Accelerating geospatial database services with Graphical Processing Units

Bachelor of Science Thesis in Software Engineering and Management

Andreas Fransson
Johan Johansson

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

The Author grants to University of Gothenburg and Chalmers University of Technology the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.
The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let University of Gothenburg and Chalmers University of Technology store the Work electronically and make it accessible on the Internet.

A design science study on database performance
Using modern scenarios to evaluate technology advancements

Andreas. Fransson,
Johan. Johansson.

© Andreas. Fransson, February 2019.
© Johan. Johansson, February 2019.

Supervisor: Michal. Palka
Examiner: Christian. Berger

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
A visualisation of geospatial data generated from a database.
Information provided in the Introduction section on page 1.

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Accelerating geospatial database services with Graphical Processing Units

Andreas Fransson
Computer Science and Engineering
Gothenburg University
Gothenburg, Sweden
Email: andreas.j.fransson@gmail.com

Johan Johansson
Computer Science and Engineering
Gothenburg University
Gothenburg, Sweden
Email: Johan.Johansson.swe@gmail.com

***Abstract*—[Introduction] With the growing need of instant or almost instant processing and retrieval when working on large data-sets we ask ourselves the following question "What impact would switching from a conventional CPU database to a GPU accelerated database have on emergency systems using large geospatial data-sets". [Methodology] We chose to use Design Science and more specifically the method called optimization. [Motivation for the study] We observed a knowledge gap in the field of geospatial analysis regarding use cases associated with emergency systems and with new technological advances both in software and hardware there is a need to reevaluate current systems. [Test results] The result displays that GPU accelerated databases and SPARK databases do not increase the efficiency of processing and retrieving of large geospatial data. [Discussion] Even if we expected the GPU accelerated database to perform better than the standard CPU database we could not see any benefit from switching to a GPU accelerated database or SPARK database. [Conclusion] Our tool we created did enable us to take more informed decisions when making decisions on what database is best for our use case, we did, however, conclude that there is no benefit for us to switch to a SPARK or GPU accelerated database. [Future work] We found several things that would benefit further research in our area, Both in technology and scale.**

***Keywords*-GPU-accelerated database, SPARK database, Windows SQL Server, Geospatial Data, Emergency Systems.**

I. Introduction

Legacy or old database technology struggles to meet the needs of today's new emerging standards concerning data processing, especially concerning large geospatial data-sets. With the growing market consisting of AI, assisted driving and autonomous driving, instant or almost instant response time is demanded from logistics services such as routing and map services. When retrieving or processing large geographical data with today's standard database technology the time consumption forces companies to limit their queries and forces pre and post-process data outside of the database to be able to deliver in time. The advances done in hardware are heavily impacting the design of database systems. Since CPU (Central Processing Unit) cores have a flat speed many-core processors such as GPU's (Graphical Processing Unit) has become a vital alternative for processing the constantly increasing amount of data [1].

Traditional database systems utilize the machines CPU to sequentially execute queries on data stored on conventional hard drives.

The GPU is commonly known as the PC's video card and is commonly used to render graphical images on displays. Using these to accelerate database systems promises up to one hundred times the speed of a conventional CPU-powered database solution in the regards of operations such as analytics. Although this is not the case for basic database operations where input and output speed is the common bottleneck and not processing power [2], [3]. GPU-Accelerated databases, utilizing modern graphics cards to accelerate data retrieval, offloading and analyzing comes with a vast amount of processing power created by working in parallel instead of in sequence as a normal CPU database does. And memory bandwidth provided by in-memory management storing information in both RAM (Random Access Memory) and VRAM (Video RAM). [4].

Emergency systems are for example a "Command and control" software for SOS Alarm letting operators have full awareness of ambulance vehicles and their routes while also updating the information available for the ambulance drivers. These systems classed as "Live Systems" come with a vast requirement list attached including performance, scalability, cost, and reliability [5]. Many companies in the emergency system field are working with large geospatial data-sets in their systems, in this thesis, we will relate to the road network of the entirety of Sweden that results in 2.8 million rows in a database. Each of these 2.8 million rows is a road segment consisting of several geographical points along with other vital information about the road segments see Fig. 1.

There is an issue with the legacy database technology limiting the processing of the large geospatial data-sets in a way that there is no feasible way to make queries that would include most or all of the geographical data needed. Many of the applications that are computing intensive and is processing huge amount of data is often regarded to have an unacceptable lengthy execution time. The only way to give these applications the computing increase it needs is by using parallelism when processing large-scale data [6]. Databases are commonly used and are present in almost every user-based system. With the rise of several commercial and open-source GPU accelerated database solutions and other solutions that

Fig. 1. Spacial Data Visualization Sweden and Gothenburg, Sweden

are made to accelerate the speed of queries on large data-sets, it is of interest to investigate what these solutions are actually capable of in terms of large or huge geospatial data-sets.

Main Research Question:

RQ1: "What impact would switching from a conventional CPU database to a GPU accelerated database have on emergency systems using large geospatial data-sets"

Sub-Questions:

RQ1.1: "What is the impact on efficiency?"

RQ1.2: "What is the impact on scalability?"

The main research question aims to reveal what type of database technology is most applicable for emergency service systems. As previously stated there are a number of technical solutions each with its own benefits, the question is if those benefits are applicable to our use case. For this specific use case, there are some requirements reflected as sub-questions to extend the credibility of the main research question and to evaluate the different solutions in more ways than just one.

