Big Data Lake Solution for Warehousing Stock Data and Tweet Data

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

***Abstract*—Purpose - This research paper aims to discover an optimal solution for parallel management and processing of financial markets data into warehousing and analysis engines used for buy-sell decision-making. Methods analyzed herein are components of the Hadoop ecosystem. Included is an in-depth analysis of parallel processing - using the Elastic MapReduce package on Amazon Web Services - and data warehousing with Apache Hive. Apache NiFi is used to direct workflow automation for data migration into the warehouse. Finally, a complete assessment of the combinations of levels of the various Hadoop ecosystem applications used is provided in the context of statistical 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 schema designs 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-performing configurations of S3 and HDFS are then assessed. (We need to get this information and report it) for the next section.

I. Introduction

DATA in the twenty-first century is expanding in volumes at large rates. Every additional source of data that can act as a medium for data communication may contain 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 [1]. This information is increasingly profitable. However, with the increasing ability to capture and store data from many disparate sources, the need to store larger volumes of the data is likewise an increasing issue when it comes time to access and use the data, as many of the data gathered exist across very dynamic, diverse, and large partitions. As such, the scalability of storing and accessing big data must increase with it, relevant to the structures and locations of these data repositories. Developing a database with the Hadoop ecosystem, our team has built a data warehousing solution to structure and store the data based on both optimized and normalized schema designs, drafted from entity-relationship models designed with the intention of enabling rapid storing and accessing through the Hadoop MapReduce process [1]. Through a combination of these systems and data storage within Amazon Simple Storage Service (S3) and Apache Hadoop's native Hadoop Distributed File System (HDFS), we analyze performance between two approaches toward data- and task-parallelism using data gathered from the stock market. Motivated by the opportunity within "Big Data" to store and structure disparate data, we designed a system to store stock data and Twitter data, which could be used to service a large scale machine learning system. To accomplish this, we aligned our project with a modern database application called Hadoop. Hadoop 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 (a compute cluster) [1], [5].

The server cluster 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. The processing framework that enables this is called MapReduce, which assigns nodes for "mapping" and "reducing" processes by applying various configurations, such as related to memory allocation and numbers of mappers and reducers [10]. As data is mapped, it is reprocessed into a derivative data set, split into tuples that are then processed across the cluster, in parallel, and reassembled in the reduce process [10]. Following the reduce process, data is delivered to the end-user.

The parallelizability of Hadoop is central to this study as the primary objective is to rapidly store and access financial market data for buy-sell decision-making. The scope of applied analysis in this study is focused on the ability to scale and process a combination of quantitative stock market data and qualitative Twitter data related to the quantitative data.

Quantitative data was gathered from a stock market data vendor through an Application Programming Interface (API) on 15-minute intervals using the R programming language. Qualitative data was gathered and processed through a Twitter API using the Python programming language. The data was structured and stored in a Hive data warehouse, which was created and managed using Hive Query Language (HQL). R will be used via Open Database Connectivity (ODBC) to access and build proof-of-concept models using the data. Once the data warehouse is developed, Python will be deployed within the data lake to derive predictive data through machine learning.

In order to provide manageable storage repositories that scale well to size and data diversity, we have implemented this project using Simple Storage Service (S3) and Elastic MapReduce (EMR) from Amazon Web Services (AWS). S3 and EMR are designed for large sets of data. Therefore, integrity of data and ingestion systems are to manage. This is essential in an environment that may need to support many simultaneous users, each with different data needs. The EMR implementation in this project is housed across three - one master and two slave - Elastic Compute 2 (EC2) servers whereas S3 is a standalone repository within the AWS cloud

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 indicator (Slow K)
	Stochastic 3-day indicator (Slow D)
Tweets	Text, URLs, Hashtags, Mentions, Users

suite.

Additionally, Apache Hive is used as a data warehouse because it is operated with HQL, which is easily communicable for SQL users. Furthermore, Hive allows for storage of massive databases tables and is well-suited for big data application integration, including the Apache software suite. The data warehouse graphical user interface is provided by Cloudera Hue.

Two schemas were designed for the warehouse in a star configuration. This first schema was designed in a full normalized fashion, which is known as a snowflake schema. The second schema was based on the snowflake schema, but denormalized to limit the number of tables, which limits the number of joins required in queries. Query performance will be measured on these schemas to determine which design will be better for this use case.

Performance of the data warehouse is based on time taken to process the data collected for this project 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.

II. DATA

This study investigates data warehouse models for housing stock price data and semi-structured alternative data. Both daily and intraday stock price data were collected for use in this analysis. Twitter was chosen as the primary source of alternative data because of ease-of-access to the Twitter API and the large volume of available data. The main features of the collected data are summarized in Table I.

