Benchmarking Hive on Spark and SQL Server with the Real Time Data Warehousing Chain.

Ian, Bassett
X06709711
MSc. Masters in Data Analytics
National College of Ireland
School of Computing

***Abstract*—The following paper focuses on the field of Data Warehousing in two aspects. The first aspect will review Big Data performance comparing the emerging Hive on Apache Spark with SQL Server to determine when it would be appropriate to switch to a big data platform. The other aspect will investigate current software in the industry and how the continuous support of communities are creating to solve current and future barriers in the profession. A current issue in Data Warehousing and Business Intelligence is the development of Real Time Data Warehousing. This paper documents the research and progress of tools in the automation process of Real Time Data Warehousing.**

Keywords- Data Warehousing; Benchmarking; OLAP Cube; Salesforce; Sentiment Analysis; Python; Real Time Data Warehousing

I. Introduction

The field of Data Analytics is continually evolving. New software and theories on approaching the latest challenges and limitations are being addressed; where the current established practices are being reviewed and altered to the needs of the industry. With the transformation of data within the last decade, data is a key structure in companies staying competitive. The current obstacles in the profession are the continuous growth of data and the limited amount of time employees have in analysing.

During the attempted acquisition of EMC, Founder and CEO of Dell Inc. Michael Dell expressed the importance of securing EMC based on their access to the data market. Dell examines how data is integrated into a wide variety of digital products, showing the opportunities that come from customers and industries generating data, placing the value in trillions of dollars.[1]

With the potential of data in discovering insights, trends and gaining a strategic advantage in the market, Dell and other companies are investing and placing Data Analytics as a high priority in their success. This viewpoint is supported as the International Data Corporation (IDC) is currently forecasting the market for Big Data technology to grow at 23.1% compound annual growth rate reaching $48.6 billion in 2019 with spending on self-service data preparation tools and visual discovery growing 2.5 times faster than traditional IT-controlled tools.[2]

Currently the industry‘s biggest challenge is to reduce the time to prepare data and to allow Data Scientists more time to examine and discover Business Intelligence. Forrester [2] believes that in 2016, Machine Learning will begin to replace manual Data Wrangling and Data Governance to make Data Ingestion, preparation and discovery quicker.

Benchmarking Hive on Spark and SQL Server with the Real Time Data Warehousing chain is a practical assessment measuring Big Data performance between Hive on Apache Spark and SQL Server through the developed environment, Real-Time Data Warehousing (RT-DW) Chain. An environment that investigates current industry tools in the implementation of Real Time Data Warehousing. The research question‘s objective is to reduce time by converting manual into automatic processes to allow Data Analysts more time for analysis

The organisation of the paper consists of five remaining sections. In section II, “Related work” is discussed with respect to the components and approaches in developing a Data Warehouse, evaluating the current practices in Real Time Data Warehousing and how certain concepts led to the development of the RT-DW chain. “Related Work” will then continue examining the reasoning and importance behind Benchmarking Big Data performance between SQL Server and Hive on Spark, establishing the Benchmarking approach and considerations when choosing data.

In section III, “Design” will document the operation of the chain, the choosing of data sources and the performance parameters being implemented for testing. “Design” will conclude with user experience requirements, documenting the needed skills for successful deployment of the RT-DW Chain.

In section IV, “Implementation” will acknowledge the importance of business input in developing the RT-DW chain. “Implementation” continues by examining the building process for testing data. Concluding with how the prototype runs in action.

In section V, "Evaluation" will begin with establishing the method taken to gather and review results. The RT-DW chain performance will be assessed by seeing how the data responded focusing on the benefits, limitations and how user experience can effect the quality of the chain. "Evaluation" will then continue by contrasting between SQL Server and Hive on Spark based on the performance parameters and concluding on which is the more promising choice moving forward.

Finally section VI, "Conclusions" will recap what has been achieved in this paper and the direction future work should take moving forward.

II. Related Work

A. Components of a Data Warehouse

Defined by Inmon[3] a Data Warehouse is *"a subject-oriented, integrated, time-variant and non-volatile collection of data that is primarily used in organizational decision making"*. As seen in Fig. 1, Data Warehouse architectures are comprised of three components, Extract Transform and Loading (ETL) from multiple data sources, the Data Warehouse and Analysis consisting of Online Analytical Processing (OLAP) cube, Reporting and Data Mining.[4]