Using software design we aim to contribute scientifically by exploring the viability of GPU-accelerated databases in emergency service systems that are working with large geospatial data-sets. Using a reference system provided by a company working with geospatial analytics system we will be creating a generalized testing system. This system will be designed with the same technology in mind that is in place today at the company we are cooperating with, we will ingest and output similar data as the reference system is ingesting and outputting today but trimmed down to be objective and generalized. The system will be able to switch the database part of the system giving us the ability to use GPU-Accelerated database technology or SPARK (A in-memory CPU database system for analytics that uses parallelism across multiple machines) instead of the currently used CPU database technology. [7]. We also aim to contribute with a detailed description of how our benchmarking is planned, prepared, conducted, and evaluated. The technical contribution would consist of the development of a testing pipeline with interchangeable databases that will be part of an Azure cloud architecture.

In the methodology section, we discuss and present the methodology we have chosen for our paper. In the Motivation section, we briefly explain the reasoning behind this paper and why this research is needed. In the Related work section, we go through the related work that has been done concerning CPU-acceleration, related work concerning GPU databases and work that has been done using the NVIDIA CUDA.

II. Methodology

For this paper we will be using Design Science as our methodology, we decided to use Design Science as our research aims to solve a real problem and Design Science entails applied research. Design Science contains an Analytic Design Evaluation method called Optimization, this directly correlates to our goal of researching possible improvements on the reference system seen in Fig. 6. We will follow the steps from DSRM (Design Science Research Methodology) [8] since it should contribute with a clear structure of our methodology. From the start, we already have a predefined system that we during this thesis aim to optimize in terms of efficiency. We will conduct tests on the system with the same data and queries but with the different database solutions, the data we will extract from these tests is the time it takes to execute the queries. With this data we will use these custom metrics to evaluate the current system:

- Speed increase (time in seconds comparing new technologies with the baseline system)
- Scalability of technology (time in seconds comparing different hardware setups but the same technology and data-set size)
- Scalability of data (time in seconds comparing the same technology but with different sizes of data-sets)

III. Motivation for the study

There is a knowledge gap present in the field of geospatial analytics for the use cases concerning command and control systems in the emergency system setting. Can we rely on generic theoretical research of technology such as the GPU-accelerated technologies to apply in every use case?

The continuous advancements in technology open up for the creation of more powerful software and hardware but also increasing the data usage exponentially while retaining at least the same requirements as earlier when it comes to data throughput, accessibility, and speed. This creates the demand for reevaluation of the current status of systems, how they are built up and what technologies should be used.

In addition to the two former motivations, we also got presented the possibility to do research on real data, real systems and having access to scenarios, requirements and a baseline system. The data that we have been given access to is real-world data on a large scale without using replication or any other synthetic fabrication of data-sets.

IV. Related work

Our main topic surrounds the GPU-Acceleration of databases and what impact this has on a database in comparison with other techniques to do data storage and processing. This section will reflect on related work in the GPU-Accelerated data analytic field and how our research relates