A. Stock Data

The stock price data was collected through an API provided by Alpha Vantage. This API provided access to daily prices, intraday stock prices, and intraday price features. 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 million 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 and stored in an S3 bucket. At the end of the collection process, the files for each category were combined and pushed to the HDFS datalake.

B. Twitter Data

Messages on Twitter (called tweets) are mainly comprised of tweet ID, timestamp, author (screen name), and text. Tweets can also contain many other features such as URLs, hashtags, emojis, and mentions. For this analysis, in addition to the main features, URLs, hashtags, and mentions were also collected in the data warehouse. A mention is a reference to another Twitter user's screen name in the text of a tweet. A hashtag is some collection of characters starting with '#' without white space. Generally, the character portion of a hashtag will be a word or set of words, but this is not necessary.

Approximately 300,000 tweets were collected from over 100 Twitter users. The data from Twitter is returned in JavaScript Object Notation (JSON). The features of interest were extracted from the JSON files and reformatted in tab separated files (TSV). After the tweets were extracted from Twitter, mentions of company names or stock symbols were extracted from the tweet text by matching sections of tweet text to company names and company stock symbols. Once the data was collected and processed, it was pushed to the HDFS datalake.

III. DATA WAREHOUSE DEVELOPMENT

Data warehouses are often conceptually designed with a *star* schema [2]. In the star design, there is a central table (called the fact table), which contains the unifying features of the dataset and keys to other tables [2]. The tables surrounding the fact table (called dimension tables) contain information related to a category of the facts [4]. In the dataset for this analysis, the unifying features are timestamp and company stock symbol, thus the fact table contains these features and several other descriptive features related to the companies. The natural dimensions of the fact table are intraday stock data, daily stock data, and tweet data. In some cases, dimensions of the facts exhibit an inherent hierarchical structure. When the dimensions of a star schema are broken out into the tables that make up the hierarchical structure of the dimensions, the schema is in its *snowflake* form [3]. In this study, two schemas were designed: one in snowflake form and the other in a more optimal form for query speed.

A. Snowflake Schema

As noted previously, the fact table of this star design contained the company information and timestamps. The primary key of the fact table is a composite key that includes stock

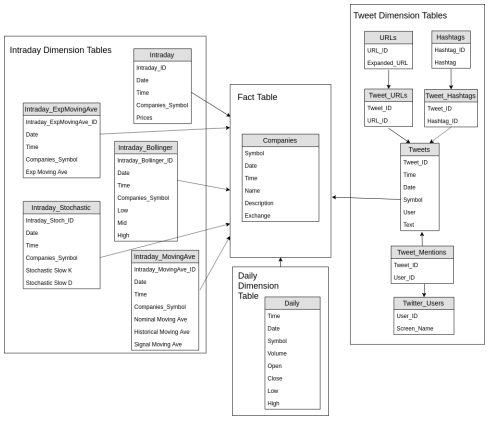

Fig. 1. Conceptual Diagram of the Data Warehouse Snowflake Schema

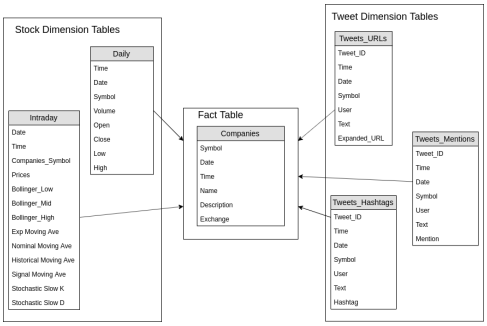

Fig. 2. Conceptual Diagram of the Data Warehouse Denormalized Star Schema

symbol and timestamp. There are three natural dimensions of fact table: daily stock data, intraday stock data, and tweet data.

Both daily and intraday stock data are naturally keyed by the combination of timestamp and stock symbol, thus naturally keyed to the fact table. The stock data was spilt into two dimensions: daily data and intraday data. The daily data comes in a form suitable for the snowflake design. However, the intraday data was normalized into five tables for the snowflake design: prices, Bollinger bands, moving averages, exponential moving averages, and stochastic indicators. The integration of the fact table and the normalized stock table is shown conceptually in Fig. 1 (left side of the schema diagram).

