USING MULTIPLE GPUS TO ACCELERATE STRING SEARCHING FOR DIGITAL FORENSIC ANALYSIS

Ethan Bayne, Robert Ian Ferguson, Adam Sampson, John Isaacs
Division of Computing and Mathematics, Abertay University, Dundee
School of Computing Science and Digital Media, Robert Gordon University, Aberdeen
e.bayne@abertay.ac.uk; i.ferguson@abertay.ac.uk; a.sampson@abertay.ac.uk; j.p.isaacs@rgu.ac.uk

ABSTRACT

String searching within a large corpus of data is an important component of digital forensic (DF) analysis techniques such as file carving. The continuing increase in capacity of consumer storage devices requires corresponding improvements to the performance of string searching techniques. As string searching is a trivially-parallelisable problem, GPGPU approaches are a natural fit – but previous studies have found that local storage presents an insurmountable performance bottleneck. We show that this need not be the case with modern hardware, and demonstrate substantial performance improvements from the use of single and multiple GPUs when searching for strings within a typical forensic disk image.

Keywords: Digital forensics, String searching, GPGPU, GPU, IGP, Parallel programming.

1. INTRODUCTION

In the last decade, we have wit-nessed consumer storage devices vastly increase in capacity, with modern hard drives offering up to 10 terabytes of data on a single drive. While this is somewhat an expected and inevitable result of technological evolution, the repercussions on the ability to perform digital forensic (DF) analysis on modern sizes of storage devices have slowly deteriorated. This is primarily due to the processing techniques that current tools employ to analyse each segment of data on the storage device; with many tools still employing processing techniques which performed adequately in the last decade, but struggle to perform admirably in the challenges we face today. As a result, the inability to analyse large amounts of digital evidence efficiently leads to increasing times required to investigate DF cases, which inevitably causes backlogs of cases. This issue raises the urgent need for more powerful and accessible methods of processing digital evidence.

String searching is a vital component employed by the majority of methods used today to ascertain evidence in DF investigations. String searching can be defined as the searching through data for one, or more, pat-terns— a particular sequence of data, which could refer to a word or file identifier. In itself, performing string searching is computationally intensive as each bit of data is analysed systematically to find specified patterns of interest; while current tools employ more efficient algorithms to lessen the computational workload, a significant amount

of time is still required to search forensic data for evidence.

To mitigate the computational strain associated with string searching, suggestions have been made to employ distributed computing techniques (Ayers, 2009); however, distributed computing setups can be somewhat expensive, complicated, and require specialist knowledge to configure and use efficiently. This study presents a more accessible model, extending on previous research by Bayne, Ferguson and Isaacs (2014), which employs parallel processing techniques and multiple GPUs on a single computer that may prove more efficient and cost-effective for DF professionals. Results ascertained from this study demonstrate that utilising multiple GPUs will yield significant improvements over deploying a single GPU, or traditional central processing units (CPUs), when searching for multiple objects within data.

Dedicated GPUs were historically created to reduce stress on the CPU when rendering complex graphical images. The Nvidia GeForce 980, a typical modern GPU, has 2048 independent processing cores which are able to compute large amounts of mathematical instructions with ease due to the hardware's efficient parallel processing design. Besides its primary function of rendering graphics, a GPU's architecture and computational power make them an ideal platform to be employed to aid scientific investigations. Aside from dedicated GPUs, recent CPU architectures have commonly integrated its own dedicated graphics core— often referred to as an integrated graphics processor (IGP). IGPs found in modern CPUs share a lot of similar architecture that can be found on GPUs, however, lack the same processing power due to size limitations on CPU chips. Nevertheless, IGPs open up the possibility of employing a powerful parallel processing platform without the need for an additional discrete GPU. In order to have the broadest compatibility with GPUs and IGPs from all vendors, a widely adopted general purpose computing on graphics processing unit (GPGPU) framework, such as Open Computing Language (OpenCL), must be adopted.

