Real-Time GPU-Accelerated Social Media Sentiment Processing and Visualization

Eugene Ch'ng[1], Ziyang Chen[1] and Simon See[2]
eugene.chng@nottingham.edu.cn; young.chen095@gmail.com; ssee@nvidia.com
[1]NVIDIA Joint-Lab on Mixed Reality, University of Nottingham Ningbo China.
[2]NVIDIA Technology APJ Centre, Singapore.

***Abstract*—Data visualization is an important aspect of data analytics in an age where decisions are all based on information. Approaches in data visualization, particularly those that have the capability of processing large-scale textual datasets and visualize them as structured information in real-time can be useful for monitoring trends in social media. In this article, we present our GPU accelerated project, which uses CUDA to distribute and parallelize the processing and analysis of textual data in order to visualize information in real-time, or close to real-time as a foundational system for the future of real-time applications which monitors trends in social media, applicable to political elections, social media analytics, and other needs in computational social sciences which are time-critical.**

Keywords—GPU acceleration, sentiment analysis, real-time visualization, big data, social media, twitter

I. INTRODUCTION

The ability of computing systems to efficiently process Big Data into usable information could have profound influences on the development of certain fields [1]. This need for parallel processing of large volume of data is amplified by and the need to generate meaningful information in real-time, which the nature of this research aims to accomplish. As of 2017, Twitter generates around 6,000 tweets per second. This corresponds to over 350,000 tweets per minute, 500 million tweets per day and around 200 billion tweets per year [3]. Twitter's ability to reach mass audiences, and the freedom to speak and reach out to large audiences can be very useful, and highly influential in politics and marketing. As a consequent, it is also an excellent source of data for academic enquiries. However, unlike numerical applications suited to CUDA's native computability, textual processing has a different set of issues. Past researches exploring GPU acceleration for text or string related processing via CUDA has been sparse, they were works conducted a decade ago, and the variants in the topic of research whilst centered on text strings are related to either genomes, or Internet documents (e.g., Parallel Cmatch algorithm, Knuth-Morris-Pratt on-line string matching, TFIDF rank search, DNA fruit fly string matching, the Naive, Knuth-Morris-Pratt, Boyer-Moore-Horspool and Quick-Search on-line exact string matching algorithms) [4]–[8]. We felt that the need to explore social media data with GPU acceleration has become important, with many application areas, especially when GPU acceleration can be integrated with real-time visualization.

In this paper, we present a foundational system for testing real-time visualization of information, using native code and APIs, and drawing from a set range of textual data sourced from Twitter's streaming API related to the 2016 US Presidential Election. Our system achieved a speed ~40,000 lines of data each second, the speed of which we believe to be capable of managing the volume and velocity in most Twitter trending hashtags. As far as we know, our work may be the first to implement sentiment processing and real-time visualization of social media trends using GPU acceleration.

II. METHODOLOGY

This section describes our design, and implementation of an integrated GPU accelerated text sentiment processing system using CUDA and OpenGL.

Our dataset has its source from Twitter's streaming API, using our big data architecture described in another paper [2]. Our data is used for simulating a sufficiently large volume and velocity where a single CPU may find it challenging to process in real-time.

