Jeff Nguyen, *jeffn@smu.edu*, Reannan McDaniel, *rmcdaniel@smu.edu*

Performance Measures of Database Types and Transformation Solutions in ETL Pipelines

***Abstract-*Purpose – ETL, Extract Transformation Load process, is a computationally expensive that can require time and costly hardware depending on the size of the data set. We aim to identify database and transformation solutions that are most efficient. Extraction, transformation, and loading is a process used to generate data that is in a usable form for, but not limited to, modeling and reporting purposes. This process can be demanding and may require larger amounts of resources as the volume of data increases.**

**Our goal is to identify storage and data transformation solutions that minimize resource usage. This will be done by comparing performance differences between writing and reading data into and out of MySQL or NoSQL databases in the extract and load phases of ETL in a local environment. Additionally, we want to compare performance differences between Python Pandas and NVIDIA RAPIDS GPU optimized data science software package in the “transform” phase of the ETL process. This sentence is a place holder for our main result.**

**Since the data is normalized, in MySQL we expect the transformation process to work faster than in NoSQL. However, the read write process would take more time in MySQL than NoSQL due to ACID compliant properties required in relational databases.**

# Introduction

Within an organization that utilizes ETL processes, many teams may require data in a particular format that first needs to be extracted from a storage solution and transformed into a format that is usable by various systems and software packages. A typical ETL pipeline involves extraction of data from source databases, transforming the data, then loading the data into a warehouse [1].

The following items are required to create the necessary extractions, transformations, loading, databases and data warehousing; MySQL is an open-source relational database system that is best used with normalized data. Normalization requires data to be ACID compliant (atomicity, consistency, isolation and durability). This allows for data to be valid in the event of a system, hardware or power failure. A NoSQL database system ideal for non-tabular data. An example of a NoSQL database is MongoDB, which is what we use for the purpose of this research. MongoDB uses JSON-document type as a schema and is also open-source. MongoDB can be indexed with primary and secondary indices. ACID is also supported in MongoDB for data validation. MySQL and

MongoDB is used to store data prior to the extraction phase and after the transformation phase. Between these two steps we use Pandas or cuDF to manipulate data.

Pandas is an open-source data manipulation library that can be used in a Python environment. This allows for ease of data munging through the use inbuilt functions, such as deleting or moving columns, adding indices or importing additional data to the data frame. NVIDIA Rapids is a GPU optimized version of Pandas. cuDF is a library within NVIDIA RAPIDS that handles data manipulation. Since cuDF is a GPU optimized library it can process data faster than Pandas due to the large amount of GPU cores preforming work whereas Pandas can only utilize a few dozen CPU cores. A great note is the cuDF syntax is similar to Pandas however cuDF requires a powerful video card to run it.

ETL is expensive due to needing physical server space or fees associated with third-party storage. Transformation is also expensive because it takes time to read data in from a database and takes time to munge, clean and transform the data. Transformation is computationally expensive process because takes time to execute code to modify data into the state that is desired. Changing business requirements may force changes in model features, reporting, or analysis which may require more time to transform the data so that it meets the desired needs. Finally, loading is expensive because writing takes resources. Pointers and indices may require re-mapping to the correct memory locations if reports are being modified. After data is transformed it is loaded into a data lake for reporting, modeling or other analysis. Data in the extraction phase may be in normal form and data in the data lake may not be normal. Non-normal data after the load phase may be beneficial to non-admin users with read-only access as it allows non-admin users to quickly query data that is needed. This briefly summarizes why ETL is expensive – due to physical space, salaries, rent, fees and finding ways to quickly move and transform data quickly while minimizing costs.

Setting up ETL is so labor intensive that the large tech firms have created ETL solutions like Azure, Informatica PowerCenter, Microsoft SQL Server SSIS and Apache suite to overcome the time and labor it takes to complete the process. Although these methods exist, the purpose of this research is not to test these solutions but to show the amount of computational time and resources it takes to perform an ETL process using different combinations of storage and transformation solutions by creating a basic ETL process from scratch.

