Pandemic Symptoms Real-Time Ranking Platform

Conference Paper · November 2021

DOI: 10.1109/TELFOR52709.2021.9653200

4 authors, including:

Aleksandar Ivanovski
Ss. Cyril and Methodius University in Skopje
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Vladimir Zdraveski
Ss. Cyril and Methodius University in Skopje
22 PUBLICATIONS **101** CITATIONS
SEE PROFILE

Pandemic Symptoms Real-Time Ranking Platform

Aleksandar Ivanovski, Marjan Gusev, Vladimir Zdraveski, and Jesper Aasa.

Abstract—COVID-19 takes an increasing share of everyday life and imposes the need for an exploratory data analysis executed by both, professionals and the general public. The primary focus of this paper is designing and implementing a system for processing the vast amount of case data available to obtain overall statistics for symptoms and rank them in real-time. Processing the current data and providing a mechanism to process new data generated in real-time from diverse and many sources is one of the current challenges. Our solution to tackle the challenge is to execute the processing in a massively parallel way enabled by CUDA along with principles and constructs for efficient parallel programming, which are eminent due to the volume and velocity of data, thus, checking the validity of a research question is it possible to process Covid-19 big data challenges more efficiently with GPU-based parallel constructs.

Index Terms—COVID-19, big data, real-time, parallel processing, symptoms ranking, CUDA

1 Introduction

After more than a year from the start of the global COVID-19 pandemic and the currently growing amount of new mutations, the need for efficient tracking and exploratory data analysis has become part of the daily workflow of many medical personnel. Currently, the amount of COVID-19 related data is on the rise and may be classified as Big Data. One of the common tasks for the medical personnel is discovering clusters among the cases, those clusters can be in terms of physical distance and contacts, symptoms, clinical conditions, disease and vaccine-related after-effects, virus mutations, and even now after a year - long-term effects.

On the other side, all of the COVID-19 cases and data can be modeled and represented within vector spaces, and unsupervised learning algorithms can be applied to achieve the data analysis tasks. From this statement, a key assertion is made i.e. regardless of the pandemics or disease, nearly all symptoms data can be encoded within vector spaces.

In this paper, we focus on providing real-time symptoms and vaccine side effects ranking and tracking. To process the huge amount of data by unsupervised learning algorithms, we are going to leverage the massive parallelism enabled by the CUDA [1] programming framework. The benefits from such a system could be leveraged by both medical personnel and the general public.

The COVID-19 case data are a combination of textual and numerical features, describing the patients' clinical case, including symptoms and a variety of other parameters (such as WBC, LYM, NEU, AST, ALT, CRP, D-dimmer, etc.). Currently, there are $\approx$ 160M recorded cases, with a growing rate of 350,000 cases per day, equivalent to $\approx$ 243 cases each minute, suggesting that a massively parallel approach is a necessity. The research question in this paper is to analyze if COVID-19 big data can be processed more efficiently with the usage of GPU.

- *A. Ivanovski, M. Gusev, V. Zdraveski are with the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, North Macedonia.*
  E-mail: aleksandar.ivanovski.1@students.finki.ukim.mk, {marjan.gushev, vladimir.zdraveski}@finki.ukim.mk
- *J. Aasa is with Vitala Health, Stockholm, Sweeden.*
  E-mail: jesper@vitala.health

2 Related Work

Many researchers are working in the field making it a hot research topic. A data-driven real-time risk assessment [2] approach uses regression trees and wavelet transformations, scalability of such solution and the enabling of massive parallelism were not discussed. Fast decision-making model [3] is based on a GP-GPU approach to differentiate between influenza and COVID-19 cases.

One of the earliest approaches to realize clustering on CUDA-enabled architecture was presented by Shalom et al. [4] concluding that text clustering with speed gains starting at 15 times faster than traditional approaches. However, the accuracy was questionable. A similar approach was analyzed in combination with dimensionality reduction in [5], and a CUDA approach with a primary focus in object segmentation [6].