Previous research by Bayne, Ferguson and Isaacs (2014) has demonstrated that OpenCL offers comparable performance and ease of implementation to vendor-specific languages such as Complete Unified Device Architecture (CUDA), which is ideal to be used with the intensive processing workload associated with DF analysis. The results from the research also demonstrated negligible performance loss when compared to the more widely researched CUDA GPGPU framework; however, with technology trends indicating continued developments of more powerful IGPs on CPUs, the authors strongly believe that a more compatible and open-standard model of the OpenCL platform favours the scientific requirements and development needs of DF tools and analyses. Significantly, however, the research countered critique that achievable performance gains are not solely limited by storage device data transfer speeds, but actually a combination of data transfer speed and processing power.

The case study presented in this paper extends upon the previous research by investigating whether employing multiple GPU or IGP devices could further improve the possible performance throughput by employing parallelisation techniques to string searching. The aim of the experiment undertaken in this paper attempts to measure the effectiveness of offloading string searching to multiple GPUs using OpenCL, while comparing the resulting performance with: a single GPU, a multi-threaded CPU, and single-threaded CPU processing techniques. Secondly, this paper discusses trends produced by running the

experiments on three different machines of varying specifications.

2. BACKGROUND AND RELATED WORK

Whilst there are no known related studies that seek to apply OpenCL parallel processing techniques to accelerate DF analysis, results from similar research that propose using CUDA have demonstrated equally promising results. Research specifically focused on assessing the potential differences between CUDA and OpenCL (Fang, Varbanescu, & Sips, 2011; Karimi, Dickson, & Hamze, 2010) establishes that the CUDA framework shows a slight perfor-mance advantage over its open-standard counterpart. This indicates that with OpenCL, a minor sacrifice in performance currently exists to obtain a wider compatibility across devices from various vendors. Conclusions from these studies also show that much of the current research on CUDA may be relevant and transferable in facilitating further OpenCL research, due to the comparable parallel approaches utilised by each technology to interface with the GPU.

A theoretical insight is presented by Skrbina and Stojanovski (2012), which discusses the preparation and processes involved in creating a GPGPU solution to accelerate file carving— a forensic technique used to reconstruct files by attempting to recognise file structures from an otherwise unstructured stream of data. The authors explore how CUDA could be utilised within the context of DF investigations, examining the different characteristics between string and pattern matching algorithms that are suitable for GPGPU parallelisation. The authors conclude that the most appropriate algorithms for parallel applications are the Boyer-Moore (Boyer & Moore, 1977) and Aho-Corasick (Aho & Corasick, 1975) algorithms for handling single- and multi-string searches, respectively. The findings presented were, nevertheless, based on theoretical grounding from algorithm characteristics.

An empirical study conducted by Marziale, Richard and Roussev (2007) investigated how the use of CUDA GPGPU processing could accelerate Scalpel, an existing open-source file carving tool. The study compares the time taken to complete various searches through different sized forensic images using the unmodified and modified GPU versions of Scalpel. Results from the study show significant improvement with GPGPU acceleration, which clearly demonstrated that incorporating GPU technology is a practical option for significantly increasing processing performance in existing DF tools. At the time of the research, however, the CUDA framework was still in beta. The authors acknowledge that the beta release possessed a number of bugs and that the compiler did not fully optimise the code; these factors may have limited the proposed solution's potential achievable performance at the time the research was conducted.

Contrasting research from Zha and Sahni (2011) states that, when incorporating a fast multi-pattern matching algorithm, the performance gain achievable from file carving is limited by the time required to read data from the disk, as opposed to the time required to conduct string searching on the data. The authors similarly conducted experiments through modifying Scalpel, where they incorporated a series of Boyer-Moore and Aho-Corasick algorithms to aid string matching on the CPU. The authors' experiments indicate that multi-threaded acceleration using a dual-core CPU did not improve the required processing time, concluding with an arguable assumption there are no advantages of using other accelerators such as GPUs, despite conducting no actual experiments involving GPGPU processing. Later research from the

same authors (Zha & Sahni, 2011b, 2013) shows that incorporating similar algorithmic techniques using GPGPU processing produced notable improvements over single-threaded CPU approaches, besting multithreaded CPU processing in some scenarios. Results of these later studies from the authors, and from previous research by Bayne, Ferguson and Isaacs (2014), form the argument that the Zha and Sahni's earlier research on processing techniques to improve file carving had not been thoroughly explored.