We are testing MySQL for the extraction and load phases and Pandas and cuDF for the transformation phase. Alternatively, we will test the performance differences for a NoSQL extraction and load with a Pandas and cuDF for transformation. We will measure the time it takes to run read, write and transform operations. Also, the CPU and RAM resources that are needed to run the processes will be captured.

ETL is a process that can perform many steps with large data with a quick turn around on output. This is important for organizations due to time constraints from other software applications, analysis and the urgency for decision making.

Extraction performance for MySQL and NoSQL instances will be compared by capturing CPU, RAM, and disk reading speeds. A set volume of data will be read from the database into a Python data transformation environment. For reading data we expect NoSQL to outperform MySQL. For MySQL we will join tables into a desired format which will require computational resources and time, whereas NoSQL will be in a format that the business determines is efficient. Once the data is loaded into the Python environment we can begin transformation.

For Transformation we are comparing Pandas and cuDF with expectations that cuDF will outperform Pandas because it is a GPU optimized package where a GPU has thousands of cores of processing data; whereas Pandas utilizes the CPU, which at most has a dozen cores for processing data. For the purposes of this research we want to see if there is a difference between cuDF and Pandas for RAM, CPU and time resources. Once the data is transformed it can be loaded into data lakes for use by teams across the enterprise.

The loading process will also take a lot of computational resources because it involves writing data to disk. Once data is transformed in Python environment it will be loaded in a data warehouse or data lake. We expect MySQL or NoSQL write operations to be more computationally and time intensive.

# Data

For this project we were able to procure data from Kelly-Moore Paints Company. Permission was granted by leadership for project use as long as the data was scrubbed of any proprietary information. The raw data came from a daily master file for finished goods and only active SKUs were captured. The final paint SKUs were 406 SKUs. All of the paint SKUs have been changed from paint to women’s boutique clothing. The new SKU methodology contains seven item types and each item type is given a style number as a unique identifier; dress=0234, blouse=0236, slacks=0238, leggings=0240, casual jacket=0242, skirt=0244, cotton tee=0246. Then ten colors were considered with a corresponding color number; red=01, white=02, black=03, blue=04, green=05, yellow=06, purple=07, pink=08, orange=09 and tan=10. Finally, six different petite sizes were generated, P0, P2,P4, P6, P8, and P10. These elements were combined by using the style number, color number and the size to create a new SKU; an example is 023401P6 which represents a red dress in size P6. To further remove proprietary information the site number and site addresses have also been modified.

The data was obtained from two different ERP systems. There are three sets of files, one from a forecasting tool that links to retail Point Of Sale to capture order recommendations for auto-replenishment. This data is generated in Excel. We also have sales data out of the same forecasting tool that is also in Excel and the actual order information from Oracle in a tsv file.

# Related Work

## MySQL and NoSQL databases

As the amount of data being collected increases, data storage and speed need to be considered as well. The traditional Relational Database may no longer be sufficient as firms are starting to collect and analy*ze* data and metadata. We are seeing many large firms move to NoSQL databases, like MongoDB and away from Relational Databases or MySQL. The NoSQL databases can manage more in less time with greater efficiency output [*2*]. When comparing MongoDB to MySQL, the speed at which MongoDB executes is nearly half of MySQL for the same amount of data*.*

MongoDB uses JSON documents as the schema, which is flexible and features the abilities of Ad-hoc queries, file storage and indexing[*3*]. MongoDB also gives the option to insert fields that do not already exist in the document.When generating ad-hoc queries, MongoDB is able to generate a request that contains upper and lower bounds, for example list all store sites that placed an order from July 1 to July 9.The ad-hoc feature also allows for regular expression searches.This is a query that searches for patterns within data.With the increase in the amount of data being captured, file storage is a concern.MongoDB is used as a file storage system with load balancing features which enables MongoDB to use horizontal scaling and keeps the application running in case of hardware malfunction. The file storage feature has other robust features such as GridFS which can divide files into chunks or data into a separate document.GridFS can also access documents by using an open-source webserver called lighttpd which is known for speed, flexibility and cyber-security. One unique feature that MongoDB highlights that MySQL does not is that data fields can be indexed with primary and secondary indices.