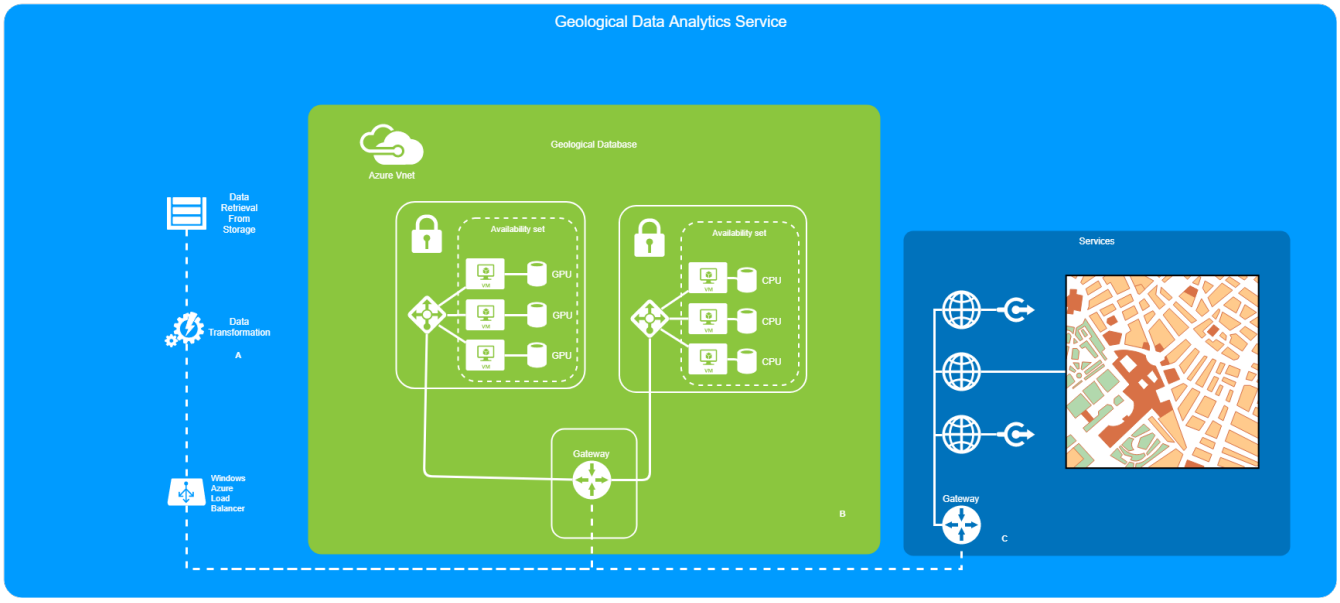

Fig. 2. System Overview

and extends into another what we can assume new field of study. The most similar and also most recent research that has been done in our area is the paper written by Evangelia Sitaridi [1], this Ph.D. thesis goes into depth about comparing CPU processing and GPU processing. It also covers large data-set processing in GPU accelerated databases. Much of the recent work is based on the NVIDIA CUDA API (What NVIDIA themselves call a ”parallel computing platform” which enables use of NVIDIA GPU’s for general computing) and most of the work that has been done in our field is concerning GPU-accelerating geospatial analysis using this new technology [1], [2], [6], [9], [10]. These papers also compare how GPU implementations compare to CPU implementations, they are similar to what we are doing but in other fields and with the exception that we are working with databases while all similar work we have found focuses on standalone software running on GPU’s and how this is beneficial. On the database side, there is also a few implementations of GPU accelerated databases replacing old CPU systems [3], [11].

V. Artifact

Our use case consists of 3 major parts depicted as A, B and C components in Fig. 2, The artifact in question will consist of steps A and B while C is present in a black box format to fulfill the use case. A starts the procedure by doing automatic data retrieval from a server (Open street maps, using their open API) or by manually submitting a request to Lastkajen (Vägverket’s open street database), the data is in a raw geospatial format and supplies those to a custom conversion tool. This step prepares the data to be compatible with common database systems and sends the converted data to component B. Component B requires CSV (Comma Separated Values) format to be read in, in addition to being geospatially formatted with WKT data it also requires a header being present identifying each value for each row with support for geospatial data types. For our artifact, we will need to support most common geospatial data types for connecting geospatial coordinates in the form of LINESTRING and POINT types. A point is a 0-dimensional object representing a single location, there are multiple versions but the geographical version used by us is a traditional longitude latitude spacial point. The LINESTRING type is a one-dimensional sequence of POINTS in a geographical space that creates line segments connecting the POINTS. Component B is the database and is located in the cloud and consists of at least one node (Server, Virtual Machine, Container or another medium in a cloud environment). Although it is not restricted to a single node, as it is a dynamic database cluster that could consist of multiple nodes sharing data or individually responsible for data-sets. Sharing in this context referring to the replication of data-sets on multiple hosts (shards) for redundancy and the individual responsibility being using hosts and tables in single responsibility pattern storing either geographical regions on specified hosts or specific data points separately to maximize distribution in load. This is the main concern for our artifact performance and we have multiple configurations available. In Fig. 2 part B we can see both a CPU and GPU solutions connected as both will be independently tested. This is where we tackle the core of the problem. After the database layer, this use case connects outwards to C which is the various services using geospatial data-sets such as map services. These services are not tested by our artifact as they are not included in the investigation regarding the database optimization for geospatial types but represented as black box realization of a use case.

