Benchmarking GPU-Accelerated Databases Against Traditional Databases

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Paul Lee
Spring, 2020

Technical Project Team Members
Paul Lee

Benchmarking GPU-Accelerated Databases Against Traditional Databases

Paul Lee
Computer Science Department
University of Virginia
psl7ds@virginia.edu

***Abstract*—Databases systems play a critical role in the ability to organize and analyze data. In an increasingly data-inundated world, databases must continue evolving to provide sufficient compute power to analyze large datasets. GPU-accelerated databases are a particular type that leverage the highly-parallel nature graphical process units to perform rapid fast queries. In this paper, the performance of a GPU-accelerated database, Kinetica, is benchmarked against PostgreSQL, a popular open-sourced database, for various queries using a standardized database benchmarking framework: TCP-DS. GPU-accelerated databases show promise in database performance for complex, OLAP query workloads.**

I. Introduction

Humans leverage data to make informed decisions constantly. From sensory observations to writing down information, every phenomena in the natural world is recorded in the form of data through various mediums. With the improvements in computer technology, such as processing power and memory systems, computers have become an integral part of data storage through the use of databases. Databases bring unprecedented efficiency by leveraging computational power to organize, process, and manage data at scale and speed. Subsequently, databases have become integral components of software systems.

Databases systems are organized collections of data that reside within a computer system. A database management software system (DBMS) provides an interface to manipulate, retrieve, and manage data within a database. These software systems have inundated markets as companies and institutions use them in managing their data warehouses. The database market size is expected to reach $65.30 billion by 2026 [1].

With the rise in big data, machine learning, artificial intelligence, and internet of things (IoT), data volumes are growing faster at an exponential rate. 90% of the world's data has been created in the past 2 years [2]. These tremendous volumes of data require computer and software systems to perform faster and more efficiently that ever before. Databases are continuously evolving to tackle these challenges. This includes using faster hardware to store and query data and optimized implementations of DBMS.

Different types of databases are used for different applications. Generally, databases fall into two broad categories: online transactional processing (OLTP) and online analytical processing (OLAP). OLTP databases are used for transaction-based operations, such as withdrawing and depositing money into a bank account. OLAP databases are used for analytical workloads, such as analyzing data sources for business intelligence. Naturally, OLAP databases are used for big data workloads, such as powering machine learning data pipelines [3].

Because OLAP databases generally deal with large datasets that can be several terabytes large, these databases must be able to deal with large workloads at speed. Many different implementations of databases use unique ways of optimizing performance through unique software and hardware implementations. For example, MariaDB, a popular open source database, can be configured to store data using a columnar format, which refers to how data is stored column-wise rather than row-wise [4]. This approach allows for faster read speed and better data compression because data is aligned with fewer disk space gaps. However, such an approach results in slower write speeds, making it unsuitable for OLTP workloads. Among various database implementations, conventional approaches use a combination of CPU, memory, and disk.

However, conventional hardware remains compute bound compared to the growth in data volumes. Kinetica is a novel DBMS that is designed to overcome hardware limitations by utilizing graphical processing units (GPUs). GPUs are frequently used in video rendering and other highly parallelizable workloads but also show potential in making database operations faster due to the ability to parallelize database workloads [5]. In this paper, Kinetica's performance benchmarked against PostgreSQL [6], an open-sourced database that is used for similar workloads, through an industry-standard database benchmarking framework [7].

II. Background

A. Kinetica and PostgreSQL

Kinetica is a proprietary database that was originally implemented for a contract with the U.S. army. Initially deployed for various government project, its gained adoption for various projects. In 2016, Kinetica was founded, built on its GPU-accelerated database and has been gaining traction in the database market for data analysis workloads [5].

PostgreSQL is one of the world's most popular open-sourced databases. Originally developed at University of California, Berkeley and released in 2016, PostgreSQL was born from Ingres, an early relational database, as part of an effort to address the limitations of databases in the 1980s. Breakthroughs included being able to define various user types and storing mass amounts of data [6].

Both Kinetica and PostgreSQL share many fundamental attributes. First, they are both relational databases. In other words, they follow the relational model, which organizes data into tables. Within each table are rows, which are synonymous with records. Each table defines a set of columns that describe an entity type of each row so that every row in a given table follow a certain schema. Second, both databases are columnar (if PostgreSQL is configured to use Citus as its storage engine), which refers to how data is stored column-oriented rather than row-oriented. Columnar databases are generally good for larger, complex workloads that are used for analytical workloads [3]. As such, both databases are suited for OLAP workloads. Lastly, both databases can be deployed as in-memory databases. In-memory databases primarily rely on memory for storage rather than disk, which has drastically slower seek and write times. In memory databases are heavily utilized for analytical workloads.

