Exploring CPU, GPU, and TPU Algorithm Runtime Costs

Miles Lane, Nate Lindley, Noah Dunn
University of Virginia
cml4jut, nfl6fh, nad5na@virginia.edu

Abstract

As computer processor options and cloud computing systems continue to advance, it can be difficult to determine what the best architecture is for any specific problem. Newer GPU and TPU processors have proven themselves far better than CPU-only systems for certain tasks, and often when large datasets are being used it is easy to jump straight into using these advanced processors. In this paper, we explore the actual cost of running large text-based datasets through basic algorithms in order to determine if and when using newer processing architectures is actually worth it. With modern cloud computing solutions, time spent using compute engines very well translates into cost, and using specialized processors can demand chip-hour pricing many times that of a typical CPU. In our study, Wikidumps data files in XML format, ranging from several gigabytes (GB) to 93 GB, are processed using several algorithms designed for text-based operations. Testing has demonstrated considerable differences in algorithm speed for each processor due to both the optimization overhead for each type of processor and the need to stream large amounts of data across each computing architecture. Our results show that using CPU-only systems can outperform GPU and TPU-accelerated ones for very simple tasks like hashing and getting a word count, but this lead in performance quickly wanes as algorithms require more on-chip processing.

1 Introduction

The relationship between the use of 'regular' CPU architectures and those accelerated by GPUs or TPUs, with regard to the concept of algorithm runtime, is currently a large area of study. In this project, we aim to gather information comparing these three processing architectures with regard to their processing capabilities, speeds, and costs. More specifically, we will be analyzing their performance when using large sets of text data to compute basic statistics and outcomes. The growth of data in all aspects of life including healthcare, commerce, education, government functions, and more has provided a great opportunity for computationally analyzing the world. While processing speeds have continued to increase for traditional CPUs, as is predicted by Moore's Law, other architectures for computing have become more and more prevalent. Among these architectures are graphical processing units (GPUs) and tensor processing units (TPUs), each optimized for differing tasks.

Using large datasets, regardless of how complex an algorithm is, can cause processors to spend immense amounts of time performing computations. When using cloud computing platforms, this time translates into a dollar cost per time using any processing architecture. In recent years, the availability of advanced processors like GPUs and TPUs for general computing tasks has grown tremendously. Platforms such as Google Colab, Google Cloud, and Amazon Web Services make accessing these processors especially easy, particularly in academic settings. When facing a computing task that involves massive datasets, it can be easy to use a GPU or TPU regardless of the actual task at hand, simply because they are colloquially known to be faster.

This study evaluates the runtime performance of several algorithms on CPU, GPU, and TPU processors in order to compare their relative performance with a large XML dataset and address two main research objectives. The first objective is to compare the runtimes of GPU and TPU accelerated machines using simple algorithms in order to determine how much they accelerate the computing process. In this project, the three main computing tasks that will be run are a string hashing algorithm, a word count algorithm, and a language detection algorithm. The second objective is to determine the actual runtime cost of each processor, for each algorithm, when the price of the processor in a cloud environment is factored in. We hypothesize that the GPU and TPU acceleration will result in faster runtimes when compared to a CPU for all three algorithms, and that as the algorithms become more complex the benefits will become more pronounced. However, we also anticipate that the performance gain achieved by the GPU and TPU processor instances will not reach levels seen in applications like deep learning because a large portion of this project deals with streaming large datasets and minimal actual computation.

In the end, this project is intended to shed light on the true performance metrics of the three types of processors when it comes to tasks that GPU and TPU processors are not generally optimized for. The results will provide valuable insights into the performance characteristics of each processor with regards to the three algorithms being run. While there is a very large number of potential algorithms that could be used here, the gradual increase in processing needed for each of the hashing, word count, and language detection algorithms provides a baseline for comparison between the CPU, GPU, and TPU. The runtimes of each algorithm will not be evaluated in isolation, as the cost per runtime for each processor will also be a factor in overall efficiency. On top of varied algorithms, each processor will be tested with varying chunk sizes streamed into their memory. This is intended to add another layer of comparisons between the three processors, as each is optimized for differing batch sizes. This project's results can help researchers and practitioners make informed decisions about the most efficient approaches for processing large datasets in various applications.

