Dealing With Big Data Outside Of The Cloud: GPU Accelerated Sort

John Vidler[*], **Paul Rayson**[*], **Laurence Anthony**[†], **Andrew Scott**[*], **John Mariani**[*]
[*]School of Computing and Communications, Lancaster University
j.vidler, p.rayson, a.scott, j.mariani@lancaster.ac.uk
[†]Faculty of Science and Engineering, Waseda University
anthony@waseda.jp

Abstract

The demands placed on systems to analyse corpus data increase with input size, and the traditional approaches to processing this data are increasingly having impractical run-times. We show that the use of desktop GPUs presents a significant opportunity to accelerate a number of stages in the normal corpus analysis pipeline. This paper contains our exploratory work and findings into applying high-performance computing technology and methods to the problem of sorting large numbers of concordance lines.

Keywords: Very Large Corpora, Concurrency, GPU Computing, High Performance Computing, Concordances, Sorting

1. Introduction

Corpus data is used in many areas of Digital Humanities, Natural Language Processing, Human Language Technologies, Historical Text Mining and Corpus Linguistics. Increasingly, however, the size of corpus data is becoming unmanageable. In Digital Humanities, for example, national and international digitisation initiatives are bringing huge quantities of archive material in image and full text form direct to the historian's desktop. Processing such data quickly, on the other hand, will almost certainly exceed the limits of traditional database models and desktop software packages. Similarly, the "Web as a Corpus" paradigm has brought vast quantities of Internet-based data to corpus linguists. However, any search or sort of results from these rich datasets is likely to take from minutes to hours to days using desktop corpus tools such as WordSmith Tools[1] and AntConc[2].

To address the problems of handling massive data sets, international infrastructure projects, such as CLARIN and DARIAH, are emerging with support for these large corpora under the umbrella of 'big data'. However, these systems do not allow for local access, storage and retrieval of large language resources to support researchers while datasets are being collected and analysed. In corpus linguistics, researchers now have access to tools such as Sketch Engine[3] and the family of BYU Corpora[4], which aim to support pre-compiled billion-word corpora. Again, though, these systems are remotely hosted, and they are also not easy to configure for the new datasets of local researchers. More recently, semi-cloud based systems are emerging, such as GATE[5], Wmatrix[6], and CQPweb[7], which can provide users with local access to large data sources. However, the installation and configuration of such systems is far from simple, making them inaccessible to most social science and humanities based scholars.

Hence, there is still a need to investigate processing efficiency improvements for locally controlled and installed corpus retrieval software tools and databases. Core tasks such as corpus indexing, calculating n-grams, creating collocations, and sorting results on billion-word databases cannot feasibly be carried out on current desktop computers within a reasonable time.

In this paper, we describe an alternative solution to accelerate such tasks by capitalizing on the local processing power of the often underused discrete Graphics Processing Unit (GPU). To highlight the possibilities of our approach, we focus on the task of concordance results sorting and show how GPU hardware can dramatically shorten the time needed to complete the task. This research forms our first case study in a larger project to investigate the untapped potential in current operating system architecture designs.

2. Background

As corpus sizes increase, the problems of processing large datasets are become more pressing. Research on high performance processing techniques for the amounts of text that the language resources community often works with is scant at best, and generally revolves around large numbers of traditional processors being used to divide the work into manageable units. Unfortunately, with corpora exceeding the multi-billion-word mark, even these measures are unable to complete experiments within reasonable time, often spanning days of operation (Wattam et al., 2014). In addition, enhancements designed for other areas of computing, e.g., Cederman and Tsigas (2010) and Rashid et al. (2010) have proved to be not well suited to corpus processing. In recent years, high-performance, general-purpose graphics processing units have become increasingly available to the scientific community, and projects utilising them have been met with considerable success as described in Deng (2010), Melchor et al. (2008) and Sun et al. (2012). On the other hand, their use in corpus linguistics and natural language processing has been limited at best, and many areas of their uses have yet to be explored.