B. TPC-DS

The Transaction Processing Performance Council (TPC) [7] is an international consortium that seeks to establish standards for software transaction performance. Founded in 1985, it has released a set of benchmarks to measure the speed and accuracy of such operations. In terms of a DBMS, a transaction refers to an atomic set of operations on a database that can result in the manipulation or retrieval of data. Transactions are supported through the Structured Query Language (SQL), which provides a language interface for databases to define and execute a transaction. One of the many benchmarks TPC has released is TPC-DS, which is well suited for database benchmarking and has become an industry standard. TPC-DS specifically benchmarks the integrity and performance of a database by providing a set of SQL scripts that run a variety of standard queries against a database. TPC-DS is configurable so that tests are run in a replicable, controlled workload environment that emulates different levels of workload and user stress against a database [7].

III. Testing Experimentation

The following environment and package versions were used to conduct the benchmarks:

- CPU: Intel Xeon Silver 4214
- GPU: Nvidia GeForce RTX 2060
- Memory: 128GB
- Operating system: CentOS 7
- Kinetica v7.0
- PostgreSQL v12.2

These settings were based on computer resources that were available at the time. Package versions of Kinetica and PostgreSQL were based on the latest stable releases at the time of testing. No configurations were made to Kinetica and PostreSQL. Only a single user was made for access purposes for each database.

Part of the TPC-DS benchmark involves generating 99 queries (in the form of SQL scripts) that run various database transactions. There are 4 classes of queries the scripts cover: reporting, ad-hoc, ad-hoc OLAP, data extraction. Reporting refers to answering well-defined questions of a business or operation. The general structure of such queries are static and are repeatedly used, with a minor parameter adjustment being made such as date. Ad-hoc queries refer to impromptu queries that must generally be made to answer a specific business question. Ad-hoc OLAP queries refer to queries made for exploratory and analytical purposes, generally involving more complex and sequential queries. Lastly, data extraction queries involve queries that seek to find relations between data sets to model predictions. Naturally, such queries are heavy on merging tables through joins. Overall, these four general class of queries encompass the following types of transactions:

- Creating tables with a variety of schemas
- Populating tables with rows
- Filtering rows from a table with certain columns
- Merging several tables
- Calculating aggregated values (mean, count, etc.)
- Grouping rows with matching values
- Sorting rows based by column values
- Getting the intersection/union of separate query results

Queries emulate the data generated from a retail store, such as store sales, customers, inventory, and items. The following example is one of the many types of queries that would be generated and ultimately executed through TCP-DS:

```
select top 100 c_customer_id
from customer_total_return ctr1
where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
from customer_total_return ctr2
where ctr1.ctr_store_sk = ctr2.ctr_store_sk)
and s_store_sk = ctr1.ctr_store_sk
and s_state = 'TN'
and ctr1.ctr_customer_sk = c_customer_sk
order by c_customer_id;
```

To prevent databases from hard-coding favorable performance, TPC-DS uses a seed to generate random queries and data for the testing suite. Specifically, the randomized generation is seeded from the timestamp at which the benchmark is run.

TPC-DS also requires an additional configuration parameter called the scale factor, which indicates the raw data size. These values are exponentially scaled and include 100G, 300G, 1000G, 3000G, 10000G, 30000G, and 100000G. For this project, a scale factor of 100G was selected, since the available testing server could fit 100G of data into memory.

After configurations were made and the sample query and data sets were generated, the Kinetica and PostgreSQL servers were restarted to prevent any potential cache that were previously stored. The TPC-DS tests for each database, on after each other, with the server being rebooted between runs to normalize for external variables. The results for each query was stored in csv files, which resided on the testing server's disk. The speed at which each query was executed was recorded as well. After each execution, the validity of the results were measured using the benchmark results generated by TPC-DS. A script was run to find any discrepancies between query results by scanning through the generated csv files and the generated benchmarked results. Validations included checking for the following:

- Matching row counts
- Matching calculations for aggregations
- Matching orderings of query results

The testing script was run 5 times for each database, and the average query execution times for each database by query class was recorded. Between each run, all the created and populated tables within each database was deleted to ensure clean run. The server was also rebooted between each run to clear any cache.