2 Related Work

Unsurprisingly, there have been many studies on the relationship between CPU, GPU, and TPU processors for a variety of applications. In one major study, it was found that CPUs perform very similarly to GPU and TPU architectures for simple tasks like linear regression. However, as tasks become more complex the GPU and TPU architectures significantly outperform the capabilities of CPUs. Some of these more complex tasks include training and using Convolutional Neural Networks and Recurrent Neural Networks, both of which are deep learning implementations that require massive amounts of data processing. Within the realm of neural network training and other complex computations, TPUs generally performed better than GPUs for models with many parameters while GPUs worked better with simpler neural networks. The performance increase seen in TPU and GPU architectures is due to their hardware configurations allowing them to perform many tasks, such as matrix operations, in parallel. On the other hand, CPU architectures are generally multi-purpose and are not optimized for the more modern applications of massive machine learning models [4].

In a separate study, it was noted that CPUs are best suited for operations that require "high single-threaded performance" [3]. Computing architectures that only use CPUs will perform best with serial processing, but they also have a relatively low latency and large memory capacity. Architectures using GPUs, on the other hand, perform best whenever massive parallelism is possible, but they are only able to handle limited multitasking operations and have a relatively low memory. Architectures with TPUs are even more different, as they are designed for processes requiring large data throughput, large batch sizes, and matrix operations. However, TPUs have a high latency and are nowhere near as universal as CPUs or even GPUs, especially as they are designed to work only with Tensorflow due to its heavy composition of matrix operations [2].

Despite how important the specifics for how each processor performs computational tasks is, it leaves out half of the equation studied in our project. Our project is focused on not only processing Wikidumps text data, but also on streaming very large pieces of this data. One key issue that our project addresses is the overhead associated with the use of GPU and TPU processors when compared with CPU architectures - especially when the operations require frequent transfers of chunks of data. In a study between CPU and GPU total processing speed, when including the time taken for data transfer between the processors and data storage, CPU architectures could be far faster. When the data being processed by the GPU was held locally, it would nearly always outperform the CPU for simple and complex tasks alike. However, when the data needed to be fetched through a network, the CPU outperformed the GPU [1]. This type of scenario is very similar to what is occurring in our project, where each chunk of data being processed must be fetched into local memory before being processed. On top of data transfer, using packages like Tensorflow and Cuda libraries can incur even more overhead that must be factored into a program's runtime. This gives reason to believe that for tasks involving a large amount of data transfer relative to actual computation, CPUs may be able to outperform GPU and TPU architectures. However, if the total program runtime begins to lean more heavily towards computation instead of data transfer and library overhead, it becomes more and more likely that GPU and TPU architectures will outperform those of CPUs.

3 Setup

For this project, we used Google Colab base CPUs and added TPU and GPU accelerators. Since Colab was used, accelerators were brought on from TensorFlow functions such as "list_physical_devices" and "TPUClusterResolver." Google Colab's hardware specifications are not completely clear, as according to their website the GPUs used vary given availability. According to some research, the CPU instances vary between 1 and 2 cored Intel Xeon processing units.

The testing was done on an XML file of over 90 GB in size. This file was found on Wikidumps. We downloaded this file using the wget linux function onto a data storage device on Google Cloud Platform (GCP) that had a 250 GB capacity to ensure it would fit. The file downloaded as a 20 GB .bz2 file. We then unzipped the file using bzip2 -d on linux to decompress it to its full >90 GB capacity.

Originally, we created three instances in Google's Cloud Platform. The first virtual machine instance we created, which was intended to be a CPU instance compared against the other two hardware instances, was a Google n2-standard-8 instance. This means the hardware was of the N2 machine series, which is considered a "balanced" series optimized for general workloads. The "8" part of n2-standard-8 represents the 8 vCPUs as part of this instance. The second was a GPU instance, which included an NVIDIA T4, considered a versatile GPU. The third was a Google TPU v2-8 instance. The v2-8 is not as high functioning as the v3-8, but it is much more cost-effective. Due to budget constraints and Colab's versatility, we ultimately decided to use Google Colab.

The CPU and GPU instances were on the cents per hour order of magnitude, whereas the TPU instances were running on the order of multiple dollars per hour. This affected how much we were able to test the TPU given a budget constraint of $50 on the Google Cloud Platform. For instance, downloading the full XML file and unzipping it from its original .bz2 form took over two hours alone, costing $10. On top of downloading the data, we also attempted to configure Jupyter notebook. Downloading Anaconda on the instance, then upgrading necessary packages, debugging, and finally running the notebook also took multiple hours as we were working out issues along the process. Other issues that we ran into during the setup included installing and later allowing TensorFlow to access CUDA drivers on the GPU instance. Ultimately we were not able to resolve this problem.