The test system will allow us to conduct our tests inside a

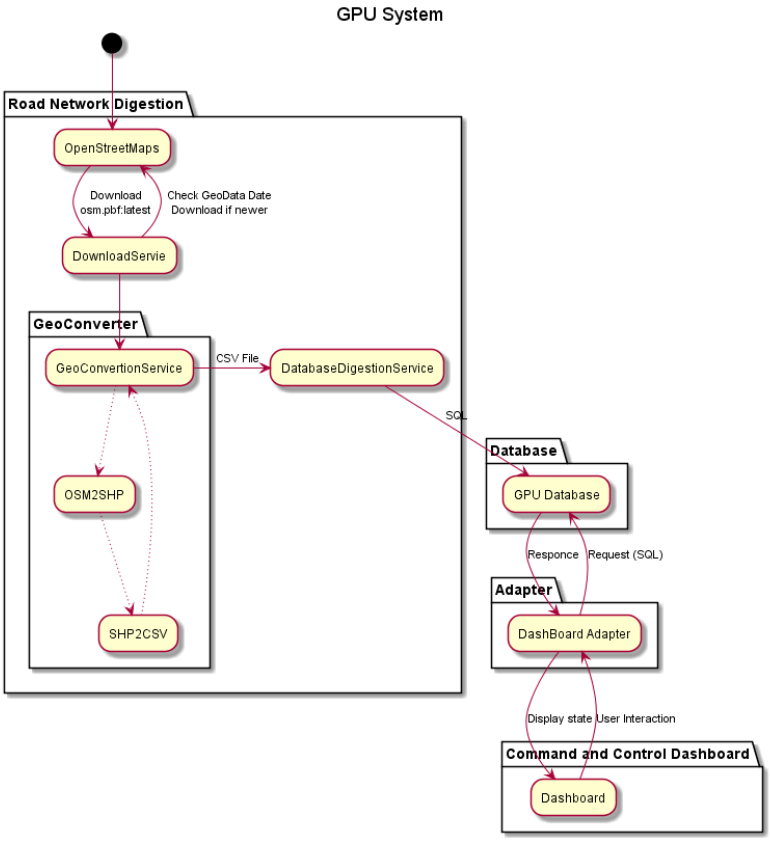

Fig. 3. GPU System Flow

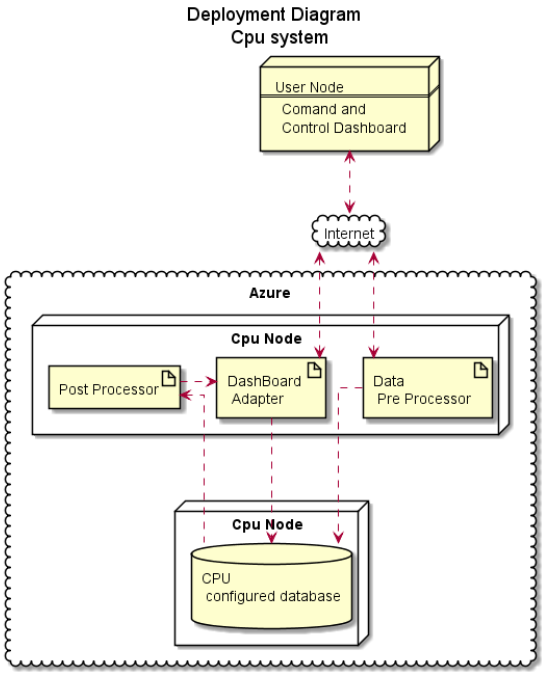

Fig. 4. Reference architecture

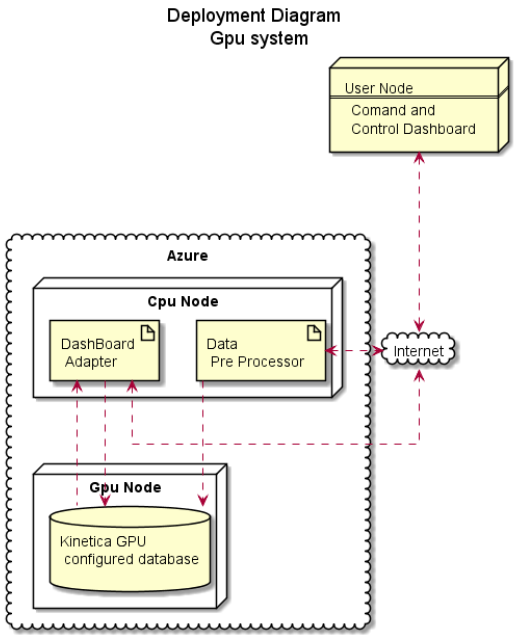

Fig. 5. Reference architecture

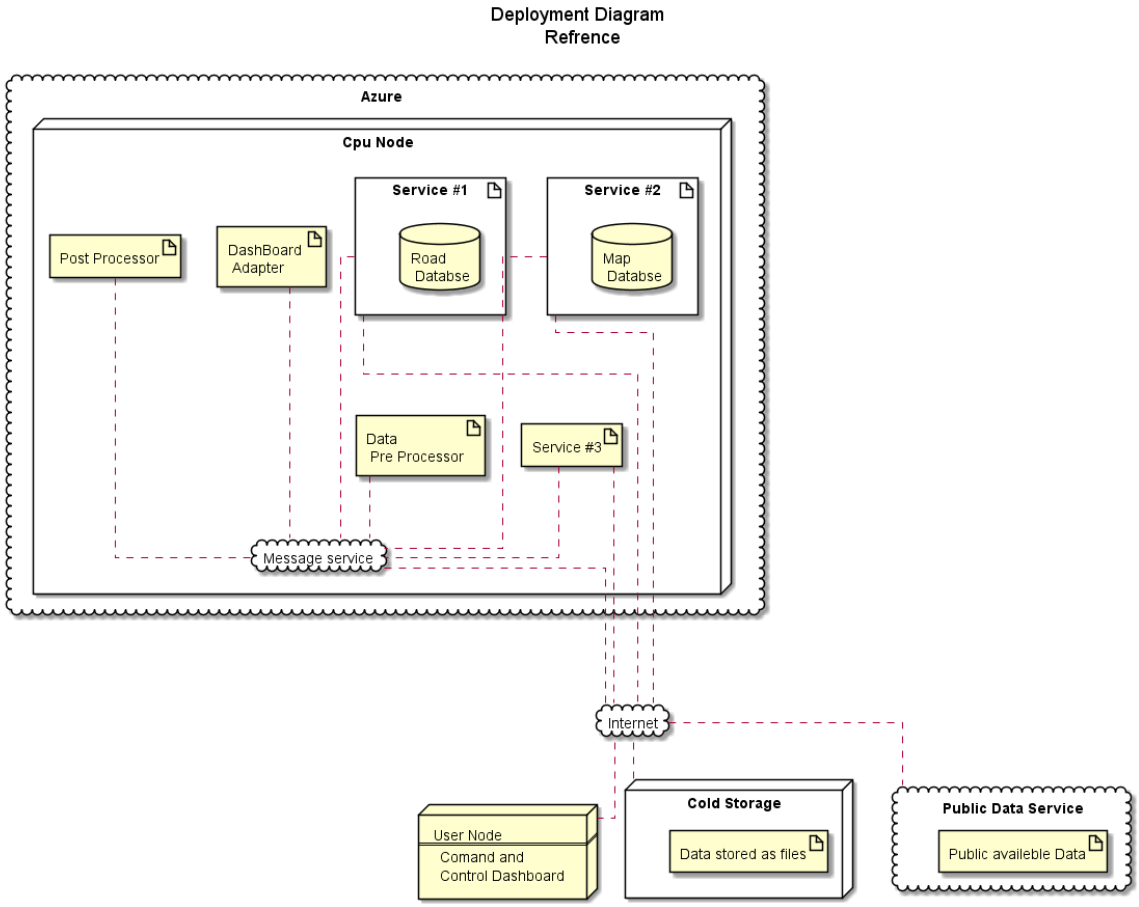

Fig. 6. Reference architecture

pipeline and without affecting any other part of the pipeline switch out the database part (GPU, CPU or SPARK) The system was developed inside an Azure virtual machine so all of the hardware variables could be controlled more efficiently.