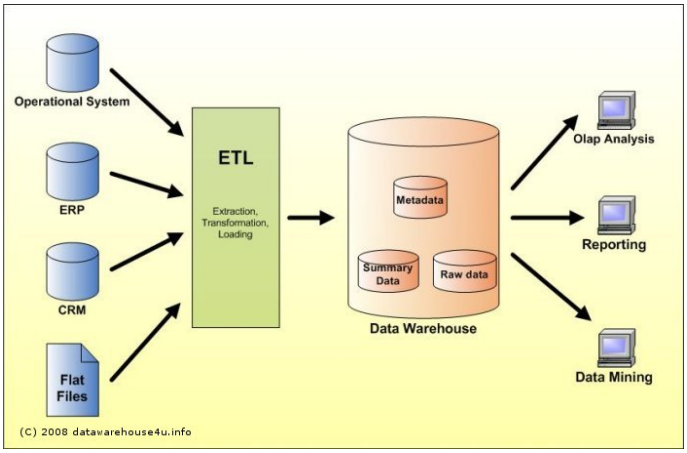

Fig. 1. Components of a data warehouse. [24]

Building a data warehouse is widely accepted through two approaches, Inmon's data driven Corporate Information Factory[5] and Kimball's business driven Data Warehouse Bus built through Dimensional Modelling and Data Marts [6].

B. Development of the RT-DW chain

Haisten[7] reviews the history of Data Warehousing and documents that Inmon's approach created the snap-shot concept, capturing data at a certain point of time and establishing Data Warehousing as a Decision Support System add-on, being independent of the source systems. Haisten continues to state that Data Warehousing should evolve from the snapshot approach and instead access data streams to write data into the warehouse environment through operating systems.

Though Haisten's approach is intriguing, getting data from streams may not be suitable in every aspect. Owen [8], Data Scientist and director of Cloudera distinguishes between Stream processing and Batch processing. Owen describes streaming as useful for computing a single data element or a small segment of recent data with streaming performing something relevantly simple. Owen continues by contrasting with Batch processing having access to all data with the ability to compute large and complex data. Taking this into account, the best approach to leverage the benefits of Batch and Streaming approaches is to develop a concept that satisfies both.

The development to satisfy both presented in this paper is the Real Time Data Warehousing (RT-DW) chain. Creation of the chain concept originated through two main ideals. Nair [9] proposed Supply Chain Analytics, an approach that combines Data Analytics with Supply Chains offering advanced capabilities ranging from drill down views, scenario and what if analysis, pattern and trend analysis and dashboards among other capabilities. Ranjan et al [10] researched Real Time Business Intelligence (RT-BI) and the proposed interior architecture with Supply Chain Analytics, detailing the progression from reactive to proactive Business Intelligence through automation.

In the creation of the RT-DW chain, both ideals contribute on how the chain should be developed. Supply Chain Analytics focuses on the business objectives and determine what needs to be achieved. The RT-DW chain should focus on what will contribute to the business objectives. In terms of Real Time Business Intelligence, data must constantly be updated; this requires the RT-DW chain to be automated on a timely basis. Building the RT-DW Chain will be based on Kimball's[14] approach of starting small and develop based on the business process through Dimensional Modelling. The RT-DW chain will be customizable based on the knowledge worker's desired objective. To determine the success of the RT-DW chain Big Data performance will be measured by comparing SQL Server and Hive on Spark.

C. Big Data performance: Hive on Spark and SQL Server

In terms of analyzing big data performance between SQL Server and Hive on Spark Price Waterhouse Coopers (PwC) and Iron Mountain [15] conducted a study that included 1,800 business leaders in North America and Europe researching their use of their respective data. The results show a disconnection between Data Science and business in data strategies, competitive advantage and a lack of focus in organizational investment when it comes to the right analytical tools and talent. [16]

This lack of understanding shows that companies need to better understand when to switch to a big platform. SQL server and Hive on Spark are Benchmarked together to determine this migration. Hive on Spark is a port of the Apache Hive SQL engine to run over Spark instead of Hadoop. Its supported by the Hive Query Language (HQL) and runs over Hive Data Warehouses.[17] In comparison with Hadoop, Hive on Spark was chosen to benchmark against SQL Server because it can run machine algorithms 25 times faster than Map Reduce programs and answer queries 40 times faster than on Hive on the Hadoop Platform. [18]

The importance of gathering results from Hive on Spark and SQL Server through Benchmarking can offer multiple benefits in a corporate environment as Managing Director of CFO Edge, Rothberg [20] highlights how Key Performance Indicators (KPIs) through Benchmarking can improve areas in operations, productivity and profitability. The Benchmarking approach to measure the SQL Server and Hive on Spark is comparing Big Data architecture and systems under user concerns[26] while using diverse and real world datasets. Suitable data requirements are based on the characteristics of Big Data, V3; Volume, Variety and Velocity.[29]

III. Design

A. Operation of the RT-DW chain.

The RT-DW chain will measure the performance between SQL Server and Hive on Spark through Python. Python was chosen over other languages such as Java, PHP and R as Python has a range of modules and packages that can support Real Time Data Warehousing. Also as a scripting language, Python can connect multiple existing components together as seen in Fig 2.