Collange et al. (2009) demonstrated a novel method of utilising GPU processing to aid file identification in DF investigation. The authors use a CUDA GPGPU implementation to calculate and compare hashes of data to identify potential image file identifiers located on storage devices. The authors concluded in their study that, with the computational power of the hardware, GPUs make an ideal platform on which to perform parallel hash calculations, potentially delivering a powerful and usable file identification technique for DF investigation. Although Collange et al. eliminate the requirement of knowing complete file hashes by searching for file identifiers, the proposed approach still requires and is heavily dependent on the CPU to verify the matches found as valid image files; this potentially slows the overall performance when faced with forensic images containing significant amounts of data.

3. IMPLEMENTATION

Extending on previous research, the main question that this study aims to answer is whether there are any significant performance improvements over a single GPU, multi-threaded CPU, and single-threaded CPU implementation, when employing multiple GPUs to search forensic data for patterns. In order to accomplish this, an application was developed to demonstrate string searching on both CPU and GPU devices.

3.1 Our Approach

To attempt to answer this study question, a C# application was developed which could analyse raw forensic data files on both the CPU and GPU. OpenCL was used to process and manage GPU operations. The processing approach adopted by the program is detailed in figure 1. Both the CPU and GPU implementations followed the same processing approach when performing analysis on the forensic data. To perform analysis, only one single read through the data was required. In multiple GPU and CPU tests, each available processor acted autonomously as illustrated in figure 2; in which the forensic data was similarly divided into a stack of 100 mebibyte (MiB) segments, then individually processed in turn by one of the available processors.

The CPU and GPU implementations differ only when performing string searching against the data, both in which follow their own unique algorithms to complete. The CPU employs a modified Boyer-Moore algorithm to accelerate searching through data— using a similar implementation of the algorithm which is still currently used in Foremost, a well-established open-source Linux-based file carver. This algorithm operates by searching a stream of data for the last byte of the target. When the algorithm discovers the last byte, the rest of the target is validated. Searching through the data stream is accelerated by the creation of a skip table. The skip table acts as a reference on the range to find the next possible match, depending on the value read; this significantly reduces the time required to search through the data stream for target matches. The GPU algorithm, in contrast, was developed to be a largely a synchronous brute-force algorithm; an algorithm which searches through each bit of the data segment

individually, relying on the raw power and parallel nature of GPU hardware alone to accelerate searching. While the GPU algorithm implemented is by no means efficient, it was designed to be simplistic in nature; the authors acknowledge that future research is needed on any additional benefits of implementing a more sophisticated GPU algorithm.