[1]`http://www.lexically.net/wordsmith/`
[2]`http://www.antlab.sci.waseda.ac.jp/software.html`
[3]`http://www.sketchengine.co.uk/`
[4]`http://corpus.byu.edu/`
[5]`http://gate.ac.uk/`
[6]`http://ucrel.lancs.ac.uk/wmatrix/`
[7]`http://cqpweb.lancs.ac.uk/`

the crowds of inquisitive people began to diminish and soon	there	were no more visitors Madame Caravan returning to her own
already to regard the corpse as though it had been	there	for months He even went the length of declaring that
He even went the length of declaring that as yet	there	were no signs of decomposition making this remark just at
the girl who had ascended the stairs were distinctly heard	There	was silence for a few seconds and then the child
in their places and was ready to go downstairs when	there	appeared before her her son and daughter-in-law Caravan rushed forward
agree to be bound by the terms of this agreement	There	are a few things that you can do with most
full terms of this agreement See paragraph C below	There	are a lot of things you can do with Project

Figure 1: A sample of the input set used for testing the sorting algorithms.

3. Method

To evaluate the potential gains of using General Purpose Graphics Processing unit (GPGPU) techniques for corpus retrieval operations, we chose to focus on one of the most common and time consuming tasks that corpus linguists need to perform, i.e., sorting the concordance lines generated from a database query. Once concordance lines are extracted or displayed in a corpus retrieval system, corpus linguists need to identify language patterns in the results set. Due to the large number of results, the concordance lines cannot be skim read manually, and so some pre-processing is required, typically sorting. Most concordancing tools, such as WordSmith Tools and AntConc, can perform a multi-level sort of the results based on the preceding and/or following words. However, this can be a very lengthy operation, especially when many hundreds of thousands of concordance lines emerging from large corpora require processing.

The words we targeted for the corpus sort experiment were taken from the published BNC frequency lists of "Word Frequencies in Written and Spoken English" (Rayson et. al)[8], which were used with a corpus generated from the Gutenburg Project books data[9] to generate CSV input sets as shown in Figure 1. These were loaded into memory in full, and stored such that the entire concordance was kept in RAM. While this may seem suboptimal, we did this to attempt to present data that represented the performance of the GPGPU device, rather than memory usage tricks. Additionally, the batch-processing style of operation used in GPGPU computing limits our ability to do most traditional sorting techniques for large datasets, such as an external merge sort, as the GPU does not support the recursion depth required to process this data. Following the above procedure, we could reduce the problem to an entirely data oriented issue and avoid the characteristics of disks, buses, networks and other hardware components interfering with the performance measures.

Based on a preliminary investigation of different sorting algorithms, we found most to be data-copy sensitive (requiring many short batch operations and many host-device memory copies). Thus, we settled on using the simplest sorting technique, the swapping sort, for the analysis here. In the swapping sort, concordance lines are directly loaded into memory on the graphics card and processed in place by comparing each line to its immediate neighbours in the input set, and swapping the entries if they are incorrectly ordered. This is repeated until the entire set is sorted. While this technique would be extremely inefficient on a CPU, it works impressively well on a GPU, as we can perform large batches (over 27,000 entries, on a nVidia GTX Titan) at once. The relevant specifications of the machine used for these tests is described in Figure 2.

- CPU: Intel "Sandy Bridge" i7, desktop edition (quad core with hyperthreading support).
- GPU: nVidia "GTX Titan" graphics card, 6GB video memory.
- RAM: 6GB Triple Channel memory.
- Disk: A generic 64GB 6GB/s SSD

Figure 2: The specification of the machine used to perform the tests.

While the exact specification for the hardware is not particularly critical, to achieve similar results, the use of a GTX Titan or better is recommended, as older cards have smaller video memory areas, resulting in higher instances of copying to and from the hard drive. The SSD is not required, but was used for these tests to expediate the test duration through eliminating the delays normally incurred through using mechanical disks.

4. Results

The results of our tests can be seen in Figures 3 and 4[10]. Each sort phase used up to 10 words to the right of the selected collocation word to sort the concordance against its neighbours. Phases were repeated until a phase resulted in no swap operations, and thus, the set was completely sorted.

As can be seen in Figures 2 and 3, the GPU accelerated sort consistently beats the sort on a normal CPU except for very small input sets. Below an input set size of 2000 concordance lines, the CPU has a slight advantage, as the GPU has a small delay involved with deploying CUDA kernels[11], causing the overall throughput to dip. On the other hand, beyond 2000 concordance lines, the GPU is several orders of magnitude faster than the CPU, and remains consistently better throughout.