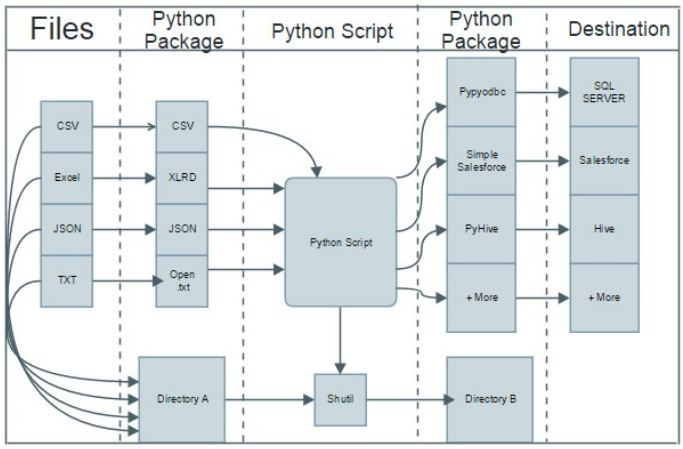

Fig. 2. Connecting existing components in Python.

To measure Big Data performance between Hive on Spark and SQL Server, two VMware environments were created to measure performance on an equal playing field. As Table 1 shows, each VMware workstation is similar with the only exception being the operating systems and tools for the data warehouses. Hive on Spark will be run on Ubuntu 14.04.LTS and SQL Server will be run on Windows 10 operating system.

Enviroment:	Ubuntu	Windows
Memory:	4096MB, 4GB	4096MB, 4GB
Processor:	1	1
Hard Disk:	80GB	80GB
CD/DVD:	Ubuntu 14.04 LTS	Windows 10
Network Adapter:	NAT	NAT
VM WorkStation:	11.0	11.0

Table 1: VMware Environments

The RT-DW Chain is developed by taking into account components of the Data Warehouse architecture as shown in Fig 1. The design of the chain will be to treat each part of the chain as a block with a specific role that needs to be achieved. In this case, this involves inserting and Benchmarking data into the warehouse, get a cube analysis and produce results. To show the flexibility of the chain, cube results are brought into the Software as a Service (SaaS) provider Salesforce. A Sentiment Analysis is also produced to show the Data Mining capabilities of the chain.

Every block is represented through a master block that activates each slave block in order through a recurring time sequence from the operating system. A master block is a Python script that calls Operating System (OS) and Sub process calls to the slave blocks, which are Python scripts that have a set of functions to carry out.

On the operating system level, a directory represents each block. Files are placed in a starting directory called block 1. When the master block activates the Python script associated with the first block, it processes the functions and moves the file into the second block/directory. The process is repeated until the file enters through all blocks to the end.

As seen in Fig 3, this process allows multiple tables, Data Marts and Data Warehouses to be populated from a sequence of scripts or one giant script. Additionally, the RT-DW chain will query and update multiple Data Warehouses at the same time, producing the results in one location. In both cases it will reduce manual input and focus on more time for analysis.

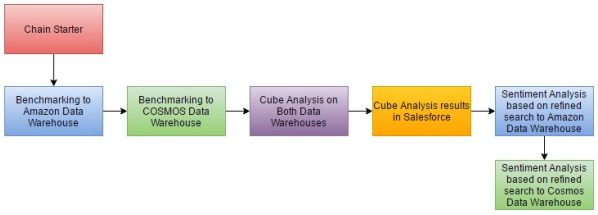

Fig. 3. Real Time Data Warehousing (RT-DW) Chain Concept

B. Data Sources

Two different data sources were chosen to examine and push each environment to its respective limits to assess time and durability. The first data source is the Amazon Review dataset.[25] Amazon has been used in a range of papers from Big Data Benchmarking[26] to SIGIR and KDD papers.[11,12] From a data perspective, Amazon was chosen for testing as the raw data consists of over one hundred and forty two million records, broken down into two main formats Reviews and Metadata. In terms of data volume, the data is suitable for stress testing. The quality of the data was another key factor as it allowed analysis to be conducted. This expanded the data into multiple new categories developing the Data Warehouse further for Benchmarking.

The second data source is COSMOS [19], a project that brings a range of Data Scientists together to investigate the ethical impact of big social data, the development of new methodological tools and technical/data solutions for the UK academic and public sectors. In contrast to Amazon, COSMOS was chosen for testing as the data is focused solely on social media[13]. COSMOS data capacity is limitless as social media data is recorded through the COSMOS Streaming API. Finally Cosmos was selected because the data presented for Benchmarking is more compact in the Data Warehouse schema structure.

C. Performance Parameters

To measure the performance between SQL server and Hive on Spark, five Key Performance Parameters (KPI) are chosen to conduct experiments on in each environment.