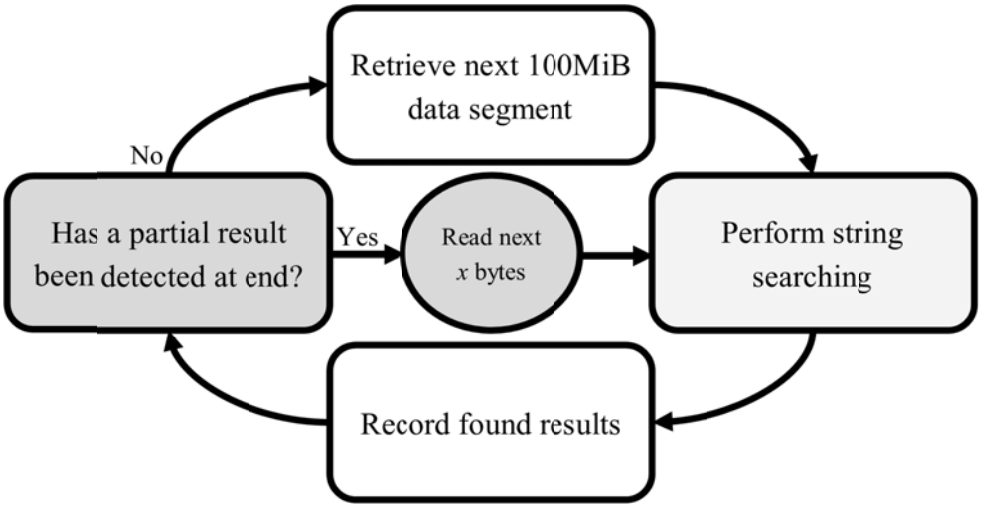

Figure 1. Processing model

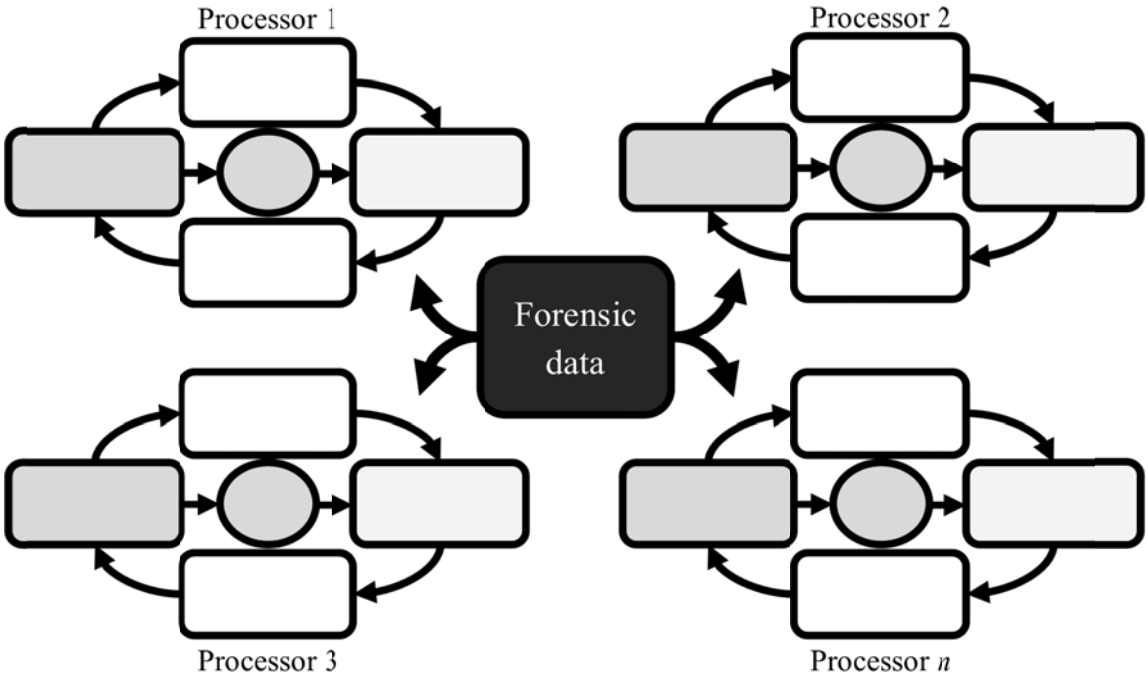

Figure 2. Multi-processor processing model

Table 1
Test platform specifications

Test Platform	**A**	**B**	**C**
Type	Desktop	Desktop	Laptop
Operating System	Windows 8.1	Windows 8.1	Windows 8.1
Processor	Intel Core i7-3770K	Intel Core i5-4690K	Intel Core i7-4700HQ
Processor Specifications	4 core @ 3.8GHz, 8 threads	4 cores @ 3.9GHz, 4 threads	4 cores @ 3.5GHz, 8 threads
Processor IGP	Intel HD4000 (Disabled)	Intel HD4600 (20 core @ 350MHz)	Intel HD4600 (20 core @ 400MHz)
Memory	8GB DDR3 1600MHz	16GB DDR3 1600MHz	16GB DDR3 1866MHz
GPU	4x Nvidia 580 GTX (1.5GB GDDR5)	Nvidia 980 GTX (3GB GDDR5)	Nvidia 970M GTX (6GB GDDR5)
GPU Specifications	2048 core @ 612MHz (combined)	2048 core @ 1203MHz	1280 core @ 924MHz
Storage Device	120GB SATA3 SSD	120GB SATA3 SSD	3x 256GB mSATA SSD (RAID0 Array)
Sequential Read Performance	362 MiB/s	245 MiB/s	1305 MiB/s

3.2 Experimental Evaluation

In an attempt to answer what bene-fits there are in employing multiple GPUs to assist in alleviating processor intensive tasks from the CPU; a test was created to benchmark processing on the following processing configurations; a single-threaded CPU, a multi-threaded CPU, a single GPU, multiple GPUs, and lastly – if available – the CPU's integrated IGP. The tests were completed on three different computers with varying hardware, the hardware specifications of each computer is outlined in table 1.

It was expected that the maximum perceivable performance to perform string searching would be limited by the component with the slowest data transfer rate or by the speed of the processor itself. To enable the processing of forensic data, the system must transfer data between the storage device, the main system memory, and – if employing GPU processing – the memory on-board the GPU itself. As the theoretical transfer speed of the system memory, GPU memory and transfers between the two on all of the three systems tested are well beyond 10 GiB/s, the storage device used to read the forensic data was apparent to be the slowest and most likely bottleneck. Measurements of each computers' storage device's sequential read speeds were recorded by storage device performance benchmarking software, CrystalDiskMark (CrystalMark, n.d.), prior to running each of the tests; to aid in the interpretation of the results and identify possible performance bottlenecks.