Ultimately, our trials with GCP instances allowed us to be more successful with Colab as we understood the value of time in using cloud computing power. Google Colab allows free access to GPUs and TPUs for up to 12 hours, and we executed our three algorithms over three different chunk sizes on three different machine types (CPU, GPU, and TPU) within that allotted time space. Additionally, Colab is pre-installed with several data science libraries including TensorFlow, which caused overhead stress during our setup of the GCP instances. This greatly reduced the time budget stress on our project by freeing up time for algorithm testing.

4 Testing

The testing of three hardware types was originally done on an XML file of over 90 GB in size. However, early testing demonstrated a relatively linear relationship between the XML file size and ratio of runtimes between the three architectures. This means that using files of sizes varying from roughly 5 GB to 90 GB resulted in similar runtime comparisons between the three processors, and since this ratio was the focus of the project, continually processing the 90 GB file was unnecessary. With this knowledge and the necessity of running many tests, XML files of roughly 15 GB in size were used for the majority of the final testing. Due to the accessible memory of the hardware used being less than that of the file size, we created a streaming process for the hardware to load and unload the data in chunks. Originally we attempted to use iterparse from the Element Tree XML/Python library to do this but later found we could stream the data by reading chunks with a loop in base python. All algorithms were run by first splitting the XML dataset into chunks of varying sizes including 128, 512, and 1028 MB. Different algorithms required different chunk sizes due to the complexity of the algorithms. A more complex algorithm had a tougher time running a 1028 MB chunk size, we found. The algorithms then iterated through all chunks until the desired data size was reached. The algorithms we ran included word count, hashing, and language detection. All algorithms were tweaked to account for hardware differences across CPU, GPU, and TPU instances.

The CPU word count algorithm was made by running an iterative loop through chunks of XML text, splitting the chunks into words using base python, then taking the length of those chunks of words and adding them to a word counter. GPU and TPU word count algorithms were similarly constructed but used TensorFlow to split the data, get the number of words, and perform other small functions. All hashing algorithms were created using the hashlib library from Python. The CPU algorithm hashed a specific chunk of text, then returned the length of the chunk that was hashed. The GPU did this same process but also used the cupy library, which uses CUDA drivers to configure the algorithm to run on a GPU. The TPU was alike in the GPU in that it used the same hashlib features to hash, but used TensorFlow to configure the TPU for parallel processing. Lastly, the language detection function was run through langdetect for all three algorithms. The GPU also used numpy and cupy libraries to configure its many processors. The TPU tried using langid, a stand-alone language detection library as well as TensorFlow in parallel, but the process ended up being much slower than just using langdetect.

Other algorithms that we attempted to run were POS recognition, average sentence length, and specific word counts. These algorithms were thought to provide more data as to the effectiveness of different processors running simple algorithms. We were also aiming to test machine learning algorithms, and found a large csv on Kaggle with which to test them with, but decided not to as we found enough difficulty setting up and running our other algorithms. Ultimately, word count, hashing, and language detection were sufficient in providing an array of different algorithms that demonstrated the performance of all three instance types.

5 Results

Figures 1 and 2 display a comparison of the speed for each processing architecture to process a dataset for each of the three specified algorithms. In general, the algorithms on the abscissa become more compute-intensive from left to right. The ordinate demonstrates how each of the three processors compares to the CPU runtime. For this axis, 1 is set as the baseline for the speed of the CPU while the speeds of the GPU and TPU runtimes are compared to it. Lower numbers equate to a lower ratio between each processor's speed and the speed of the CPU, while values greater than 1 indicate the processor performed better than the CPU. Figure 1 displays a comparison of these runtimes when using chunks of 128 megabytes (MB) of data, while Figure 2 displays the same comparison when using chunks of 512 MB of data. Both figures show similar results, with the more computationally intensive langdetect algorithm noticing a positive impact from GPU and TPU acceleration.