These performance parameters are:

- Time
- CPU
- Memory
- Disk
- Data Loss

When choosing these KPIs to measure, time was chosen because to get the latest up to date information, the RT-DW chain and Data Warehouses will have to deliver results as close to zero latency as possible. CPU, Memory and Disk Performance were chosen because the chain is dependent on how each environment can withstand.[27] These parameters determine how much the computer can process and the possibility of overheating and crashing could occur, which would damage the system. Finally Data Loss is recorded to discover any data leakage, loss and damage. [21] As the cost of data loss can lead to inaccurate results and in terms of sensitive data a loss of ten thousand records could cost between $1.5 and $2 million dollars [22]

Each file is benchmark tested using CSV format through lines, the reasoning behind this is based on the size limit. CSV can contain more data compared to spreadsheets such as Microsoft Excel. With regards CPU, Memory and Disk Performance, files are loaded together into the RT-DW Chain so all pre-function results are similar when recording starts. Table 1 shows the data volume loads through the RT-DW chain.

Amazon Record count	Cosmos Record Count
2500	2500
10000	10000
25000	25000
50000	50000
100000	100000
250000	250000
500000	500000

Table 2: Benchmarking Tables based on CSV file lines

D. User Experience Requirements

To successfully deploy the RT-DW Chain, expertise in four skills are required:

- Data Warehouse Architecture
- JavaScript
- Python (or a similar programming language)
- Operating System experience

Data Warehouse Architecture experience is needed to connect between Fact and Dimension tables and to not comprise data while importing. JavaScript is needed to develop a cube model that reflects a Data Warehouse for cube analysis. Python is needed as the blocks are built from Python and require experience with programming concepts, range of different packages and understanding syntax. Operation Systems experience requires how to automate the system.

IV. Implementation

Incorporating the RT-DW chain to measure Big Data performance and to convert manual into automated processes requires taking into account a number of key points for successful deployment. This involves three phases, Business Preparation, RT-DW chain development and Big Data Execution.

A. Business Preparation

Before starting development, it is important to take into account what needs the chain's objective is attempting to fulfil and what data sources are needed to achieve the end result. Porter et al[23] describes it best by saying *"A data warehouse must deliver the right data to the right people. However, the data warehouse cannot deliver all the data people want."* Based on this business emphasis, a data warehouse schema is created based on the importance, usefulness and quality of each data segment.

To measure the user's level of experience in developing Data Warehouses, the Data Warehouse Architect will need to look at what is necessary in the schema when creating tables by taking into account how improper use of different data variables may tweak results and cause incorrect execution and analysis. Once the Data Architect has finalized the Data Warehouse structure and the RT-DW chain to the needs of the business, the next stage involves development.

B. RT-DW Chain Development

Based on the business plan, the Data Warehouse will be implemented in its environment and the RT-DW chain will have three blocks, containing a benchmarking, cube analysis and processed block. On the operating system level a root directory is developed containing three sub directories. The reasoning behind this is that all files will be located together in the same area and processing paths are closer when executing.

In terms of benchmarking data into the warehouse, the approach involves reading files from a directory, interaction with the server, loading multiple tables and passing the file to another directory. Data is loaded into Hive and SQL Server table through bulk loading where two requirements must be met. This involves temporary files and support tables. With regards benchmarking results between SQL Server and Hive on Spark, performance parameters are recorded in the benchmarking script and loaded into their individual table. CPU, Memory and Disk are recorded through variable assigned snapshots that start at the beginning of the script and repeats after each table import. Time is documented by subtracting pre and post table load variables. Data Loss testing is saved by querying the Data Warehouses and CSV files using the Count Function. The file count subtracts with the post Data Warehouse count to determine if there is any data loss.

Once loading is complete, the second block will activate to perform cube analysis. This involves Python interacting with the server again, establishing workspaces and importing a user created cube model to perform drill downs, slicing and querying on multiple Data Warehouses. As seen in Fig 4, The cube model is built in JSON format and requires JSON experience in assigning dimensions, measures, aggregates, mappings, connection to established joins among other features to the cube. To interact with Salesforce, Data Architects communicate through the REST API using Python as long as the user has a username, password and security token.

To implement Data Mining into the chain, a range of techniques can be used. For this instance, a Sentiment Analysis is conducted by interacting with CSV files in multiple blocks and a web service[28]. After the results are processed, the data can be imported into a data warehouse

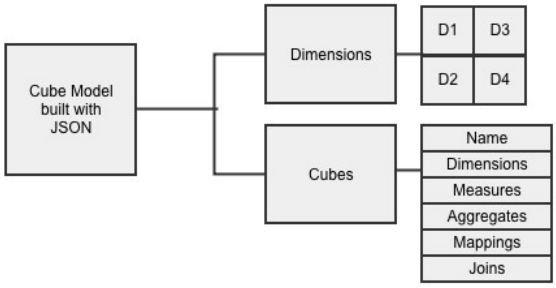

Fig. 4. Requirments to build a JSON cube model.

table, Salesforce using a Bulk API or exported into a file.