978-1-5386-4028-9/17/$31.00 ©2017 IEEE

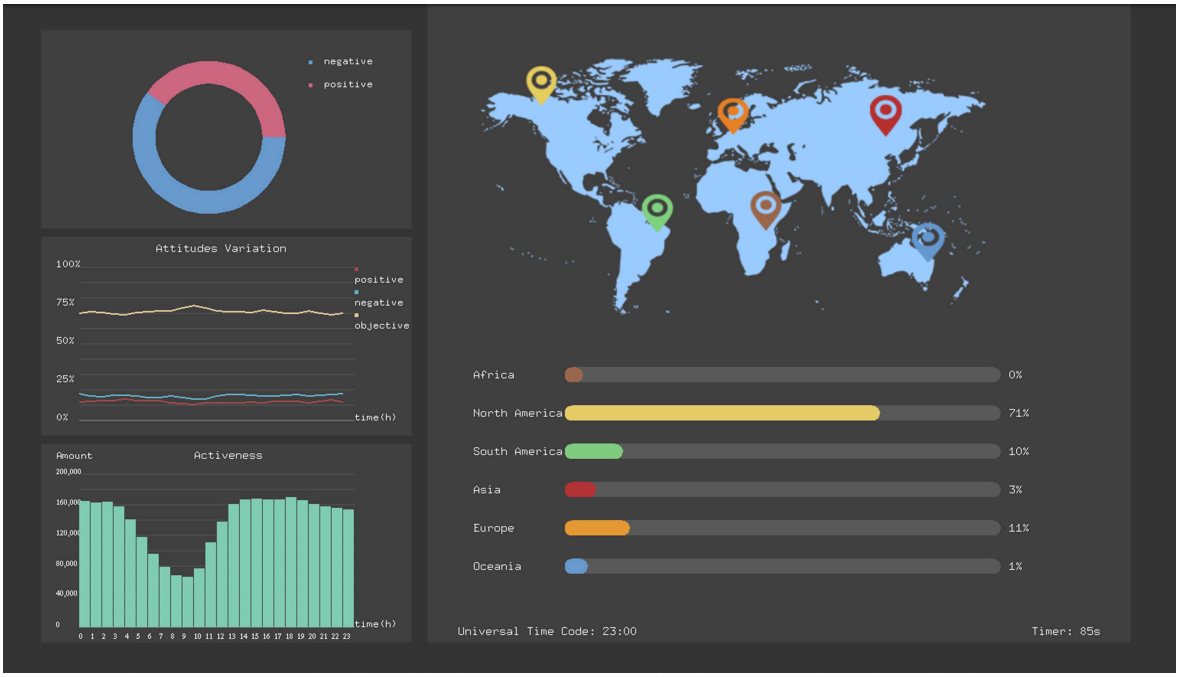

Fig. 1. An overview of the flow of data in our GPU accelerated sentiment processing and visualization system

We used the Twitter hashtag #election to acquire an extremely large set of textual data (from 27 May 2016 to 29 Nov. 2016) yielding in total 69.96GB of Twitter data containing these attributes: *{time, tweet id, tweet, source, lang, name, screen name, location, description, followers count, friends count, listed count, created at, favorites count, time zone, statuses count, longitude, latitude, place, user id}*. We sampled a single day from the dataset, with a continual range of tweets for this study from 22 Nov. 00:21 to 23 Nov. 00:21, with a data volume of 3,353,366 tweets, sized at 1.5GB and the attributes *{time, tweet, time zone}* were selected for our work. The 3 million lines of tweets will be sufficient to simulate a single day of 'real-time' visualization as a foundation to future work.

Our equipment is our custom built Intel® Xeon(R) CPU E5-2620v3 2.40GHz x 12, with 64GB of RAM. Our GPUs were NVIDA GEFORCE GTX750 Ti (PCIe x8) used for visualization, and a Tesla K80 24GB for CUDA processing. The NVIDIA device driver is v367.48. Ubuntu 14.04 LTS sits on our server with CUDA SDK version v8.0.44 and OpenGL v4.5. The C programming language (gcc 4.8.4) was used for the development.

We prepared a set of data containing: Tweets (3,353,366 lines x 21 attributes) within 6 major regions and sites (Africa 126, Asia 167, Europe: 142, North America 205, South America 45, Oceania 89), a positive word list (290 words), a negative word list (271 words), and the lengths of the char arrays above. Only single-word comparisons were conducted as our purpose was to evaluate the parallelised text processing of our system. Future work will include more sophisticated analysis with multi-word terms, co-presence of words, etc.

Each data above were loaded into a 2D char-array in the Host (CPU) and copied with cudaMalloc to the global memory of the Device (GPU). The Device does not support C/C++ String libraries or text-processing functions (e.g., length, indexing, etc.), as such the char arrays were used, and algorithms had to be written natively.

A. GPU Accelerated Textual Processing

We distributed our data with *N* lines into a number of blocks and threads within the Tesla K80 with 24GB.

We used our natively written code for each operation within the Device. The operations are not atomic but is a set of operations conducted to accomplish a task:

1. **Extracting Tweet Attributes:** Each line of tweet has 21 attributes of which 3 were selected. For each attributes we take the location of the commas (,) as a loci for extracting each attributes. We extracted 3 attributes *{time, tweet, time zone}* from each tweet for processing below.
2. **Extracting Keywords:** For each attribute *tweet*, we extracted each word based on the string residing between either a comma and a space (Hello, I am Donald Duck, I live in Disney Land) or two spaces (e.g., "I am Donald Trump"), and variants of these patterns.
3. **GPU Acceleration:** We distributed our algorithms into blocks and threads based on N lines of data.
4. **Comparing Negative and Positive Keywords:** Each extracted word in the tweet was compared first with the

length of each negative and positive word, and if they match, they are compared character-wise.