Cui et al. achieved notable results in [7], leveraging CUDA for highly dimensional data clustering. Acceleration provided with massive GPU parallelism of the T-SNE dimensionality reduction in combination with a model for vector embeddings was realized by Chan et al. [8]. The same research group provided the application of the accelerated model for large datasets and human-in-the-loop based tasks in [9].

A novel CUDA enabled clustering algorithm with general-purpose usage was presented by Lacerda et al. [10], which has the potential for real-time applications, such as clustering news articles in aggregators. Zhang et al. [11] presented a general-purpose speedup in text-mining with the main focus on small to medium-sized data sets.

978-1-6654-2585-8/21/$31.00 ©2021 IEEE

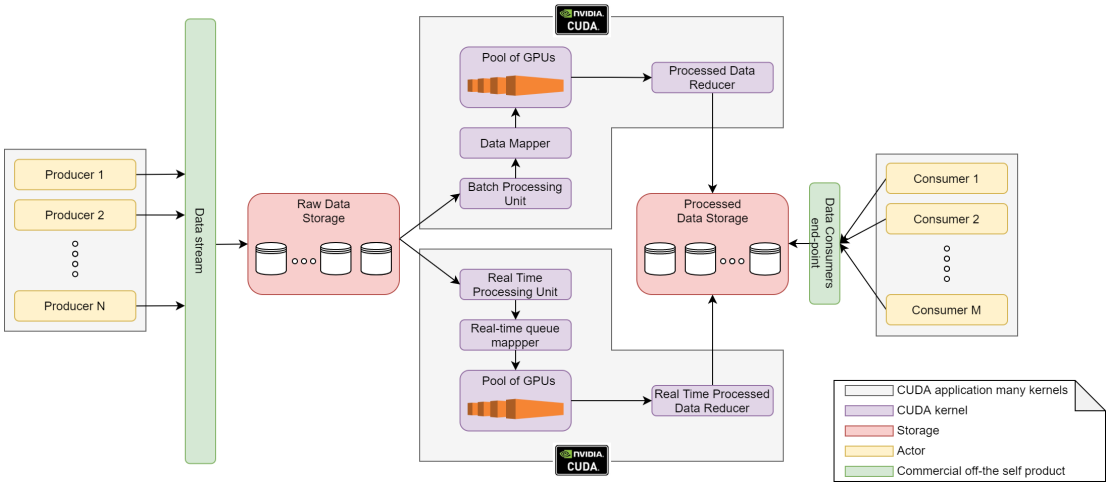

Fig. 1. Solution Architecture Overview with the core building blocks

3 Solution and Implementation

For a realization of such a system, its main building blocks along with the actors and their responsibilities must be defined following parallel programming practices and software engineering approaches. Scalability and robustness should be the starting point and main differentiator.

Fig. 1 presents a general overview of the system architecture and main building blocks. The parts of the system can be divided into commercial off-the-shelf products (COTS), distributed storage mechanisms that are necessary given the data volume, and two CUDA applications being the primary focus of this research.

The main separation in the system is two CUDA applications, i.e. Batch Processing and Real-Time Processing, together serving as lambda architecture. The need for a separate batch processing application is inevitable because of the already present data for approximately 160M cases, that has to be processed before any real-time processing can be executed.

Our approach is incrementally detailing the abstractions to render concretions, so the commercial off-the-shelf products will be discussed first.

- **Data Stream**: It must enable a real-time data stream of the data provided by the producers and their different mediums for event publishing. Apache Kafka can be leveraged for serving this purpose.
- **Raw Data Storage**: The main purpose is to store the raw patients' data obtained from the data stream, and generated by the producers. Distributed storage is forced due to the volume of data, and suitable products are any distributed document-based databases, such as Amazon DocumentDB, or Google Cloud Datastore.
- **Processed Data Storage**: Analogy with the raw data storage can be made, but processed data is stored, using the same products.
- **Data Consumers end-point**: It has to provide the data to the consumers via a common interface, for the general public a web application can be leveraged both SaaS or custom-made. For the medical professionals along with the core processed data, extensive analytics tools must be provided.