To build the master block to activate each Python script. A script consisting of only operating system and sub process calls are required. As seen in Fig 5, this master script can be enhanced to start up a server, create file count requirements and with the flexibility of calls, the user can decide which scripts to turn on or off. To implement Real Time Data Warehousing, Windows Operating System can run the chain starter script through Windows Task Scheduler at a minimum of 5 minutes; where on a Linux operating system Ubuntu Crontab can run tasks at a minimum of one minute. Each approach can repeat the running of the Chain Starter Script after N amount of minutes.

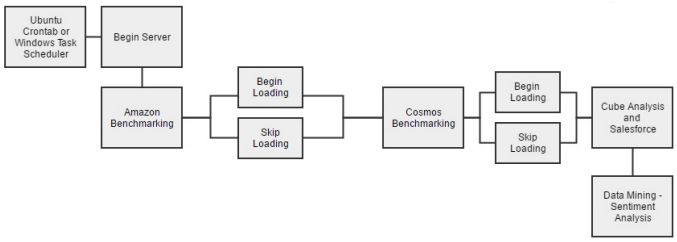

Fig. 5. Master block to activate the RT-DW chain.

C. Big Data Execution

With Data from multiple sources and analysis needed to be conducted before Benchmarking, data is prepared through combining multiple data frames into Denormalization as one giant flat file. N amount of files are then placed in the starting directory. After N amount of minutes, Crontab or Windows Task Scheduler runs the chain starter script. The script starts the server and checks if N amount of files are in the beginning directory of each warehouse.

If N is greater than the requirement: Benchmarking will begin loading the specified tables in the python script through temp files and support tables.
If N is less than the requirement: Benchmarking is skipped and the chain moves to the next phase of the chain.

The cube Salesforce block then performs cube analysis from the warehouses and uploads the results into Salesforce. Finally Data Mining Sentiment Analysis communicates with the web service[28], processes and loads the results into the user's desired format. The process is then repeated after N amount of times.

Cube_Record	Amazon_Price	Amazon_Record_Count	Cosmos_Unique_Tweets	DateTime
A-0127	None	0	2533	2016-04-10 16:02:43.112000
A-0128	1455.86	4	2533	2016-04-10 16:13:43.004000

Fig. 6. Cube Results in Salesforce.

V. Evaluation

A. Evaluation Method

As seen in Table 1, fourteen datasets are chosen for testing, seven from Amazon with the remainder from COSMOS. These datasets range from 2,500 to 500,000 records and the volume of data is equal between Amazon and Cosmos. These datasets are placed into the RT-DW chain and benchmarked together during stress testing.

There are three cycles of testing:

- Amazon: testing is conducted with Amazon data.
- Cosmos: testing is conducted with Cosmos data.
- Amazon and Cosmos: testing is conducted with both data sources.

Each cycle is broken down into five sub sections to determine the effects duplication has on the RT-DW chain. Denormalised files are divided into the following duplication parameters:

- 0%
- 12%
- 24%
- 36%
- 48%

The testing process is repeated three times to determine variation in results.

B. RT-DW Chain Evaluation

The RT-DW chain has been optimized from the initial concept. The first concept originated from accessing a directory manually similar to an SSIS For Loop Container and executing files through a directory. This then evolved to using Ubuntu Crontab and Windows Task Scheduler and a pre-defined directory destination path. Changing from a manual to an automated process similar to Haistens approach of using operating systems is a positive direction moving forward as Data Analysts just need to acquire data and place it in a directory and let the operating system take care of it.

The benefits are:

- Reduction in time, leading to more focus on other activities.
- Removal of continuous manual input when importing data into the warehouse.
- Removal of continuous manual input when conducting Data Mining.
- Cube analysis is now based on searching up the latest results from the data warehouse/Salesforce

However, the drawback is shifting responsibility and being dependent on the operating system. This will require users to have a sustainable machine that can meet and handle the demands of the processes being implemented. This is supported as processing large datasets, preventing data loss and to remove manual processes from the user requires more pressure on the operating system.

During Benchmarking, pressure on the operation system was taken into account by moving from the originally programmed 'line-by-line' to bulk loading and support tables. Bulk loading and support tables were also used because:

- Bulk loading is faster to process compared to 'line by line'.
- Python can only update Hive through bulk load.
- Small data can support 'line to line' but as volume of data increases, data loss happen on SQL Server and can create key errors.
- Support tables are used because though there are unique denormalised records, they are not unique when normalised.
- Support tables are used to filter data because Hive does not have a key system similar to SQL. This filter converts tables into Primary Key tables and provides Hive with accurate cube analysis.

When Benchmarking data into the RT-DW chain, evaluation began by using small datasets to discover if cube analysis results were accurate or inaccurate. In terms of a knowledge worker's level of experience, the results will be based on how the user developed the Data Warehouse and the creation of the cube model. This was examined further by altering the Data Warehouse by placing columns into different tables and switching compound keys with primary keys. The results showed data loss and inaccurate cube analysis compared to the properly implemented Data Warehouse. Once accurate results were recorded in the cube, data loads were increased to determine Benchmarking performance.