5. **Calculating Sentiment Scores:** We calculated the number of positive tweets and negative tweets based on the positive/negative keywords and assigned +1 the each appearance of each word. These are stored within two integer arrays in the Device.
6. **Location of Sentiment:** The time zone attribute of each tweet is extracted, and the corresponding tweet's extracted words are compared with the positive/negative keywords. The scores are then assigned to each country's integer array.
7. **Device to Host Memory Copy:** All calculated scores are stored in the intermediate Device arrays using the cudaMemcpy() and returned to the Host at the end.

Even though our sentiment processing is simple, the algorithm has three nested loops and the computational complexity is necessary in native code and is a natural test of our algorithms when we compare CPU and GPU text-processing speed in section 4. The pseudo code is given in Table 1. There are two similar pseudo codes for our operations listed above (in 4 and 6). Operation 4 is listed in Table 1 (line 7-12), operation 6 is done in the same line in 7-12, without having another table for practical reasons.

Table 1. Pseudo code of device kernel

```
1.  [CPU]   Load data {tweets, sentiment tokens, countries} into GPU global memory
2.  [GPU]   ProcessCompareText (for each thread)
3.          .Initialize arrays for storing return scores
4.          .while more text
5.                  .split char array and get attributes {time, tweets, location}
6.                  .for each tweet, split into words
7.                          .for each tweet words
8.                                  .for each word in the word list
9.                                          .if word matches sentiment words
10.                                                 .record subjective {pos, neg}
11.                                 .end for
12.                         .end for
13.                 .end for
14.         .end while
15. [GPU]   GPU returns scores
16. [CPU]   CPU uses scores for OpenGL visualisation
```

B. CPU Data Processing

After the data within the global memory of the GPU is processed, they are passed to the Host. The Host receives the Device memory and calculates the ratio between positive and negative sentiments, which is a fairly quick process. The Host also stores the intensity of Twitter activity in terms of the number of tweets each hour within a day whilst they are being loaded into the Host memory. Finally, the Host calculates the percentage of the population involved in the hashtag #election: For $u \in continents$ *and* $a \neq \emptyset$, $P=s/L*100\%$ where P is the percentage of population active in the hashtag, s is the population discussing within a continent, and L stands for all recognized locations in the data.

The calculated values are then passed to OpenGL for rendering.

C. Real-Time Visualization

To achieve the final step of our goal to build and test a real-time monitoring system for social media, we constructed a user interface using OpenGL and SDL. SDL manages the inputs for key-presses, and the mouse is used for starting and stopping the simulation. Within the OpenGL window, we split our display into 5 compartments (Fig. 1).

Positive/negative sentiment ratio – this aims to show the changes to positive and negative sentiment within a graph.
Variation of attitudes – this shows the variations between negative (blue), positive (red) and (neutral) sentiments over time
Histogram of activity intensity – this is a diagram showing the number of discussions occurring within each hour in a day.
A world distribution map – the world map indicates the percentage of population discussing the issues surrounding our hashtag.
Sentiment of each continent – the bar chart corresponds to the world distribution map's activities.

III. Results

Even though our system algorithm is computationally expensive, we achieved a speed that is capable of receiving real-time inputs from Twitter, the world's most active, and largest social media platform. Our GPU accelerated processing reached a speed of 43,057 lines per second when only 6,000 tweets are generated every second [3] on average.

Our parallel algorithm achieves significant results when the amount of data pushed through to the GPU is measured against the time it took to process the data. Fig. 2 illustrates the performance comparison.

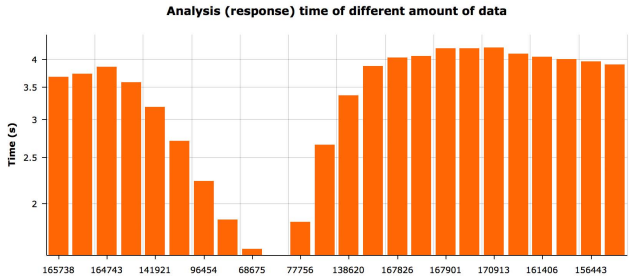

Fig. 2. GPU textual processing time (y axis) and the density of textual contents (lines of texts as volume in the x axis)

We conducted a test to compare the performance of our algorithms between CPU vs GPU, by subsequently raising the order of magnitudes. The result is shown in Fig. 3.

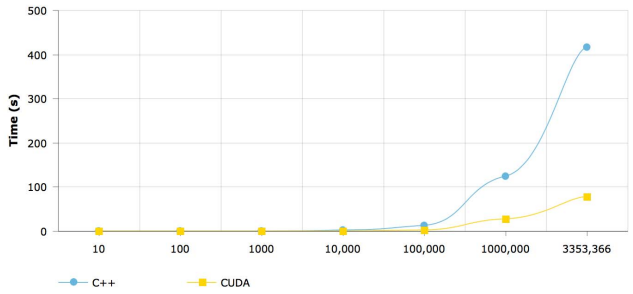

Fig. 3. A graph corresponding to the text-processing time (y axis in seconds) between GPU and CPU with the volume of tweets in orders of magnitude (x)

Fig. 4 inspects the intersection between CPU and GPU performance when data is needed to be loaded into Device memory. The graph indicated that CUDA uses more time to get data into the global memory. However, around the 4,000 mark, the superior performance of GPU parallelization can be seen. This demonstrated that the CPU could be used for data sizes that are <5,000.