The two applications are going to be designed and implemented using core parallel programming principles and optimization techniques and during the research. Alongside block and grid sizes as standard parameters, a custom one - batch size is implied defining the number of cases that are going to be processed in a single batch.

The Batch processing application consists of host code which is responsible for launching the kernels, and it starts with the data mapper kernel, which is responsible for mapping the raw data into batches of cases the best practices and principles must be considered because this kernel can become a bottleneck. Asynchronous and Overlapping Transfers with Computation along with Zero copy are leveraged to provide state-of-the-art performance. As the batches are starting to pile up in the Unified Memory, a kernel for performing the reduction - in our case building a histogram-like data structure with symptoms as keys, and the number of occurrences as values is launched. Reduction principles and optimization methods are embodied in this kernel and collision-free updates and simulated atomic updates are leveraged.

A listener is responsible for handling the data stream events, and when new data are available it launches the real-time processing application, which consists of host code which is responsible to calculate the optimal batch size for the current data, and launch a kernel responsible for mapping the queued data to batches, and immediately reduction kernels which are in some way similar to the batch processing ones are launched, but the difference is using low-cost coherence and speculative acquisition of atomic data on the GPU, which are crucial for time-sensitive processing. As the reduction kernel produces results procedures are launched to persist them into Processed Data Storage. Techniques should be considered to tackle variable grid, block, and batch sizes due to the amount of published data. This application requires a greater amount of computing power, with a pool of GPUs supporting unified memory access.

All of the above-mentioned kernels are executed on a

pool of CUDA-enabled GPUs being part of a cloud service or on-site infrastructure.

The proof of concept was realized by the programming language Python, overcoming the speed challenges proposed using C-wrappers to get native performance, and using Numba library for built-in CUDA support.

All of the required kernels were implemented using principles for efficient parallel programming. The amount of data is opposing usage of asynchronous transfers and leveraging unified memory. Data storage was implemented in NoSQL approach using Firestore - service from Google Firebase, and Firebase Cloud Functions were used to invoke the Real-Time processing unit which encapsulates the kernels for real-time data processing. Logging was performed to evaluate this proof of concept implementation.

We calculate speedup of CPU parallel with Java Multi-threading implementations versus the serial implementation.

4 Evaluation Methodology

During research real COVID-19 patients' data were unavailable due to privacy regulations. Since our system is agnostic of the data origin, and performs computation on vectors, evaluation and conclusions could be conducted and drawn from generated data.

Data will be processed with a proof-of-concept implementation of the proposed architecture, processing, storage, and overall end-to-end times will be measured for various block and grid sizes on two devices. Serial and multi-threading implementations have been realized to obtain a relevant comparison basis.

Clustering will be performed on processed data to obtain metrics relevant for complex analysis performed in the Medical Professionals use case.

Producers were simulated with data generation, such that for every data-set size, 5 to 10 symptoms were generated for every mock patient, and the proof of concept implementation was executed against these data.

5 Results

The batch processing was executed on NVIDIA GTX 1650 (Fig.3), and NVIDIA Tesla K80 (Fig.2) provided from Google Cloud Platform. The atomic operations used while computing the counts implied using a lower number of threads per block, i.e. 128 on GTX 1650, and 256 on Tesla K80.

Table. 1 summarizes the results, and the best are 20.03s on Tesla K80 and 331.53s on GTX 1650, thus showing that speedup and performance of the GPU are directly proportional.