5. Discussion and Conclusions

The results here show that normal CPU processing becomes impractical when sorting beyond 40,000 concordance lines.

[8] `http://ucrel.lancs.ac.uk/bncfreq/flists.html`

[9] `http://www.gutenberg.org/wiki/Gutenberg:The_CD_and_DVD_Project`

[10] The data for these plots as well as additional data can be found at `http://johnvidler.co.uk/academia/cmlc-2014/`

[11] A CUDA kernel is the GPGPU equivalent of a CPU thread

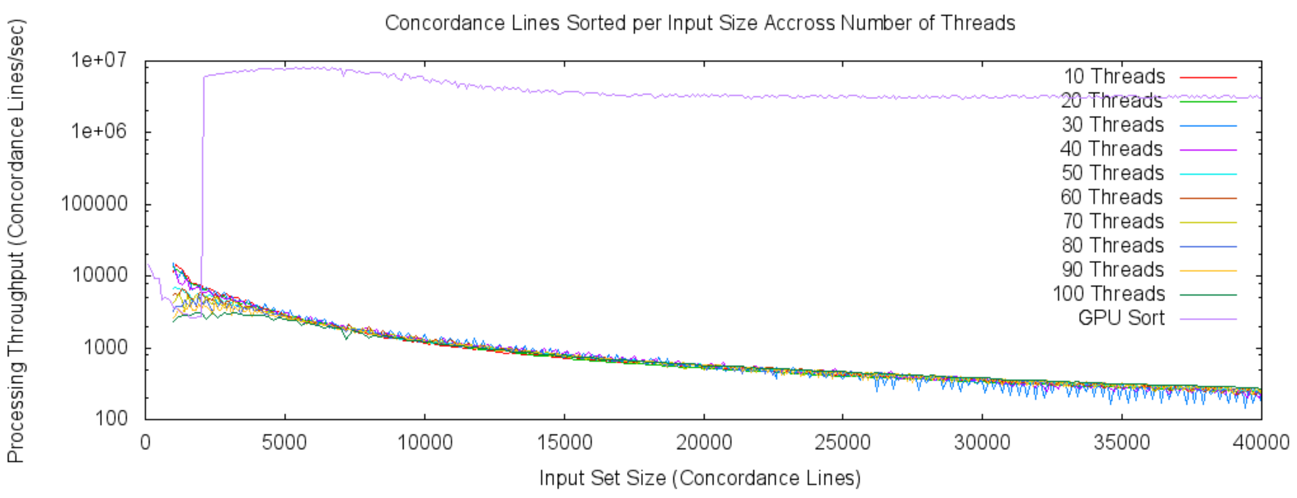

Figure 3: The measured CPU and GPU performance measurements shown on the same axis. Beyond 40,000 concordance lines, the sorting technique took so long to complete on the CPU as to be useless, while the GPU continued to perform exceptionally well. Note that the y-scale is logarithmic.