Unlike the stock data, the tweet data did not naturally join to the fact table of the schema. Like the stock data, the tweet data came with a timestamp, but stock symbol is not a nominal feature of tweets. The stock symbol feature was generated by extracting matching strings during the data collection process; therefore, stock symbols are not guaranteed to be non-null in the tweet dimension. Additionally, the combination of timestamp and stock symbol is not guaranteed to be a primary key into the tweet table for tweets with non-null stock symbol fields. However, Twitter assigns a tweet ID for each tweet, which is guaranteed to be a primary key. Thus, the same features can still be used to join the tweet dimension to the fact table, but cannot serve as a primary key. As noted previously, three secondary features of tweets are used in this study. Each of these features, mentions, hashtags, and URLs, represents a hierarchical member of the tweet dimension. Tables were created for each of these features to follow the snowflake design. These additional tables required join tables because there is a many-to-many cardinality between the tweet dimension and each of the three secondary members. The result of the tweet dimension normalization is shown conceptually in Fig. 1 (right side of schema diagram).

B. Denormalized Star Schema

While the snowflake schema is suitable for general use cases. The number of tables should be limited to decrease query time to support fast data transfer to a machine learning system. The number of intraday tables and tweet tables were reduced to support fast query times. All intraday tables were combined into one table. Each secondary member of the Twitter dimension hierarchical structure was subsumed into a copy of the main tweet table, reducing the number of tables in the tweet dimension from seven to three. The schema resulting from this denormalization process is shown conceptually in Fig. 2.

IV. Implementation

Multiple vendors provide platforms and infrastructure for data warehousing technologies, the project was implemented using Amazon Web Service (AWS). AWS is a set of cloud computing services provided by Amazon.com that are accessible over the internet. AWS provides multiple services for many different applications. Elastic MapReduce (EMR) was used for the Hadoop implementation in this project.

A. AWS Elastic MapReduce

AWS EMR is a pre-configured Hadoop compute cluster. The cluster can be provisioned and terminated on demand as needed. It comes with a configurable set of the Hadoop ecosystem elements pre-installed and ready to use. Fig. 3 shows the ecosystem of EMR ecosystem.

B. Apache Hadoop

Hadoop is an open source software framework for the distributed storage and distributed processing of a very large datasets. The core of Apache Hadoop consists of a storage part: Hadoop Distributed File System (HDFS) and a processing part MapReduce.

1) Hadoop Distributed File System: Unlike traditional file system, HDFS is a distributed file system designed to run on commodity hardware. HDFS the architecture consists of a master (NameNode) and at least one slave (DataNodes) [11]. The NameNode is the controller which consist storing data and metadata of the data no the DataNodes. The metadata includes a Namespace lookup table used to locate each file from the DataNodes. The Fig. 4 shows the Architecture of HDFS [6].

2) Hadoop MapReduce: Hadoop MapReduce is a programming model for large scale data processing. With the high demand of computing power, computer architecture has evolved over the years from serial computing to parallel computing where tasks are distributed and executed simultaneously across multiple processors within the same computer.

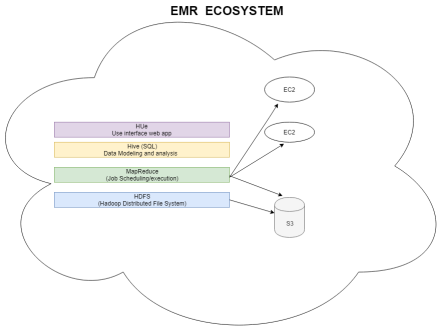

Fig. 3. Elastic MapReduce Ecosystem

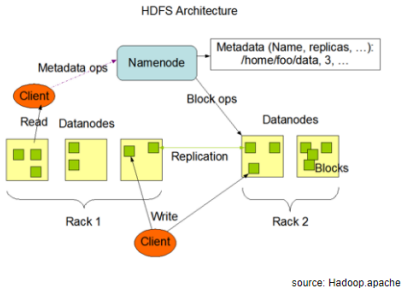

Fig. 4. HDFS Architecture

With MapReduce, instead of using one server instance with multiple processors, multiple servers with multiple processors are used for computation. This is called distributed computing system. The parallelism in Hadoop is categorized into data-parallelism and task-parallelism. Data-parallelism is focused on processing data across multiple processors whereas task-parallelism is focused on distributing execution threads across multiple servers [7]. Utilizing one or the other is dependent upon whether data is being provided into the system or being utilized for events such as machine learning. This is what enables management and processing of ”Big Data” (multi-terabyte datasets) in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. A MapReduce framework consists of two types of workers: mappers and the reducers.

MapReduce uses the divide and conquer technique where the input is divided into a set of small tasks and each task is identified by a key-value pair [8]. The key serves as a task ID and the value as the task output. Each task is then processed and executed by the mapper and the outputs are processed and merged by the reducer [9].

C. AWS Simple Storage Service

As the name implies, the S3 is a storage system. It is used to store and retrieve any amount of data any time, from anywhere on the web. S3 is reliable, fast, and inexpensive.

D. Apache Hive

Apache Hive is a data warehouse application built on top of Hadoop. Hive was used to structure, organize, model, and query the data using HQL.