IV. Results

The average execution times of the 5 iterations of the experiment are shown in Figure 1:

	Average Execution Time(s)	
Query Class	**Kinetica**	**PostgreSQL**
Reporting	1.4	5.2
Ad-hoc	1.7	9.8
Ad-hoc OLAP	2.3	23.2
Data extraction	2.1	12.7

Fig. 1. Average execution type by query class

For Kinetica, the standard deviation for reporting, ad-hoc, ad-hoc OLAP, and data extraction queries was 0.24s, 0.07s, 1.3s, and 0.17s respectively. For PostgreSQL, these values were 0.17s, 0.05s, 0.13s, and 0.17s. Both databases scored 100% accuracy in terms of data validity for all trials. Across all query classes, Kinetica outperformed PostgreSQL. For reporting, ad-hoc, ad-hoc OLAP, and data extraction, Kinetica was faster than PostgreSQL by 73.1%, 82.7%, 90.1%, and 83.5%, respectively.

The larger difference in performance between Kinetica and PostgreSQL for OLAP and data extraction queries was expected. These query classes involve joining multiple tables and performing intermediate calculations. Because these tasks are more easily parallelized, GPU-driven optimizations become more pronounced. As such, the more complex and larger the queries, the better competitive performance Kinetica exhibits. In contrast, PostgreSQL uses CPU hardware to conduct its queries. Naturally, however, CPUs are better for sequential calculations than parallel ones. Although CPUs can have multiple cores to multithread calculations, it is nowhere the number found in GPUs, which can have hundreds. As such, the performance differential between Kinetica and PostgreSQL is not as large for simpler queries (such as those found for reporting and ad-hoc queries) because many of these queries involve simpler SELECT queries with few or no intermediate calculations needed.

V. Conclusion

GPU-accelerated databases are designed to outperform traditional databases for complex queries. The results of the tests show that Kinetica drastically outperforms PostgreSQL on various queries. The starkest differences in out performance were seen in ad-hoc OLAP and data extraction queries, which generally encompass more complex, larger queries (characterized by several table joins, several sequences of calculations, and intermediate computations). This was expected behavior, since Kinetica leverages the highly parallel nature of GPUs to speed up SQL join queries, which can utilize multiple threads.

The results of this paper tested basic operational performances of databases in an isolated, non-production environment. In reality, database deployments are complex, as they require tuning and complex configurations for various use cases. In addition, the tests conducted in this paper did not use any sort of cache, an important element in improving database read performance. The testing methodology discussed in this paper can be extended to measure how each database performs in different enviornments and with different configurations.

For companies and institutions with GPUs in their compute infrastructure, Kinetica shows promise of delivering fast computations for business intelligence workloads. There is the consideration of hardware costs for having additional GPUs. However, the sheer efficiency Kinetica delivers for lower costs in hardware (Nvidia GeForce RTX 2060 at $299 [8] vs. Intel Xeon Silver 4214 at $694 [9]) can actually reduce costs. Pairing the same GPUs with machine learning workloads can bring upon synergistic benefits and allows Kinetica to be a powerful data-retrieval layer to feed model training jobs.

VI. Acknowledgement

This project was completed with the help of Professor Upsorn Praphamontripong and UVA Research Computing.

References

[1] Data, R. and. (2019, July 23). Enterprise Database Market To Reach USD 155.50 Billion By 2026: Reports And Data. Retrieved from https://www.globenewswire.com/news-release/2019/07/23/1886552/0/en/Enterprise-Database-Market-To-Reach-USD-155-50-Billion-By-2026-Reports-And-Data.html
[2] deMontigny, J. (2017, August 16). Internet Trends 2017. Retrieved from https://stratabeat.com/internet-trends-2017/
[3] Stojanovski, V., Rami, V. (n.d.). In-Memory OLAP vs. Traditional OLAP. Retrieved from https://it.toolbox.com/blogs/vladimirstojanovski/in-memory-olap-vs-traditional-olap-092908
[4] MariaDB Foundation. (n.d.). Retrieved from https://mariadb.org/
[5] Dilworth, J. (n.d.). About Kinetica - Leadership, History, Openings. Retrieved from https://www.kinetica.com/about/
[6] A Brief History of PostgreSQL. (n.d.). Retrieved from https://www.postgresql.org/docs/8.4/history.html
[7] Homepage V5. (n.d.). Retrieved from http://www.tpc.org/
[8] (n.d.). Retrieved from https://www.nvidia.com/en-us/shop/geforce/gpu/?page=1&limit=9&locale=en-us&category=GPU&gpu=RTX 2060
[9] Intel® Xeon® Silver 4214 Processor (16.5M Cache, 2.20 GHz) Product Specifications. (n.d.). Retrieved from https://ark.intel.com/content/www/us/en/ark/products/193385/intel-xeon-silver-4214-processor-16-5m-cache-2-20-ghz.html