The test consisted of performing string searching against a 20 gibibyte (GiB) forensic image stored on the computer's local disk. The forensic image is of a storage device containing a large amount of files of varying common file

formats. By simulating the forensic data, the ground truth is known on what files the test results should find; however, for accuracy assurance, the forensic data of the storage device was additionally scanned with Foremost to validate what results the application should find. Whilst the size of the forensic data used in this study is small in comparison to the size of typical modern day storage devices; the forensic data used in the study is large enough to exhaust the cache on all tested machines, providing processing rates comparable to that achievable with analysing larger amounts of forensic data.

In order to test the efficiency of each processing configuration, three different searches were conducted. These searches consisted of searching for a series of 5, 19, and 40 different patterns, with each pattern identifying a different file header— which comprises of a series of bytes which identify the start of a file. The completion of each search of patterns would return with the total time it took to search the forensic data, and additionally the total of each pattern found within the forensic data. The time taken to complete the analysis would be recorded and the results were validated with what was known about the forensic image. Each series of patterns were searched for 5 times each, in which it was observed that each of the 5 times produced shown minimal variation of up to one second. Due to the consistency of times produced, the mean average time was used for analysis. Effects of data caching were eliminated by clearing the computer's cache between each test.

4. EXPERIMENTAL RESULTS

In this section, the performance of utilising multiple GPUs to help string searching will be presented and discussed. The time that each test platform took to conduct string searching on the three test platforms are presented in table 2. From this table, it is clear to see some patterns emerging between the three computers which took the test.

Table 2
Time taken to conduct string searching (secs.)