A. CPU system

The flow of the two CPU systems (Windows SQL Server and SPARK) is the following:

1) Data is located online.
2) Data is downloaded to cloud host drive.
3) Data is converted and filtered from OSM to SHP format.
4) Data is converted from SHP to CSV format.
5) Data is digested into the database.
6) Data is extracted from the database with SQL.
7) Software executes and processes the data extracted from the database.
8) Data is sent to the front-end of the system to be displayed.

This system is a stripped down version of a real system we have observed within the company we work together with and is minimized to only contain what we need to conduct our tests and also to be more generalized.

B. CPU system Architecture

Using our reference architecture seen in Fig. 4. This is a stripped down from all non-essentials version of a deployed system. Created for testing and made into a simplified test architecture for both CPU and GPU. They do differ in detail but are essentially made to be equal and to be modeled after their respective technologies benefits or restrictions. The CPU system is bound to do post-processing on data retrieval after SQL queries.

C. GPU system flow

The flow of the GPU baseline system is the following:

1) Data is located online.
2) Data is downloaded to cloud host drive.
3) Data is converted and filtered from OSM to SHP format.
4) Data is converted from SHP to CSV format.
5) Data is digested into the database.
6) We instruct the database to execute a series of queries on the data in the database.
7) The response is sent to the dashboard adapter.
8) Data is sent to the front-end of the system to be displayed.

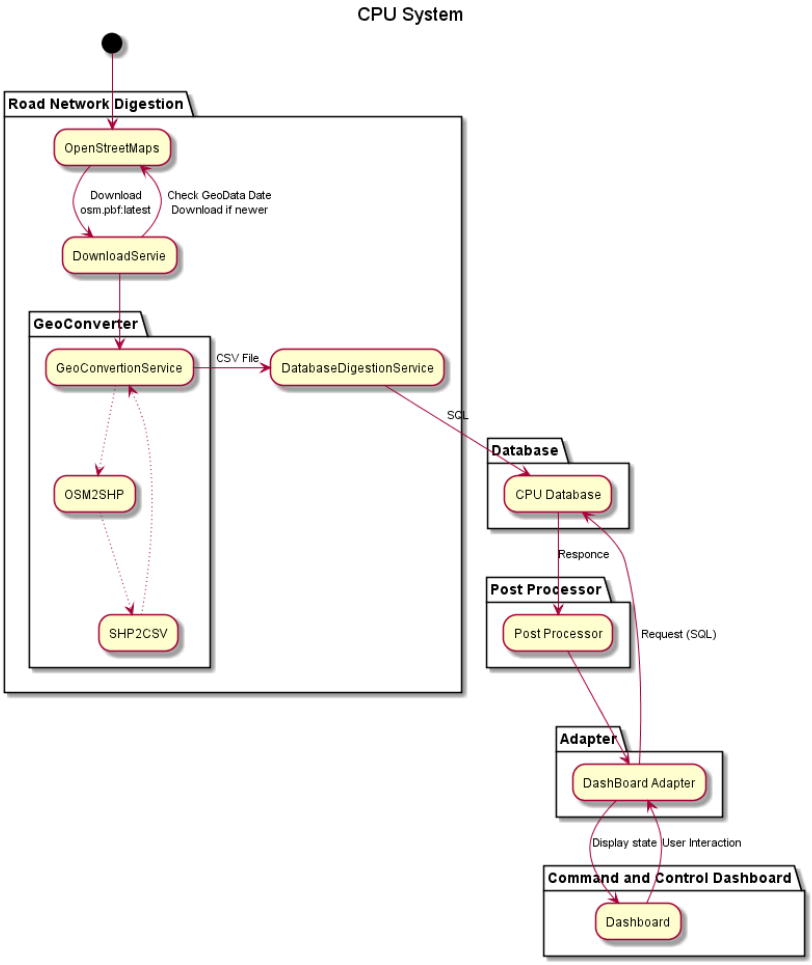

Fig. 7. CPU System Flow

Since both the starting point and the ending point of this system is identical to the baseline system replacing the old system becomes possible.

D. GPU system architecture

The GPU architecture allows communication between the dashboard adapter and the database without post-processing. There is a requirement for a middleman component to translate user input to queries but no processing of data outside the database domain.

The databases implemented in the artifact and researched are Windows SQL Server, SPARK and Kinetica. Both Windows SQL Server and SPARK are referred to as CPU databases as they do not use GPU acceleration and rely on the CPU to perform all operations including but not restricted to data management in form of data-set updates, inserts, retrieval and also compute tasks like aggregate functions and UDF's (User Defined Function). SPARK, on the other hand, does not support the same amount of data types as Windows SQL Server and Kinetica, who both support the geospatial data types used in this research. The approach is to store the data as basic data types and in the database do conversions and computations when needed. Kinetica uses both CPU's and GPU's to perform its different operations. Kinetica uses the GPU to be able to hot-swap commonly accessed data points and even operations to be able to perform the most usual requests in an optimized manner. This while also supplying full support for extending the core database.