On the other hand, the multi-threading approach which was executed on i7-9750H with 12 concurrent threads took 4081.24s, (more than 1 hour) of processing. The serial approach was executed on the same hardware, but it ran out of memory and crashed after 7709.25s. Processing time compared against different approaches is presented in Fig. 4

From the results we can observe that our approach provides 1231.03% performance increase compared to traditional multi-threaded approach.

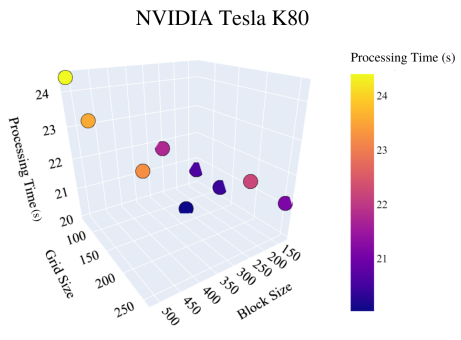

Fig. 2. Processing times on NVIDIA Tesla K80

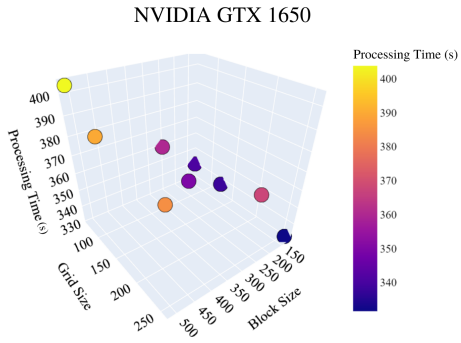

Fig. 3. Processing times on NVIDIA GTX 1650

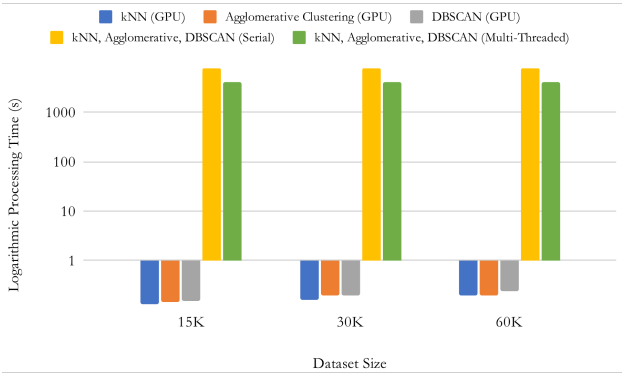

Fig. 4. Processing Time between approaches

6 Discussion

During the research, we have executed clustering algorithms of processed data. The results shown in Table 2, justify the feasibility of our architecture. Parallel implementation of K-Nearest Neighbours clustering was executed on various subsets of data with varying sizes and the average time needed to learn the clusters was 1.46 s. Parallel hierarchical clustering was also executed and the average time to learn the clusters was 2.17 s. Execution of DBSCAN clustering provided an average time of 2.86 s.

TABLE 1
Summary of data clustering results

	Device	NVIDIA GTX 1650		NVIDIA Tesla K80	
Clustering Algorithm	Dataset Size	Cluster Learning Time *(s)*	Average Time *(s)*	Cluster Learning Time *(s)*	Average Time textit(s)
kNN	15000	1.85	2.36	0.13	0.16
	30000	2.35		0.16	
	60000	2.89		0.20	
Aglomerative Clustering	15000	2.15	2.69	0.14	0.18
	30000	2.89		0.19	
	60000	3.02		0.20	
DBSCAN	15000	2.32	2.96	0.15	0.19
	30000	2.96		0.19	
	60000	3.59		0.24	

6.1 General Public use case

General Public has access to the system via a web application, which can be integrated into Worldometer [12]. Symptoms along with the counts of associated cases are listed in a simple interface, and discrete data count visualizations are presented. The primary goal of this use case is to provide real-time information in an illustrative and self-explanatory manner, such that conclusions about the spread of the disease and its manifestations can be drawn. Our architecture proposal enables such real-time data processing and filtering - which are still not available.