Test Platform A

	Single CPU	Multi-CPU	Single GPU	Multi-GPU	Single IGP
5	367	72	110	61	n/a
19	1897	385	125	61	n/a
40	3858	781	141	62	n/a

Test Platform B

	Single CPU	Multi-CPU	Single GPU	Multi-GPU	Single IGP
5	400	113	115	85	136
19	1907	485	125	91	192
40	3888	992	141	110	278

Test Platform C

	Single CPU	Multi-CPU	Single GPU	Multi-GPU	Single IGP
5	392	71	57	35	72
19	2172	456	74	53	151
40	4458	924	98	76	267

Platform A, which had 4 discrete GPUs working in parallel in the multiple GPU test, produced the least time variation when searching for all pattern amounts; however, when only one of those GPUs were used in the single GPU test, the time taken gradually increased with the more patterns searched. Comparing the decline in performance between the 5 and 40 pattern searches, it is observed that the single GPU was still more capable than the multi-threaded CPU at processing higher amounts of patterns. The single GPU required an additional 21.9% time to search for the 35 extra patterns, which is significantly less than the 90.7% extra time that the multi-threaded CPU required utilising all 8 threads. The variation in time to search for greater amounts of pat-terns from both CPU implementations were expected due to the limitations of the algorithm used; however, it is acknowledged that it may have performed better with the use of a different algorithm specifically designed for multi-pattern searching, such as the Aho-Corasick algorithm.

Both platforms B and C also show very similar trends in their results, however, results from the multiple GPU test vary significantly to that obtained from platform A. This variation could be explained by the fact that that platforms A and B used only a pair of graphics units in the multiple GPU test; which consisted of a dedicated GPU, paired with the CPU's IGP— the latter of which lacks the power of its discrete counterpart. Nevertheless, looking at single IGP performance, it boasts good results over CPU counterparts when tasked with searching for greater amounts of patterns. What is interesting to observe, however, is how considerable the IGP contributed during the multiple GPU tests; on average, the addition of an IGP improved reduced the time required to search by 27% over a single dedicated GPU.

In contrast to the GPU results, the CPU results scaled very linearly with the amount of search patterns defined, with the single CPU tests producing the weakest times as anticipated. Results from running the modified Boyer-Moore algorithm over many threads in the multi-threaded CPU test saw analysis time drop to around a quarter of the time required by single-threaded CPU, and still demonstrated a steep linear increase in time and the amount of patterns searched for. This result was expected, however, the significance of this result demonstrates a limitation on how fast the CPU can, at its fullest capacity, search for multiple patterns in data. Whilst the authors acknowledge that the CPU results could hypothetically be improved upon with the use of a different algorithm; it could still be argued that the same trend would exist due to the underlying architecture. The CPU would still not be as capable of processing large amounts of data with repetitive instructions at the same levels that are produced by GPUs, and multiple GPUs respectively

Figure 3 presents the processing rate (pr) from each test, which were calculated by $pr = \frac{Size\ of\ data}{Time\ taken}$ to show the results by mebibytes per second— allowing easier comparison to the sequential read performance from each computer's storage device. From these results we can see the effects that storage device transfer speeds have on the performed tests and the times produced. Platform A's multiple GPU tests indicated that the results were limited to storage device transfer, and not by GPU performance limiting factors. Likewise, platform B, during the 5 pattern search appeared to have been affected by the same limitations; however, platform C's superior storage device performance showed no obvious slowdown to the processing times produced by the processing techniques.

Reflecting on the processing rate, it would be reasonable to exclude the transfer limitations of the storage devices when considering the overall better processing option, as the single- and multiple-GPU tests show significantly better performance than both CPU tests when string searching.

.