VI. Test results

Both of our tests were constructed to be general in a sense of basic operations like retrieval, updates, inserts and to be specific for the geospatial data types in question like aggregation statistics on the different geometrical data types. These tests are included in the tool as a reference example on how to measure differences between the databases but on the same level. The tests were run concurrently four instances at a time over exactly a hundred times. All the individual data stored separately in files and then gathered and averaged out to form a more accurate test result. To demonstrate this we used the road network databases gathered from the Swedish road authorities (See a visual representation in Fig.

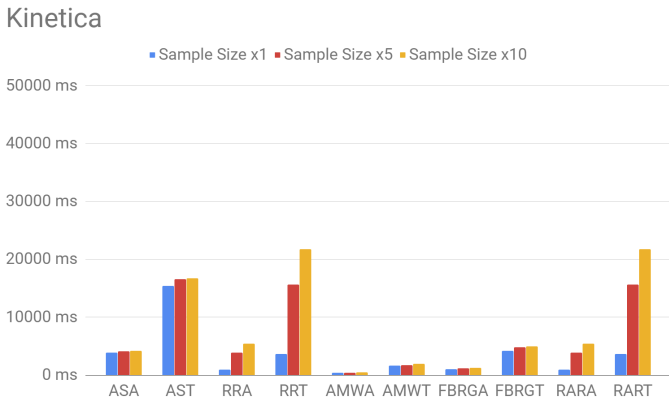

Fig. 8. Kinetica Test Results

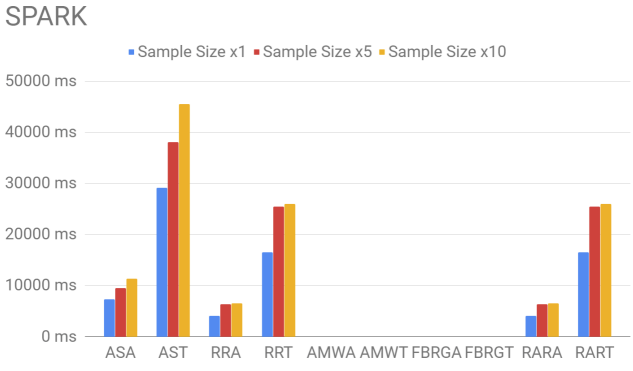

Fig. 9. SPARK Test Results

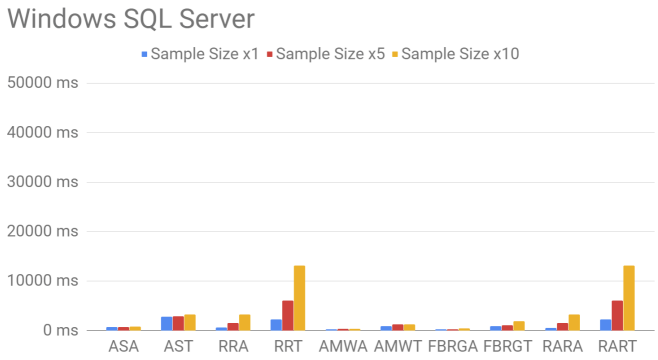

Fig. 10. Windows SQL Server Test Results

1) and sliced it up in full size ”Sample size x10” half-size ”Sample size x5” and one-tenth size of Sweden’s total road data ”Sample Size x1” see column labels in Fig. 10, Fig. 8, Fig. 9. All the test were run with our test strategy previously mentioned using concurrency and multiplicity to average out to a simulation of a use case. In this case for digesting and retrieving data either by size or by a geospatial function. The result in question is showing a type of query and size in accordance to the amount of time taken for the query to complete. Showing clearly in the sample test runs is a pattern where SPARK mostly is held back by slow access time to data making a trend of much slower performance than its counterparts no matter sample size. Also what you can see in Fig. 9 on column groups AMWA, AMWT, FBRGA, and FBRGT is that the SPARK database has no data for the geospatial tests, as mentioned in the artifact section this is because of the lack of implementation and would require us to rely on third-party plugins. Moving over to Kinetica we can see a trend that after the initial data multiplication causing a hit to the performance by more than tripling the time to query completion. But the gap between the five and ten times sample size is significantly reduced, pointing towards better management of bigger data-sets. As for Windows SQL server, the performance impact between the sample sizes is more linear.

Mapping of terminology used in Fig. 10, Fig. 8, Fig. 9.

1) Column Representations:
   - Sample Size x1:
     One-Tenth Of Sweden’s Full Road Network.
   - Sample Size x5:
     One-Fifth Of Sweden’s Full Road Network.
   - Sample Size x10:
     Sweden’s Full Road Network.

2) Test name conversion:
   - ASA: aggregationStatsAverage
   - AST: aggregationStatsTotal
   - RRA: retrieveRecordsAverage
   - RRT: retrieveRecordsTotal
   - AMWA: aggregateMinWKTAverage
   - AMWT: aggregateMinWKTTotal
   - FBRGA: filterByRadiusGeometryAverage
   - FBRGT: filterByRadiusGeometryTotal
   - RARA: retrieveAllRecordsAverage
   - RART: retrieveAllRecordsTotal

VII. Discussion

What impact would switching from a conventional CPU database to a GPU accelerated database have on emergency systems using large geospatial data-sets?