C. KPI Benchmarking: SQL Server VS. Hive on Spark

Test: Time performance
Result: Hive on Spark outperforms SQL Server.

In measuring time performance based on before and after data load time functions, Hive on Spark outperforms by loading to near zero latency in milliseconds compared to SQL Server's seconds/minutes. Though slower in processing data it is to be noted SQL Server does perform well below

100,000 records however once over the threshold, processing time jumps as seen in Fig 7.

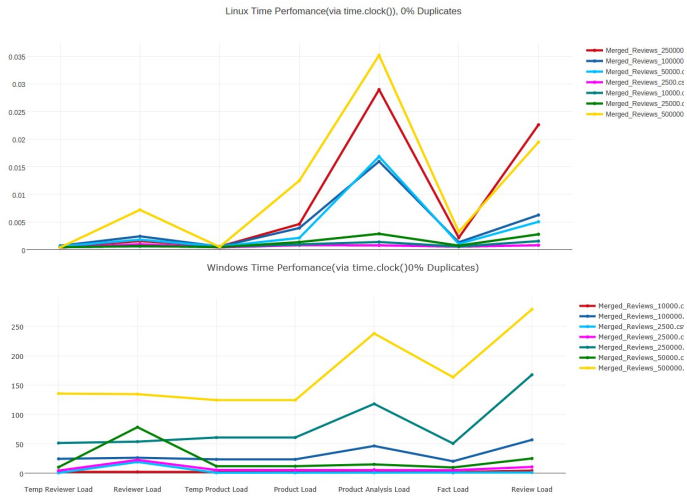

Fig. 7. Hive on Spark (Top Measured in milliseconds.) SQL Server (Bottom Measured in Seconds)

In measuring time in the RT-DW chain, Hive on Spark out performs SQL Server again as SQL Server require more steps to implement, Hive allows overwriting tables where SQLs limitation requires you to send data to a CSV file and to write to an empty table. Another drawback of SQL is that to conduct cube analysis keys are required, this means the deactivation and reactivation of Foreign Key constraints. From observing both Amazon and Cosmos directories, it appears that Windows focus solely on one Python function at a time until completion where Linux can run additional functions depending on the progress of another function.

From multiple test results, processing time is consistent with a minimum variation on Hive on Spark's milliseconds, however as seen in Fig. 8; SQL Server processing time varies based on the computer performance.

Amazon_file	fact_table_load	review_table_load
Merged_Reviews_500000.csv	163.605741406	279.16893666
Merged_Reviews_500000.csv	418.349961964	598.16485341
Merged_Reviews_500000.csv	100.107775606	287.071481313
Merged_Reviews_500000.csv	124.330318821	260.72143564

Fig. 8. Variation of Windows processing time of 500,000 records.

Test: Data Loss performance
Result: SQL Server successfully imports every record, Hive on Spark show data irregularity.

The maximum record count from the seven datasets are 935,000 denormalized rows. Using SQL Server, both the Amazon and Cosmos data warehouse imports each row successfully. In contrast, Hive on Spark shows data irregularities. Two different approaches were used to address this issue.

The first approach originally used the Python package Tempfile where small duplicate data generated after a file is processed over six digits. Tempfile though can cause a more severe data irregularity by doubling the amount of the file. As seen in Fig. 9, It appears this comes into effect after N amount of files are being processed. The alternative approach involved using Windows Temp file approach by creating a CSV file in a directory. This approach was implemented in Windows as Windows NT and later do not support re-opening temp files with the Python package. As seen in Fig. 9, the results show a loss of data in each file load.

duplicate	amazon_file	pre_temp_reviewer	post_temp_reviewer	temp_reviewer_records_
0%	Merged_Reviews_10000.csv	27500	37500	-10000
0%	Merged_Reviews_100000.csv	1887516	1987521	-100005
0%	Merged_Reviews_2500.csv	25000	27500	-2500
0%	Merged_Reviews_25000.csv	0	25000	-25000
0%	Merged_Reviews_250000.csv	1537512	1787516	-250004
0%	Merged_Reviews_50000.csv	1787516	1887516	-100000
0%	Merged_Reviews_500000.csv	37500	1537512	-1500012
12%	Merged_Reviews_100000_12.csv	87500	187503	-100003
12%	Merged_Reviews_10000_12.csv	77500	87500	-10000
12%	Merged_Reviews_250000_12.csv	187503	687511	-500008
12%	Merged_Reviews_25000_12.csv	50000	75000	-25000
12%	Merged_Reviews_2500_12.csv	75000	77500	-2500
12%	Merged_Reviews_500000_12.csv	687511	1687519	-1000008
12%	Merged_Reviews_50000_12.csv	0	50000	-50000
24%	Merged_Reviews_100000_24.csv	0	100003	-100003
Merged_Reviews_10000.csv		26614	35825	-9211
Merged_Reviews_100000.csv		834530	933853	-99323
Merged_Reviews_2500.csv		24571	26614	-2043
Merged_Reviews_25000.csv		0	24571	-24571
Merged_Reviews_250000.csv		535532	785383	-249851
Merged_Reviews_50000.csv		785383	834530	-49147
Merged_Reviews_500000.csv		35825	535532	-499707

Fig. 9. (Top - Tempfile, Severe Data Duplication) — (Bottom - Created CSV tempfile, Data Loss)

Test: Computer Performance (CPU, Memory, Disk)
Result: CPU is pushed to it's limit on Hive on Spark, where Windows starts at max CPU and reduces over time depending on load, Memory and Disk are effected more on Windows compared to Ubuntu.