MySQL uses SQL where NoSQL does not have a standard query language. MySQL is a tabular, relational database structure that could prove problematic with scaling larger datasets. The ease of use for MySQL is desirable no matter the size of an organization. It is powerful in its ability to use various data packages and computer languages other than SQL. Since MySQL is relational and tabular, the data is able to utilize primary and foreign keys which can sort through duplication records quickly. With the type and amount of data being collected, many companies are moving towards utilizing NoSQL databases to store and analyze data.

For the purpose of this research and taking into consideration these features, we anticipate that the transformation phase will take more time to process in NoSQL compared to MySQL due to data duplication; however the expected results for read/write in the NoSQL database will outperform MySQL in time and processing due to the robust properties in MongoDB.

## Pandas and NVIDIA Rapids

When performance between Pandas and NVIDIA RAPIDS is discussed, the high-level differences between how CPUs and GPUs operate need to be described to understand how these two software packages differ. Central Processing Units, also referred to as CPUs, handle the majority of complex computation needed for a computer system to operate. CPUs generally handle complex tasks and fetch smaller volumes of data from volatile storage quickly [4]. A CPU contain up to a dozen cores where each core can handle several processes (also known as “threads”) at a time. Each thread can handle data manipulation using Pandas, web browsing, or general application execution.

This NVIDIA RAPIDS was released in 2018 and is relatively new. RAPIDS takes advantage of hundreds or thousands of GPU cores to preform repetitive calculations and tasks – compared to consumer CPU’s that have 6 – 12 cores, GPUs can be leveraged to speed up computation. Capacity in the forms of functionality and storage and also time are critical when determining databases management systems. RAPIDS takes deep learning and captures the ETL process and workflow into a more efficient tasks by optimizing time with end-to-end GPU acceleration. For comparison when using an analysis, ETL, training, and inference workflow an NVIDIA DGX-2 server with 16x Tesla V100 GPUs outperforms an AWS r4.2xLarge instances by 10 times [5].

A graphics processing unit (GPU), are processing units specialized for mathematical operations. GPUs can have several thousand cores that handle large volumes of mathematical computation. The volume of cores coupled with fast dedicated volatile memory allow GPUs to process large volumes of data that require simple calculations compared to a CPU. Due these properties GPUs are extensively used in machine learning and artificial intelligence applications because they require large volumes of linear algebra and calculus-based computations. cuDF is a package with the NVIDIA RAPIDS library that was designed to utilize the benefits of GPU architecture to manipulate data. Since there are thousands of threads on a GPU that can handle computation versus a CPU that can handle at most several threads, data processing is much more rapid.

# methodology

In order to compare the performance of each phase of the ETL process, CPU usage, RAM usage, and the amount of time it takes to execute each process was captured. These metrics allow for comparison between the MySQL-Pandas/cuDF ETL process and the NoSQL-Pandas/cuDF ETL process. These metrics show the amount of resources utilized and the rate at which work was performed, where lower CPU and RAM utilization, and lower process execution time is desirable.

Before extraction occurred, data procured from Kelly Moore Paints Company had sensitive and proprietary information removed or renamed. This process was described in the “Data” section of this document. The data pulled from Kelly Moore Paints Company data warehouse was the result of an ETL process and was in a format convenient for reporting and analysis. For the MySQL portion of research, the data needed to be reverse engineered to get it back into its normalized state, emulating what the data would look like before Kelly Moore Paints Company ETL process. For the NoSQL portion of research key-value pairs were generated and populated into MongoDB.

The data procured from Kelly Moore Paints was in .xlsx and .txt formats where each file is daily data. This data was merged into one .csv to allow for ease of ingestion into MySQL. A date column was added to the data by extracting it from each file name using regular expressions. Data was then loaded into MySQL and NoSQL databases. This data served as the initial database.