When switching from the current standard of using conventional CPU databases with geospatial data-sets to GPU accelerated or SPARK solutions you might expect that there would be a big improvement working with big data-sets. However, our results indicate that with the amount of data we used in our tests (2.8 million entry’s or road segments) the conventional CPU database was more efficient than both our GPU accelerated database and our SPARK database. This could indicate that the size of our data-set is not substantial enough to benefit from GPU accelerated or in-memory databases like SPARK thus implying that for our use it would

be more efficient to use conventional CPU databases that do not use any in-memory or GPU acceleration. Regarding SPARK the only indication that we can see is that it is worse than both the CPU database and the GPU accelerated database in the regards of our use case.

Regarding scalability, we notice in our results that when running the tests on bigger data-sets GPU experiences a more linear and expected difference in time comparing different data-sets, while GPU accelerated database displays more of a flattening our curve indicating that given a big enough data-set the GPU accelerated database becomes more efficient than the conventional CPU database. When observing SPARK we can also identify this curve indicating that even though it is much slower than both other technologies, given big enough data-sets it could be more efficient than the conventional CPU-bound database.

To be able to conclude where the line is drawn where the data-set is large enough so that we would benefit from a GPU database instead of a CPU database new research could be conducted using the system that we have developed but ingesting bigger data-sets into the pipeline.

Other research that has been done concerning geospatial data processing and Analysis [12] [6] shows that using GPU's does improve performance, our results indicate that given a larger data-set the GPU accelerated database would have better efficiency would corroborate these results.

VIII. Threats to validity

Most of the issues we ran into that we deem to be either hindering our research or affecting the results of our thesis are external factors. One problem we ran into was finding database solutions using either GPU or SPARK databases that also supported the data types we were working with, this limited us and in the end, we only had the possibility to use Kinetica as GPU database and SPARK as cluster CPU database, this was an issue that made our results harder to generalize. Another problem we noticed is several external factors could possibly affect our test results, factors such as the Azure virtual machine hardware not being fully consistent, network connections interacting with the virtual machine, heavy traffic to and from Azure might lead to inconsistencies in test results and the location of the hardware geographically also caused some delay. We have put in the following effort to mitigate these external factors:

- We conducted our tests during hours that the servers were not heavily utilized.
- All of the tests with the different databases were run on the same geographical Azure server park location.
- We chose the same or equal hardware on all of the database solutions.

During the creation of our artifact we came across the possibility of integrating all of the functionality of the reference system directly into the different databases we ran tests on, using different methods such as UDF's and Python scripts. Taking this route indicated it would result in a more efficient system than what we ended up with in the end. The reason why we did not take this route is that every system would have to have its own unique implementation and would require a complete factorization for each database. In addition, it would also be almost impossible to generalize and almost impossible to compare the different technology solutions with each other. Since we wanted to end up with a testing pipeline with interchangeable databases so we could compare the results with as few changed factors as possible we chose to disregard this option.

Data-sets used in the research are proprietary, as mentioned earlier the original state of this data is open source and free access from Vägverket or Open Street Maps. The unique features inherent to the proprietary data-sets are not used in any of the tests so does not affect measurement other than in size. This creates a situation were the test results presented in this paper are not reproducible to full extent.

IX. Conclusion

The use of tools like this enables us to perform better assumptions when making decisions. This tool is directed towards software entities whether it is private or public to be able to prototype simple use cases mirrored in multiple databases to later be able to make that now informed decision on what platform to bind into their needs. Because databases while sharing a common term and function are in fact very specialized and affected by circumstances it is vital to make that decision on what fits your needs with that in mind. The conclusion we met when testing our use case and parameters were that there were no benefits to change database from the reference CPU Windows SQL Server database to an in-memory database (SPARK) or GPU accelerated and in-memory database (Kinetica).

X. Future work

Throughout the research period we encountered multiple obstacles in different forms, and we made a habit of documenting our findings to support further studies in the future were circumstances might have changed. Included in this is the following:

1) Dockerization utilizing cross-platform containers.
2) The current state of GPU analytic databases for analytics.
3) UDF "User Defined Functions" usage.

A. Dockerization utilizing cross-platform containers

While developing the artifact we wanted to stick to an easy to deploy anywhere solution, we started adopting Docker. Because the reference system had just been ported to Docker this seemed like an appropriate time to follow that path and gain some simplicity in managing the test system architectures. These components are however bound to Windows containers and all our GPU database solutions were Linux only systems. But as Windows now comes with a minimum Linux image and LCOW Docker can now run both Linux and Windows containers, simultaneously. We set up the foundation, and it proved to be compatible with Docker-compose. An issue first

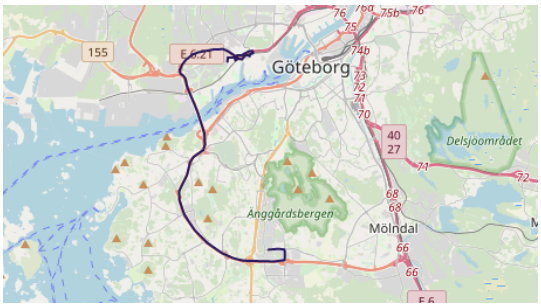

Fig. 11. UDF A* Resulting Path

arose when we tried to deploy in the cloud. Azure and AWS also both have nested vitalization switched off and because every user instance in Azure is a virtual one we cannot start our Linux Docker containers. Although this is and will not always be the case. Azure has just started allowing this by activating necessary components in their vitalization software for some of its heavy CPU bound hardware but not for the instances needed for our research. This might change in the future as less overhead is supplied when you can containerize components in a system and newer hardware might allow newer technologies.