As Hive on Spark out performs SQL Server in time, the trade off is Data irregularities, The reasoning behind this could be based on CPU performance. Multiple test recordings show a repeat pattern that Hive on Spark pushes CPU performance to the environment's limit for the short period of time. By contrast, SQL Server's CPU performance starts strong but adjusts based on the data load. While the chain is running, Memory and Disk performance on Hive on Spark start and maintain at below 60%. In SQL Server the start point begins at above 50% and can increase above 90% depending on data load.

D. RT-DW Chain: SQL Server VS. Hive on Spark Discussion: Which One?

The findings from Benchmarking based on user concerns[26] have not produced a universal answer on which approach is the best as each evaluation has their

positive and negative results. It is to be noted though that the VMware environments could be the cause of these limitations.

If these limitations are resolved based on better hardware specifications, Hive on Spark would out perform SQL server. However, if hardware is not the reason and it is based on the software, perhaps the approach in tackling Big Data should be re-evaluated. Haisten's [7] viewpoint was taking a certain criteria of data from a stream and place it into a Data Warehouse has merit as it offers less impact on the hardware. With regards the RT-DW chain, perhaps slicing data into smaller segments may be best. This would reduce processing time, pressure on the hardware and prevent data loss. This is supported for when the python tempfile is successfully working with smaller datasets compared to larger datasets as seen in Fig. 10.

```
['Merged_Reviews_25000.csv', '0', '25000', '0',
['Merged_Reviews_2500.csv', '25000', '27500', '2
7500']
['Merged_Reviews_10000.csv', '27500', '37500', '
37500']
['Merged_Reviews_500000.csv', '37500', '537504',
500', '537504']
['Merged_Reviews_250000.csv', '537504', '787508'
8', '537504', '787508']
['Merged_Reviews_50000.csv', '787508', '837508',
8', '787508', '837508']
['Merged_Reviews_100000.csv', '837508', '937513'
```

Merged_Reviews_10000.csv	27500	37500
Merged_Reviews_10000_36.csv	87500	97500
Merged_Reviews_2500.csv	25000	27500
Merged_Reviews_25000.csv	0	25000
Merged_Reviews_25000_36.csv	147500	172500
Merged_Reviews_2500_36.csv	172500	175000
Merged_Reviews_50000.csv	97500	147500
Merged_Reviews_50000_36.csv	37500	87500

Fig. 10. (Top - Data Irregularity after six digits) — (Bottom Successful deployment with smaller data loads.)

VI. Conclusion

The research question being addressed was to convert manual into automated processes to reduce time to allow Data Analysts more time for analysis. Researching current concepts, practices and acknowledging the state the industry, the concept of Real Time Data Warehousing Chain was developed to achieve this.

The findings show that their is great flexibility in the RT-DW chain allowing communication to Data Warehouses, CRM systems and offering Data Mining capabilities. In terms of benchmarking SQL Server and Hive on Spark, their was a trade off based on the 'user concerns' performance parameters. These findings show from a hardware perspective that the user has to have a suitable machine. With regards software, the approach of handling 'Big Data' was reflected upon by proposing alternative methods.

Future work to be implemented in the RT-DW chain should focus on enhancing preparation tools by slicing data into separate CSV file blocks for the RT-DW chain to transfer into the Data Warehouses. This will increase speed, create zero data loss and to reduce computer performance pressure. This can be done by testing one giant file or gathering from a stream as Haisten envisioned. Secondly, RT-DW chain should be placed into different environments such as a high performance computing cluster on a cloud to determine it's effects. Finally it should be noted, The RT-DW chain was developed based on technologies that are based on the time period, different Python packages and languages may provide varied results. It is important to look at what is currently available in the industry to enhance on the current concept.

The RT-DW chain was used to measure Big Data performance but what makes it an efficient tool moving forward is it can be applied to any situation through preparation and the continuous support by coding communities.

VII. Acknowledgments

- Dr. Simon Caton
- Dr. Julian McAuley
- Dr. Peter Burnap

References

[1] J. Bort, "Tech billionaire Michael Dell says 'big data' is the next trillion-dollar tech industry", Business Insider UK, 2015. [Online]. Available: ''http://uk.businessinsider.com/dell-big-data-is-next-trillion-dollars-2015-12?r=US&IR=T''. [Accessed: 11- Jan - 2015].