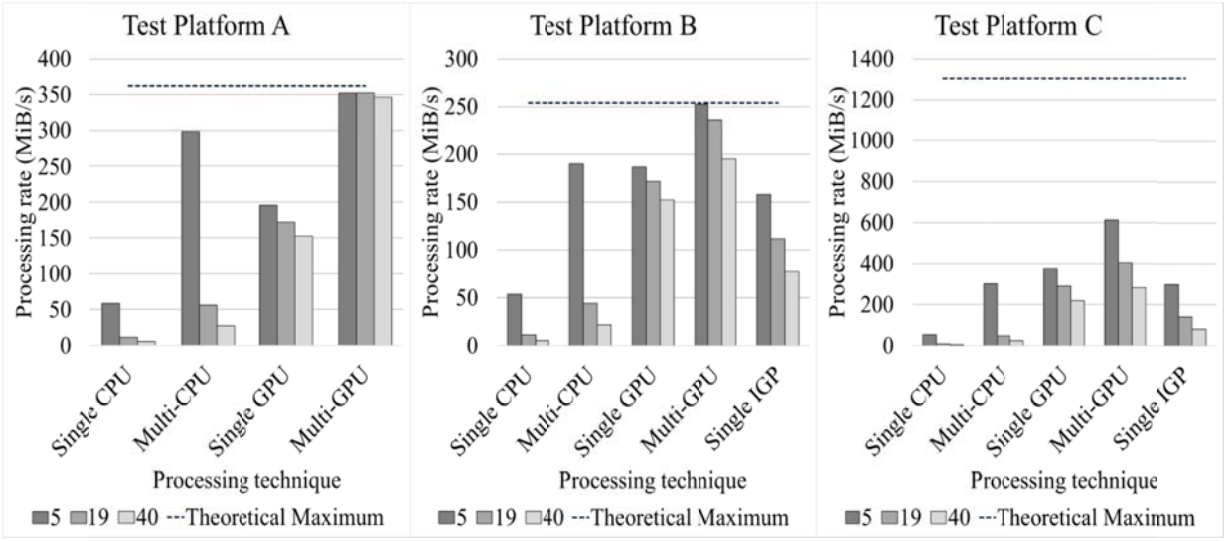

Figure 3. Processing rate conducting string searching (MiB/s)

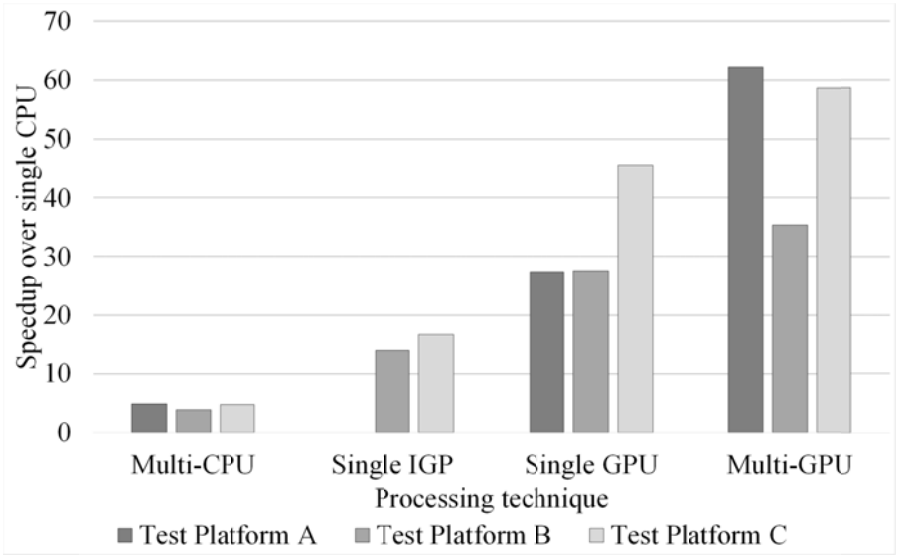

Figure 4. Speedup using single CPU as baseline

Lastly, figure 4 shows the speedup (sp) that each test (t) offered over the single CPU test (cpu) performed on each machine ($sp = \frac{cpu}{t}$). Trends appear throughout and show significant improvement between the tiers tested, with the exception of test platform B's multiple GPU test, which the potential speedup was limited by the effects of storage

drive transfer speeds. Nonetheless, analysing the overall speedups, we can observe significant improvements when applying multiple GPU processing to aid in the task of string searching.