B. The current state of GPU analytic databases for analytics

There are two highly recommended native GPU-accelerated databases (that is not an add-on plugin for an older already existing database but actually created with GPU in mind) when it comes to real-time analytics in the cloud, MapD, and Kinetica. We have been in contact with both these database providers extensively and each offer different strengths and weaknesses. Unfortunately, at the end of the implementation of this project, MapD does not support the geospatial data types needed for our data-sets. When talking with their development team a patch to allow this is en route for beta-deployment after summer 2018.

C. UDF "User Defined Function" usage.

As an investigation into geospatial data analytics, we explored the usage of UDF's and "offloading" complex computing functionality to the database. We tested this by implementing multiple samples into our artifact. The samples in question are compatible with the sample data also included in the artifact in the form of a geospatial road network. The computation is in form of an A* [13] and a Dijkstra's path-finding algorithm [14] written in Python. Because Kinetica supports on the fly UDF's which functionality was also implemented to be able to via our test system update or create new UDF's and upload to a host. The samples work by taking two parameters in form of two geospatial points present in the road network. The UDF will then calculate the shortest path between these two points in the database. The result will create a new table in the database consisting of one LINESTRING representing the optimal path and is visualizable in the database itself, see Fig. 11. UDF's Are supported in all three databases, but in different varieties. SQL Server has its own TRANSACT-SQL language to write UDF's in, SPARK has booth Java and Python but in this case, does not support the geospatial data types to be able to run the path-finding UDF provided. There could be a case for using third-party plugins or manual data conversion and libraries to be able to implement a similar UDF to the one we provided along with our artifact. Kinetica supports both Python and Java with the addition of access to CUDA functionality when running in GPU mode(Required to access GPU functionality).

D. Working with bigger data-sets

As previously mentioned in the discussion continuously ingesting bigger and bigger data-sets into our system should provide us an answer to the question "at what amount of data does the GPU or SPARK database become more efficient than the CPU database". This could also give us an indication of what technology scales best working with huge data-sets.

References

[1] E. Sitaridi. (2018) Gpu-acceleration of in-memory data analytics. [Online]. Available: http://www.cs.columbia.edu/~eva/gpu_thesis.pdf[Accessed1Mar.2018]

[2] S. Breß, F. Beier, H. Rauhe, K. Sattler, E. Schallehn, and G. Saake, "Efficient co-processor utilization in database query processing," *Information Systems*, vol. 38, no. 8, pp. 1084–1096, 2013.

[3] (2018) Go.mapd.com. [Online]. Available: http://go.mapd.com/rs/116-GLR-105/images/MapD%20Technical%20Whitepaper%20Summer%202016.pdf[Accessed29Mar.2018]

[4] P. Bakkum and K. Skadron. Accelerating sql database operations on a gpu with cuda. [Online]. Available: https://www.cs.virginia.edu/~skadron/Papers/bakkum_sqlite_gpgpu10.pdf[Accessed29Mar.2018]

[5] M. Shukla and J. Asundi, "Considering emergency and disaster management systems from a software architecture perspective," *International Journal of System of Systems Engineering*, vol. 3, no. 2, 2012.

[6] N. Stojanovic, "High performance processing and analysis of geospatial data using cuda on gpu," *Advances in Electrical and Computer Engineering*, vol. 14, no. 4, pp. 109–114, 2014.

[7] M. Zaharia, M. Chowdhury, M. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. [Online]. Available: https://www.usenix.org/legacy/event/hotcloud10/tech/full_papers/[Accessed26Mar.2018]

[8] A. Nilsson. (2018) Design research. [Online]. Available: https://gul.gu.se/courseId/82758/node.do?id=40469641&ts=1518448734215&u=1940967794[Accessed1Mar.2018]

[9] Y. Xia and L. Kuang, "Accelerating geospatial analysis on gpus using cuda," *Journal of Zhejiang University*, vol. 12, no. 12, pp. 990–999, 2011.

[10] J. Li, M. Finn, and M. Blanco Castano, "A lightweight cuda-based parallel map reprojection method for raster datasets of continental to global extent," *ISPRS International Journal of Geo-Information*, vol. 6, no. 4, p. 92, 2017.

[11] (2018) How gpu-powered analytics improves mail delivery for usps. [Online]. Available: https://www.datanami.com/2016/07/25/gpu-powered-analytics-improves-mail-delivery-usps/[Accessed1Apr.2018]

[12] J. Zhang and D. Wang. (2014) High-performance zonal histogramming on large-scale geospatial rasters using gpus and gpu-accelerated clusters. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6969489

[13] R. Belwariar. A* search algorithm. [Online]. Available: https://brilliant.org/wiki/dijkstras-short-path-finder/

[14] T. Abiy, H. Pang, W. Williams, J. Khim, and E. Ross. Dijkstra's shortest path algorithm. [Online]. Available: https://www.geeksforgeeks.org/a-search-algorithm/