E. Cloudera Hue

Cloudera Hue is an open source web application that served as user interface for Hadoop components. Hue provides access to Hadoop from within a browser, allowing users to interact with the Hadoop ecosystem applications. Hue is an alternative to accessing the Hadoop ecosystem applications from the command line interface. Hue was used to interact with the EMR cluster and run Hive scripts.

V. Results

Maybe a table comparing read times for the different options. Performance testing needs more research. Results may be contained in something like Table II.

- Snowflake vs Denormalized Star
- S3 vs HDFS
- HDFS mapreduce settings?

VI. Analysis

EXAMPLE Discussion of Analysis: Applying S3 with an optimized schema using 10 reduces, map memory allocation of 2,048 mb and a reduce memory allocation of 4,096 mb 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, the resulting p-value is significant at ‘Pr ¿ —t—‘ ¡ 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

- Discuss sample size of measures
- Looks like a repeated measures analysis

VII. Conclusion

EXAMPLE Conclusion: The scalability of large sets of data is of ever-increasing importance in the data community, which itself is ever-increasing with the advancement of modern technology. Modern technology itself is enabling both an increase in both data volume as well as data diversity being captured and stored. Consequently, parallelism is of utmost importance in making use of this data. Herein analyzed are methods that are proven within the scope of big data. As modern technology continues to advance, these parallel principals will serve as the root from which methodologies will be developed. As with the businesses and industries the data captured will serve, there will remain a trade-off that must be assessed for each business model. This study has provided two primary storage methods - S3 and HDFS - in addition to different data warehousing schema design - normalized versus optimized, star versus snowflake - and data distribution

TABLE II
RESULTS OR ANALYSIS TABLE

Storage Method	Schema	MapReduce Settings
S3	Normalized	MapConfig1 / ReduceConfig1
		MapConfig1 / ReduceConfig2
	Denormalized	MapConfig2 / ReduceConfig1
		MapConfig2 / ReduceConfig2
HDFS	Normalized	MapConfig1 / ReduceConfig1
		MapConfig1 / ReduceConfig2
	Denormalized	MapConfig2 / ReduceConfig1
		MapConfig2 / ReduceConfig2

- herein, MapReduce - techniques. Through analysis using repeated measures, S3 outperforms HDFS when X, Y, Z and HDFS outperforms S3 when Z, Y, X. Furthermore, MapReduce configurations are constant across both storage methods - HDFS and S3 - with an optimized star schema performing faster, but a normalized snowflake schema performing more reliably within a conventional three-tiered system architecture.

REFERENCES

[1] R. Kune, P. Konugurthi, A. Agarwal, R. Chillarige, R. Buyya, "The Anatomy of Big Data Computing," Software: Practice and Experience, Vol. 46 no. 1, pp.79-105, Jan. 2016.

[2] W. H. Inmon, "Unstructured Data and the Data Warehouse," in *Building the Data Warehouse*, 4th ed. Hoboken: Wiley, 2005, ch. 11. Accessed on Nov. 6, 2019 [Online]. Available: https://learning.oreilly.com/library/view/building-the-data/9780764599446

[3] 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

[4] A. Gorelik, "Historical Perspectives," in *The Enterprise Big Data Lake*, 1st ed. Sebastopol, CA: Wiley, 2019, ch. 2, pp. 25-47.

[5] "Extract, Transform, and Load Big Data with Apache Hadoop," Intel, USA, 2013. Available: https://software.intel.com/sites/default/files/article/402274/etl-big-data-with-hadoop.pdf

[6] D. Borthakur, HDFS Architecture Guide, The Apache Software Foundation, August 22 2019. Accessed on: Nov. 8, 2019. [Online] Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[7] D. Sitaram, G. Manjunath *Moving To The Cloud*. Boston: Syngress, 2012, pp. 205253.

[8] S. Zhao, R. Li, W. Tian, W. Xiao, X. Dong, D. Liao, S. U. Khan, L. Keqin, Divide-and-conquer approach for solving singular value decomposition based on MapReduce, Abbrev. Title of Journal, vol. 28, no. 2, pp. 331350, Nov. 2016. Accessed on: Nov. 10, 2019. [Online]. Available: doi:10.1002/cpe.3436,

[9] D. Miner, A. Shook, MapReduce Design Patterns, 1st ed. Sebastopol, CA: O'Reilly Media, 2012, pp. 56.

[10] D. Miner, A. Shook, Hadoop MapReduce v2 Cookbook, 2nd ed. Sebastopol, CA: O'Reilly Media, 2015, pp. 1.

[11] T. White, Hadoop: The Definitive Guide, 1st ed. Sebastopol, CA: O'Reilly Media, 2009, pp. 44.