Extraction performance was measured by reading a set volume of data from the database into Python data frames for data manipulation. One MySQL instance was created and relations were loaded with data and serve as the relational example. For the non-relational model, data was modified and prepared in key-value pairs in a MongoDB or another NoSQL environment. Since our initial data was in a relational form, in order to compare the performance between MySQL and NoSQL, the data needed to be loaded into NoSQL in key-value pairs. The initial loading performance of CPU, RAM, and processing time for each database was captured for later comparison. Extraction will occur in a python environment where data was be read into data frames using pandas and NVIDIA’s RAPIDS cuDF. Read performance metrics were measured by average CPU load, average RAM load, and processing time. It was expected that the NoSQL database would outperform the MySQL database because of weaker concurrency than the ACID properties associated with SQL and that SQL is more computationally expensive to read and write data [2].

Transformation performance comparisons were performed on Python pandas and NVIDIA’s RAPIDS cuDF package. RAPIDS cuDF is a GPU optimized version of pandas and both are nearly identical in syntax. Python pandas is popular for its data structures and data manipulation tools which allow for rapid cleaning and analysis [6]. Data was transformed, cleaned, and manipulated in identical ways for both pandas and RAPIDS. The amount of time it takes to complete data transformation, average CPU loading, and average RAM usage will be used for comparison metrics.

Once the data is transformed in the Python environment it was loaded into a MySQL and NoSQL data warehouse. This operation was handled in Python where the same performance metrics (CPU, RAM, compute time) was be measured and compared.

Two sets of comparisons were conducted for this research: MySQL versus NoSQL read/write performance; and Pandas versus cuDF transformation performance. The CPU, RAM and time metrics were captured from each instance and compared via significance testing.

# results, analysis and discussion

We expect for MySQL instances the reading and writing operations to take more time and computational resources compared to the NoSQL instance due to ACID requirements needed in relational databases. For transformations we expect cuDF to transform data into its desired state much more rapidly compared to Pandas due the core count advantage GPUs have over CPUs. Once performance metrics for each phase of the ETL process is available, significance testing and simple comparisons will be performed to determine which database-transformation combination is the most efficient.

# required resources

The following items are required to create the necessary databases, perform data transformation, and data warehousing.

1. MySQL
2. NoSQL (MongoDB)
3. Python Pandas
4. NVIDIA RAPIDS (cuDF)
5. NVIDIA PascalTM or newer architecture GPU

References

1. H. Homayoun, “An Approach for Testing the Extract-Transform-Load Process in Datawarehouse Systems,” Colorado State University, 2017.
2. A. Abdullah and Q. Zhuge, “From Relational Databases to NoSQL Databases: Performance Evaluation,” Research Journal of Applied Sciences, Engineering and Technology, vol. 11, no. 4, pp. 434–439, Oct. 2015.
3. “MySQL vs NoSQL - Which One Is More Useful (With Infographics),” EDUCBA, 06-Jan-2020. [Online]. Available: https://www.educba.com/mysql-vs-nosql/. [Accessed: 23-Feb-2020].
4. S. Shah, “Do we really need GPU for Deep Learning? - CPU vs GPU,” Medium, 07-Dec-2018. [Online]. Available: https://medium.com/@shachishah.ce/do-we-really-need-gpu-for-deep-learning-47042c02efe2. [Accessed: 22-Feb-2020].
5. S. Prasanna and M. Harris, “RAPIDS Accelerates Data Science End-to-End,” NVIDIA Developer Blog, 25-Jul-2019. [Online]. Available: https://devblogs.nvidia.com/gpu-accelerated-analytics-rapids/. [Accessed: 27-Jan-2020].
6. W. McKinney, “Getting Started with Pandas,” in Python for Data Analysis, 2st ed. Sebastopol, CA: O’Reilly Media, 2017, pp. 123-165.