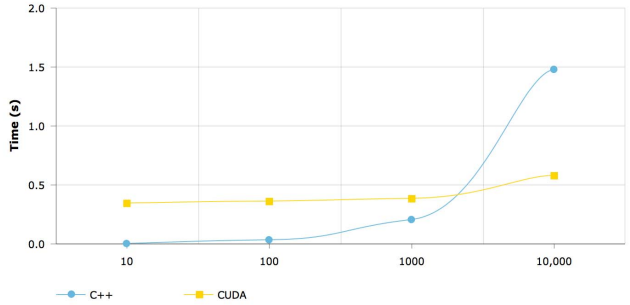

Fig. 4. A comparison of time needed (y) for textual processing between CPU and GPU showing the intersection between two performances when the data size reaches the threshold of ~5,000 (x).

The efficiency of CUDA can be dramatically higher than programs running on CPUs. GPU performance remains relatively stable if we exponentially increase the volume of data. CPU performance on the other hand can become unmanageable (Fig. 5).

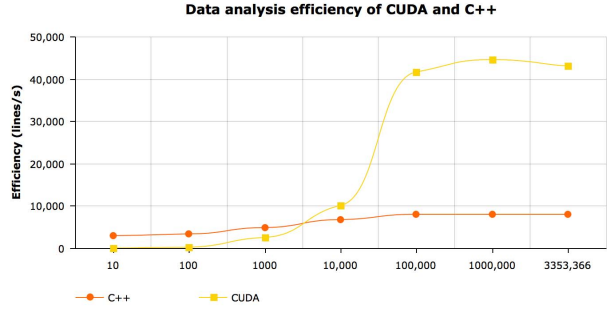

Fig. 5. Comparison of the stability of performance (y) between CPU and GPU when data is increased in the order of magnitudes (x)

IV. Conclusion

In this paper, we developed and evaluated a GPU accelerated social media data processing system, based on the premise that the future of meaningful information visualization requires big data processing, in real-time. We provided the motivation, and build the case for our pilot project in the background, and implemented a system consisting of a GPGPU data processing backend and a visualization frontend using OpenGL. Our hardware is self-contained, with two GPGPUs – GTX750 Ti for visualization and a Tesla K80 for data processing.

Our performance evaluation comparing CPU and GPU yielded good results. We demonstrated that the volume and velocity of data did not affect the performance of our system and that our GPU tasks were stable even when data size reaches *n* order of magnitudes.

In summary, when the volume of data generated from social media is exponential, reaching the size in the order of magnitude, distributing and parallelizing algorithms within the GPU will be necessary for real-time visualization.

Work is under way to extend this project with more sophisticated sentiment analysis algorithms, with optimized code and a more efficient integration between data processing and visualization.

Acknowledgement

The authors wish to thank NVIDIA for supporting the NVIDIA Joint-Lab on Mixed Reality with equipment and developmental needs, without which this project cannot be realized.

References

[1] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of 'big data' on cloud computing: Review and open research issues," *Inf. Syst.*, vol. 47, pp. 98–115, 2015.

[2] E. Ch'ng, "The Value of Using Big Data Technology in Computational Social Science," in *The 3rd ASE Big Data Science 2014, Tsinghua University 4-7 August*, 2014, pp. 1–4.

[3] ILS, "Twitter Usage Statistics," *InternetLiveStats.com*, 2017. .

[4] M. Schatz and C. Trapnell, "Fast exact string matching on the GPU," *Cent. Bioinforma. Comput. Biol.*, 2007.

[5] N. Jacob and C. Brodley, "Offloading IDS Computation to the GPU," in *Proceedings of the 22nd Annual Computer Security Applications Conference IEEE ACSAC'06.*, 2006, pp. 371–380.

[6] 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.

[7] C. S. Kouzinopoulos and K. G. Margaritis, "String matching on a multicore GPU using CUDA," in *Informatics, 2009. PCI'09. 13th Panhellenic Conference on*, 2009, pp. 14–18.

[8] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and I. S., "Gnort: High Performance Network Intrusion Detection Using Graphics Processors," in *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, 2008, pp. 116–134.