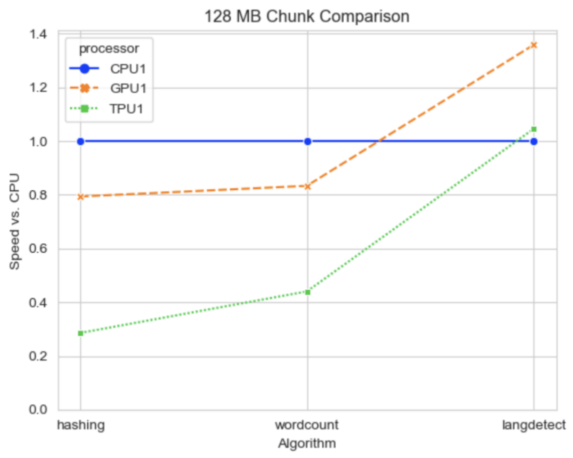

Figure 1: Processor Performance Against CPU by Algorithm, Using 128 MB Chunks

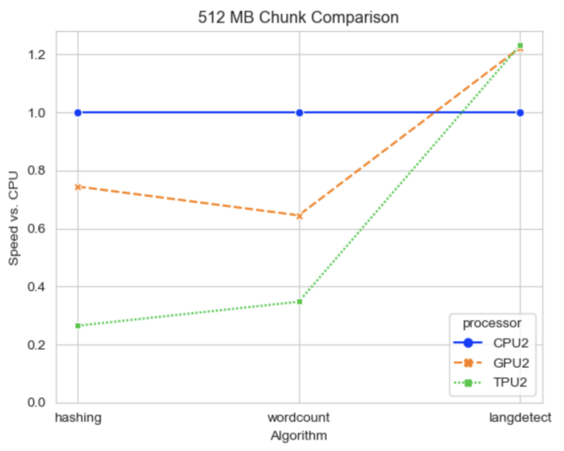

Figure 2: Processor Performance Against CPU by Algorithm, Using 512 MB Chunks

Cost estimates of the three hardware architectures and the three algorithms can be seen in Figure 3. In all of the algorithms studied, the CPU runtime was significantly cheaper than using either the GPU-accelerated or TPU-accelerated architectures. This was even despite the acceleration in efficiency when running the langdetect algorithm. We made cost estimates by extrapolating the runtime of our algorithm onto the full dataset from wikidumps and using the posted prices for various computing instances on Google Cloud.

Chunk Size	Algorithm	CPU	GPU	TPU
128 MB	Hashing	$0.033	$0.142	$2.280
	Wordcount	$0.018	$0.097	$2.168
	Langdetect	$0.259	$0.885	$10.581
512 MB	Hashing	$0.015	$0.134	$2.536
	Wordcount	$0.010	$0.103	$1.896
	Langdetect	$0.234	$0.890	$10.268

Figure 3: Pricing of Running Selected Algorithms on a 100 GB dataset

6 Conclusion

In our study, we found that GPU and TPU acceleration can provide significant speedups for certain algorithms and large datasets, but the performance gains depend on the specific task and the size of the data being processed. For example, hashing, which is already relatively optimized for CPUs, may not benefit as much from GPU or TPU acceleration as more complex tasks that involve large matrix operations, which can take advantage of the parallel processing capabilities of these devices. In addition, we found that the GPU and TPU provided quicker runtimes when using the same algorithms on larger chunks. In both of these cases, the overhead of transferring data to and from the GPU or TPU most likely outweighs the performance gains achieved by parallel processing.

These findings highlight the importance of evaluating the performance characteristics of specific algorithms on different hardware platforms. While GPU and TPU acceleration can offer significant performance gains for some tasks, their benefits may not extend to all types of algorithms and datasets. Understanding the strengths and limitations of each hardware platform is essential for making informed decisions about the most efficient approach for processing large datasets in various applications, including those related to machine learning and deep learning.

Looking forward, there are several avenues for future work that could build upon the findings of this study. One potential area for expansion is to investigate a wider range of algorithms and applications to determine which are best suited to accelerated computing. Additionally, future research could explore the performance of GPUs and TPUs when dealing with larger data sets, as well as examine the impact of different types of data structures and data processing pipelines. Overall, continued research into the capabilities and limitations of accelerated computing has the potential to yield significant insights and advancements in a wide range of scientific and technological fields.

7 Metadata

The presentation of the project can be found at:

https://github.com/nad5na/BDSFinalProject/blob/main/video1126429624.mp4

The code for the project can be found at:

https://github.com/nad5na/BDSFinalProject

The data for the project can be found at:

https://dumps.wikimedia.org/backup-index.html

References

[1] F. Fahim. CPU vs GPU vs TPU: Understanding the Difference Between Them. 2022.

[2] Hazelwood K. Gregg, C. Where is the data? Why you cannot debate CPU vs. GPU performance without the answer. 2011.

[3] Dimitrijević B. Nikolić T. Stojcev M. Nikolić, G. A Survey of Three Types of Processing Units: CPU, GPU and TPU. 2022.

[4] Wei G. Brooks D. Wang, Y. Benchmarking TPU, GPU, and CPU Platforms for Deep Learning. 2019.