamp, stock symbol, and stock market.

A. Normalized Schema

Discuss normalizing the data - 3NF

B. Optimized Schema

Discuss the difference for optimization

IV. TECHNOLOGIES

Rikel

V. RESULTS

Maybe a table comparing read times for the different options

- S3
- HDFS
- HDFS mapreduce settings?

VI. ANALYSIS

Analysis of the results

VII. CONCLUSION

The conclusion goes here.

REFERENCES

[1] W. H. Inmon, "Unstructured Data and the Data Warehouse," in *Building the Data Warehouse*, 4th ed. Hoboken: Wiley, 2005, ch. x, sec. x. Accessed on Nov. 6, 2019 [Online]. Available: https://learning.oreilly.com/library/view/building-the-data/9780764599446

[2] I. Moalla, A. Nabli, L. Bouzguendam and M. Hammami, "Data warehouse design approaches from social media: review and comparison," Social Network Analysis and Mining., Vol. 7, no. 1, pp. 1-14, Jan. 2017. Accessed on: Nov. 6, 2019 [Online]. Available doi: 10.1007/s13278-017-0423-8