5. CONCLUSION

Multiple GPUs have shown to be a fast and reliable solution to handle string searching, a vital component used in DF tools and methods to analyse and extract data. Despite the effects of storage device transfer rates, analysing with multiple GPUs – or a dedicated GPU paired with an IGP – provided a significant speedup to other processing methods tested, whilst showing the least degradation when tasked with increasing numbers of search patterns. Whilst the conclusion of more processors equate to faster searching may not be deemed a novel finding, it signifies the very processing improvement that DF tools need to incorporate. Whilst the DF community may have pushed the capabilities of what is possible with CPU processing by employing a range of algorithms, DF tools require more powerful implementations to overcome the difficulties in processing larger modern sizes of data. GPU research in this field has not been fully explored, yet has demonstrated the most promise from this study and others.

The authors also acknowledge the clear limitations that storage device transfer rates have on the results produced from string searching, which introduced a limit to how fast all processing methods could process data. Despite this limit, however, the results of this study demonstrate that employing multiple GPUs to perform string searching maximises the potential processing rate, strengthening the authors' previous findings that the performance of DF tools are not limited by storage device transfer speeds— but rather a combination of storage device transfer speeds and processing power.

6. FUTURE WORK

Whilst undertaking this study employing multiple GPUs to analyse forensic data, it was evident that a more specialist GPU string searching algorithm would improve results significantly. The authors intend to research and conduct further trials of GPU processing with algorithms to attempt to achieve faster parallel searching through data.

REFERENCES

[1] Aho, A. V, & Corasick, M. J. (1975). Efficient String Matching : An Aid to Bibliographic Search. Communications *of the ACM*, *18*(6), 333–340.

[2] Ayers, D. (2009). A second generation computer forensic analysis system. *Digital Investigation*, *6*, S34–S42. http://doi.org/10.1016/j.diin.2009.06.013

[3] Bayne, E., Ferguson, R. I., & Isaacs, J. (2014). OpenCL Acceleration of Digital Forensic Methods. In *Proceedings of Cyberforensics 2014* (pp. 50–57). University of Strathclyde, Glasgow.

[4] Boyer, R. S., & Moore, J. S. (1977). A Fast String Searching Algorithm. *Communications of the ACM*, *20*(10), 762–772.

[5] Collange, S., Daumas, M., Dandass, Y. S., & Defour, D. (2009). Using graphics processors for parallelizing hash-based data carving. *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, 1–10. http://doi.org/10.1109/HICSS.2009.494

[6] CrystalMark. (n.d.). CrystalDiskMark. Retrieved June 2, 2016, from http://crystalmark.info/software/CrystalDiskMark/index-e.html

[7] Fang, J., Varbanescu, A. L., & Sips, H. (2011). A Comprehensive Performance Comparison of CUDA and OpenCL. *2011 International Conference on Parallel Processing*, 216–225. http://doi.org/10.1109/ICPP.2011.45

[8] Karimi, K., Dickson, N. G., & Hamze, F. (2010). A Performance Comparison of CUDA and OpenCL. *ArXiv E-Prints*, *1005.2581*, 12.

http://doi.org/10.1109/ICPP.2011.45

[9] Marziale, L., Richard, G. G., & Roussev, V. (2007). Massive threading: Using GPUs to increase the performance of digital forensics tools. *Digital Investigation*, *4*, 73–81. http://doi.org/10.1016/j.diin.2007.06.019

[10] Skrbina, N., & Stojanovski, T. (2012). Using parallel processing for file carving. *ArXiv E-Prints*, *1205.0103*(March).

[11] Zha, X., & Sahni, S. (2011a). Fast in-Place File Carving for Digital Forensics. *Forensics in Telecommunications, Information, and Multimedia*, 141–158.

[12] Zha, X., & Sahni, S. (2011b). Multipattern String Matching On A GPU. *IEEE Symposium on Computers and Communications*, 277–282.

[13] Zha, X., & Sahni, S. (2013). GPU-to-GPU and Host-to-Host Multipattern String Matching on a GPU. *IEEE Transactions on Computers*, *62*(6), 1156–1169.