[2] G. Press, "6 Predictions For Big Data Analytics And Cognitive Computing In 2016", Forbes, 2015. [Online]. Available: http://www.forbes.com/sites/gilpress/2015/12/15/6-predictions-for-big-data-analytics-and-cognitive-computing-in-2016/. [Accessed: 15- Dec- 2015].

[3] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology". ACM SIGMOD Record, 26(1), pp.65-74, 1997.

[4] W. Inmon, "Building the Data Warehouse. 4th ed". Hoboken: John Wiley and Sons, 2005.

[5] M. Breslin, "Data Warehousing Battle of the Giants".Business Intelligence Journal, p.7, 2004.

[6] R. Kimball, "The data warehouse lifecycle toolkit. 2nd ed". Indianapolis, IN: Wiley Publishing, Inc, 2008.

[7] M. Haisten, "The Real-Time Data Warehouse: The next stage in Data Warehouse Evolution". DM Review, 1999.

[8] S. Owen, "What are the differences between batch processing and stream processing systems?", Quora, 2014. [Online]. Available: https://www.quora.com/What-are-the-differences-between-batch-processing-and-stream-processing-systems. [Accessed: 12- Mar- 2016].

[9] P. Nair, "Supply Chain Analytics", CSI Communications, vol. 38, no. 7, pp. 11 - 12, 2014.

[10] B. Sahay and J. Ranjan, "Real time business intelligence in supply chain analytics", Info Mngmnt and Comp Security, vol. 16, no. 1, pp. 28-48, 2008.

[11] J. McAuley, C. Targett, J. Shi, A. van den Hengel, "Image-based recommendations on styles and substitutes" SIGIR, 2015

[12] J. McAuley, R. Pandey, J. Leskovec, "Inferring networks of substitutable and complementary products" Knowledge Discovery and Data Mining, 2015

[13] P. Burnap, O. Rana, M. Williams, W. Housley, A. Edwards, J. Morgan, L. Sloan, and J. Conejero."COSMOS: Towards an Integrated and Scalable Service for Analyzing Social Media on Demand", International Journal of Parallel, Emergent and Distributed Systems, 2014.
[14] R. Kimball. M. Ross., "The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling"; Second Edition. Wiley, 2002.
[15] R. Petley, C. Reid, J. McClean, K. Jones and P. Ruck, "Seizing the information advantage How organisations can unlock value and insight from the information they hold.", A PwC report in conjunction with Iron Mountain., 2015.
[16] S. White, "Study reveals that most companies are failing at big data", CIO, 2015. [Online]. Available: http://www.cio.com/article/3003538/big-data/study-reveals-that-most-companies-are-failing-at-big-data.html. [Accessed: 11- Mar- 2016].
[17] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. Franklin, S. Shenker and I. Stoica, "Fast and Interactive Analytics over Hadoop Data with Spark", 2012.
[18] C. Engle, A. Lupher, R. Xin, M. Zaharia, M. Franklin, S. Shenker and I. Stoica, "Shark: Fast Data Analysis Using Coarse-grained Distributed Memory".
[19] P. Burnap, "COSMOS Legacy Page", Cosmosproject.net, 2014. [Online]. Available: http://www.cosmosproject.net/. [Accessed: 11- Feb- 2016].
[20] A. Rothberg, "What is Benchmarking and Why Is It Important?", CFO Edge, 2013.
[21] S. Liu and R. Kuhn, "Data Loss Prevention", IT Professional, vol. 12, no. 2, pp. 10-13, 2010.
[22] R. Layland, "Data Leak Prevention: Coming Soon To A Business Near You", BUSINESS COMMUNICATIONS REVIEW, pp. 44 - 49, 2007.
[23] J. Porter and J. Rome, "Lessons from a Successful Data Warehouse Implementation", CAUSE/EFFECT, 1995.
[24] "be happy and make others to be happy: Data Warehouse Architecture", Krishnareddyoracleapps.blogspot.ie, 2012. [Online]. Available: http://krishnareddyoracleapps.blogspot.ie/2012/07/data-warehouse-architecture.html. [Accessed: 15- Apr- 2016].
[25] J. McAuley, "SNAP: Web data: Amazon reviews", Snap.stanford.edu, 2016. [Online]. Available: https://snap.stanford.edu/data/web-Amazon.html. [Accessed: 15- Nov- 2015].
[26] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li and B. Qiu, "BigDataBench: a Big Data Benchmark Suite from Internet Services", 2014.
[27] J. Henning, "SPEC CPU2000: Measuring CPU Performance in the New Millennium", Computer, vol. 33, no. 7, pp. 28 - 35, 2000.
[28] Sentiment Analysis, [Online]. Available: http://text-processing.com/api/sentiment/. [Accessed: 03- Feb- 2016].
[29] P. Zikopoulos, "Understanding big data". New York: McGraw-Hill, 2012.