6.2 Medical Professionals use case

Medical Professionals could leverage an application allowing them to conduct powerful exploratory data analysis and pattern recognition which are crucial for efficiently tracking new mutations and rate of spread. Our system provides mechanisms for the execution of complex queries on processed data. Medical professionals could track symptoms and be presented with clusters among data in geographical regions for specific mutations of virus in a given time period. Analysis of correlation between different mutations and post-vaccine symptoms and side effects could be performed.

6.3 All-Purpouse use case

This use case extends the capabilities and benefits of our system even further by exposing all processed data publicly. Based on the open host principle a published language will be provided, such that processed data could be consumed via REST or GraphQL and various applications on top of these data could be realized and further research could be conducted.

7 Conclusion and Future work

The main focus of our research was to discover and explore whether data-driven and GP-GPU approaches could be used to provide real-time symptoms, long-term consequences, vaccine and disease side effects ranking, and tracking. The feasibility of our research is proven and implementation of the proposed system architecture is realized. Scalability, processing throughput, reusability, and representation of processed data are core features of our system.

In the future, significant resources and effort should be considered in the aspects of data collection, storage, and overall data-driven approach while tackling the next pandemic. Patient and disease data along with processing architecture could provide knowledge and insights which were never considered before, leading to better handling and prevention of infectious diseases.

References

[1] NVIDIA, P. Vingelmann, and F. H. Fitzek, "Cuda, release: 11.2," 2021. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[2] T. Chakraborty and I. Ghosh, "Real-time forecasts and risk assessment of novel coronavirus (covid-19) cases: A data-driven analysis," *Chaos, Solitons & Fractals*, vol. 135, p. 109850, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960077920302502

[3] M. Lucero, N. Miranda, and F. Piccoli, "Viral diseases propagation analysis in short time," in *Cloud Computing, Big Data & Emerging Topics*, E. Rucci, M. Naiouf, F. Chichizola, and L. De Giusti, Eds. Cham: Springer International Publishing, 2020, pp. 41–57.

[4] S. A. Shalom, M. Dash, and M. Tue, "An approach for fast hierarchical agglomerative clustering using graphics processors with cuda," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 35–42.

[5] A. Bagga and D. Toshniwal, "Parallelization of hierarchical text clustering on multi-core cuda architecture," *high performance computing on graphics processing units*, 2012.

[6] R. Campana-Olivo and V. Manian, "Parallel implementation of nonlinear dimensionality reduction methods applied in object segmentation using cuda in gpu," in *Algorithms and technologies for multispectral, hyperspectral, and ultraspectral imagery XVII*, vol. 8048. International Society for Optics and Photonics, 2011, p. 80480R.

[7] X. Cui, J. S. Charles, and T. Potok, "Gpu enhanced parallel computing for large scale data clustering," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1736–1741, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X12001707

[8] D. M. Chan, R. Rao, F. Huang, and J. F. Canny, "T-sne-cuda: Gpu-accelerated t-sne and its applications to modern data," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2018, pp. 330–338.

[9] D. M. Chan, R. Rao, F. Huang, and J. F. Canny, "Gpu accelerated t-distributed stochastic neighbor embedding," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 1–13, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S074373151830875X

[10] G. R. Lacerda Silva, R. R. De Medeiros, B. R. A. Jaimes, C. C. Takahashi, D. A. G. Vieira, and A. De PáDua Braga, "Cuda-based parallelization of power iteration clustering for large datasets," *IEEE Access*, vol. 5, pp. 27 263–27 271, 2017.

[11] Y. Zhang, F. Mueller, X. Cui, and T. Potok, "Gpu-accelerated text mining," in *Workshop on exploiting parallelism using GPUs and other hardware-assisted methods*, 2009, pp. 1–6.

[12] DadaxLimited, "Worldometer," 2021. [Online]. Available: https://www.worldometers.info/