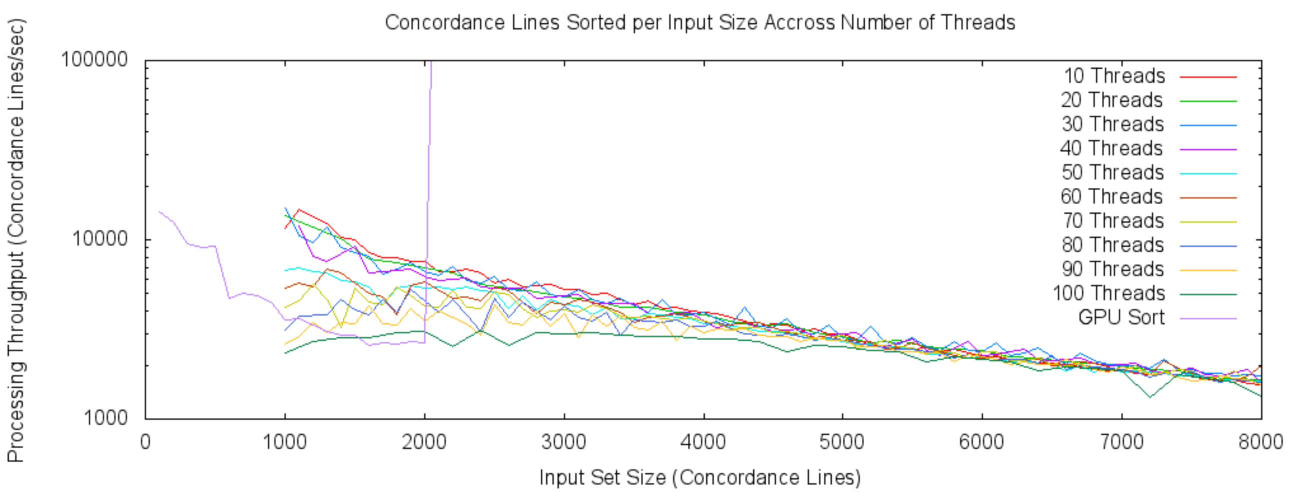

Figure 4: The first portion of Figure 3, showing the initial CPU advantage for very small numbers of concordance lines.

In contrast, our framework for GPU processing allows such operations to be completed exceedingly quickly, e.g., with 10 million concordance lines being processed per second even with large datasets. Further gains can be made through the use of threading on GPGPU devices, although the library support available to the developer is not what one would expect. This leads to problems implementing more traditional sorting algorithms, such as external merge sorting, with such a large input set. Many traditional sorting algorithms require access to the hard drive and other resources that are unavailable to the card during runtime. The processing used in our tests is best described as 'out-of-data-path' processing, as the data would not naturally be processed where we are processing it. Normally one would expect that moving the data further from where it is stored would result in slower overall performance. However, with the immense processing power available in a GPU, once there, the gains more than make up for the longer data path. Of course, while out-of-data-path processing may not be ideal from a memory utilization perspective, our results show that the performance benefits are worth the additional overhead in the application described here and indeed, our approach is likely to be useful in many other areas of natural language processing. In conclusion, our results show that the implementation of even simple algorithms on GPU hardware has significant promise for linguistic analysis of large corpora. Our approach thus has important implications for the development of more powerful desktop linguistic analysis tools. Given the performance shown in the case study presented in this paper, we next intend to implement other standard corpus retrieval operations using our framework. Following this, we will start implementing support tools for various other corpus annotation and NLP operations, e.g. Part-Of-Speech (POS) tagging.

References

Daniel Cederman and Philippas Tsigas. Gpu-quicksort: A practical quicksort algorithm for graphics processors. *J. Exp. Algorithmics*, 14:4:1.4–4:1.24, January 2010. ISSN 1084-6654. doi: 10.1145/1498698.1564500. URL `http://doi.acm.org/10.1145/1498698.1564500`.

Yangdong Steve Deng. IP routing processing with graphic processors. *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pages 93–98, March 2010. doi: 10.1109/DATE.2010.5457229. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5457229`.

Carlos Aguilar Melchor, Benoit Crespin, Philippe Gaborit, Vincent Jolivet, and Pierre Rousseau. High-Speed Private Information Retrieval Computation on GPU. In *Proceedings of the 2008 Second International Conference on Emerging Security Information, Systems and Technologies*, pages 263–272, Washington, DC, USA, August 2008. IEEE Computer Society. ISBN 978-0-7695-3329-2. doi: 10.1109/SECURWARE.2008.55. URL `http://portal.acm.org/citation.cfm?id=1447563.1447928`.

Layali Rashid, WessamM. Hassanein, and MoustafaA. Hammad. Analyzing and enhancing the parallel sort operation on multithreaded architectures. *The Journal of Supercomputing*, 53(2):293–312, 2010. ISSN 0920-8542. doi: 10.1007/s11227-009-0294-5. URL `http://dx.doi.org/10.1007/s11227-009-0294-5`.

Weibin Sun, Robert Ricci, and Matthew L. Curry. GPUstore. In *Proceedings of the 5th Annual International Systems and Storage Conference on - SYSTOR '12*, pages 1–12, New York, New York, USA, 2012. ACM Press. ISBN 9781450314480. doi: 10.1145/2367589.2367595. URL `http://dl.acm.org/citation.cfm?id=2367595`.

Stephen Wattam, Paul Rayson, Marc Alexander, and Jean Anderson. Experiences with Parallelisation of an Existing NLP Pipeline : Tagging Hansard. In *Proceedings of The 9th edition of the Language Resources and Evaluation Conference*, 2014.