Design and Performance Analysis of a Portable

Multi-GPU for Artificial Intelligence, Machine

Learning and Information Security

Foo Chang Jie#1, Muhammad Azfar Bin Adam#2, Nicholas Koh Wei Xuan#3, Wong Chong Peng#4

*#Information and Communications Technology (Information Security), Singapore Institute of Technology  
10 Dover Drive, Singapore 138683*

11902190@sit.singaporetech.edu.sg

21902165@sit.singaporetech.edu.sg

31902207@sit.singaporetech.edu.sg

41902128@sit.singaporetech.edu.sg

***Abstract*— Graphical Processing Units (GPUs) have been increasingly used in various IT aspects with examples such as Artificial Intelligence (AI), Machine Learning (ML) as well as Information Security (IS). With the increased usage of such tasks, it is critical to correctly identify the most efficient method to conduct these tasks. There are multiple ways in which GPUs can be connected to form a multi-GPU setup - Directly connecting it to the motherboard, via a PCIe switch and via riser cards. The aim of this experiment is to find the optimal setup to perform AI, ML and IS tasks in a multi-GPU setup. The method to determine the optimal setup would be via the use of industrial benchmarks based on the aforementioned fields. By performing the same benchmarks on the different setups, it would be clear to see the differences between each setup in their respective fields. Optimisation will then be done after analysis of results, where different experiments will then be conducted to further enhance the setup.The goal of this project is to obtain the knowledge on how GPU usage can be optimised for these tasks, with hopes of sharing with the community to further increase productivity in the respective fields.**

***Keywords***— **Multi-GPU, Information Security, Artificial Intelligence, Machine Learning, Benchmark, Hashcat, MLPerf, HPL**

1. Introduction

The usage of GPUs have evolved throughout the years. GPU used to be a specialized processor designed to specifically render graphics, and in recent years, have been used in many different aspects of IT. One example would be the usage of GPUs for cryptocurrency, and the rise of this caused the market to have a shortage of GPUs for the past few years.[1]

The ability to process multiple pieces of data simultaneously is a big upside for tasks such as machine learning and hash cracking. This led to people having specific setups built to fulfil this task. One good example would be mining rigs for cryptocurrency, with a multi-GPU setup focused solely on the task of cryptocurrency mining.

There are many different setups that people use for tasks such as hash cracking and cryptocurrency mining, while there are various different setups, which setup is the most effective at its role? Why are they the most proficient in that particular aspect? These are some of the questions that the team asks themselves while researching the various uses of GPUs. The answers to these questions would prove to be immensely useful, the productivity of tasks in the fields that are covered would increase, giving the community in general more work done in the same amount of time.

This work aims to provide the benefits of the different setups with relation to the specific IT fields - Machine Learning, Artificial Intelligence and Information Security. The goal of this experiment is to find the optimal multi-GPU setup for the aforementioned aspects of IT. Research has to be done beforehand to identify the setups that can be used for these tasks. To simulate the stress testing of tasks in these areas, industrial benchmarks will be used as a fair gauge on the different setups that will be selected for the experiment. Careful analysis and repeated experimentation is expected to assist the team in finding the optimal setup with valid claims. Results of the findings can be found at: <https://github.com/nickwx97/IS10-ITP>

1. Related Works

**Benchmarks**: Benchmarks has been a tool that is used to determine the performance of a specific setup. By performing the same benchmark with different setups, it will become clear which setup is the better one. However, as quoted from David Patterson “For better or worse, benchmarks shape a field”.[2] How would someone determine the validity of a benchmark? Would a single benchmark be the emphasis of technology advancement just because of a superior score?

It is important to understand how a benchmark works and what it scores off from before taking the scores at face value. It would be difficult to gauge the differences between different setups without a benchmark, thus the selection of benchmarks is important - for the correct usage for the correct judgement for the specific scenario.

1. Specifications

The experiment’s premise is limited to the hardware that we are offered. Table I contains the specifications of the setup used for this experiment. The hardware used for the experiments were not switched at any point in the experiment.

TABLE I  
Hardware Specification

| **Item** | **Details** |
| --- | --- |
| Motherboard | ASUS X99-E WS/USB 3.1 |
| CPU | Intel Xeon(R) E5-1650 v4@3.60GHz, 6 cores 12 threads |
| Memory | 131072 MB DDR4 2400 MHz (8x Crucial 16 GB 2400 MHz) |
| GPU | 4x ASUS TURBO-GTX1080Ti-11G |
| PCIe switch | PLX Technology PEX 8614 PCIe |

1. Setups

To understand how each setup behaves to the different aspects of IT, it is critical to know the different setups that exist in the community. The team has identified 3 different setups detonated as Setup A, Setup B and setup C.

Setup A will be treated as the baseline to all setups, as it involves the GPUs directly connected to the motherboard. This setup can be seen as the most common setup that is used to perform such tasks. The base temperature of the GPUs for this setup is around 45~50°C The actual setup used for this experiment can be found in Figure 1.



Fig. 1 Setup A

Setup B involves the use of a PCIe switch, extending to 4 GPUs via USB cables. PCIe switches are generally used to expand the amount of connected GPUs per setup. The PCIe switch is connected electrically to the motherboard via PCIe 3.0 x4 and the individual cards PCIe 3.0 x1. The base temperature of the GPUs for this setup is around 25~28°C. This can be seen on Figure 2, which was the setup used for this experiment.



Fig. 2 Setup B

The PCIe switch we used utilises a PLX Technology PEX 8614 chip. The port configuration for our switch is a PCIe 4x to 8 PCIe 1x in a fan-out topology. It has a reported maximum latency of 140 ns in this configuration. The switch uses a non-blocking internal switch architecture and has guaranteed error-free packets. It also has a max payload size of 2048 bytes and has an integrated 5 Giga Transmission per second(GT/s) Serializer/Deserializer(SerDes). Each port has auto-negotiation, lane reversal and polarity reversal. Lastly, it has a low power draw of only 1.74W. Below is a visualization of the switch’s port configuration.



Fig. 3 PCIe Switch Port Configuration

Setup C involves the use of riser cards, similar to a setup which normal mining rigs use. The individual cards are connected to the motherboard electrically via a PCIe 3.0 x1. The base temperature for the GPUs for this setup is around 25~28°C. This can be depicted in Figure 4, an image of the setup used for the experiment.

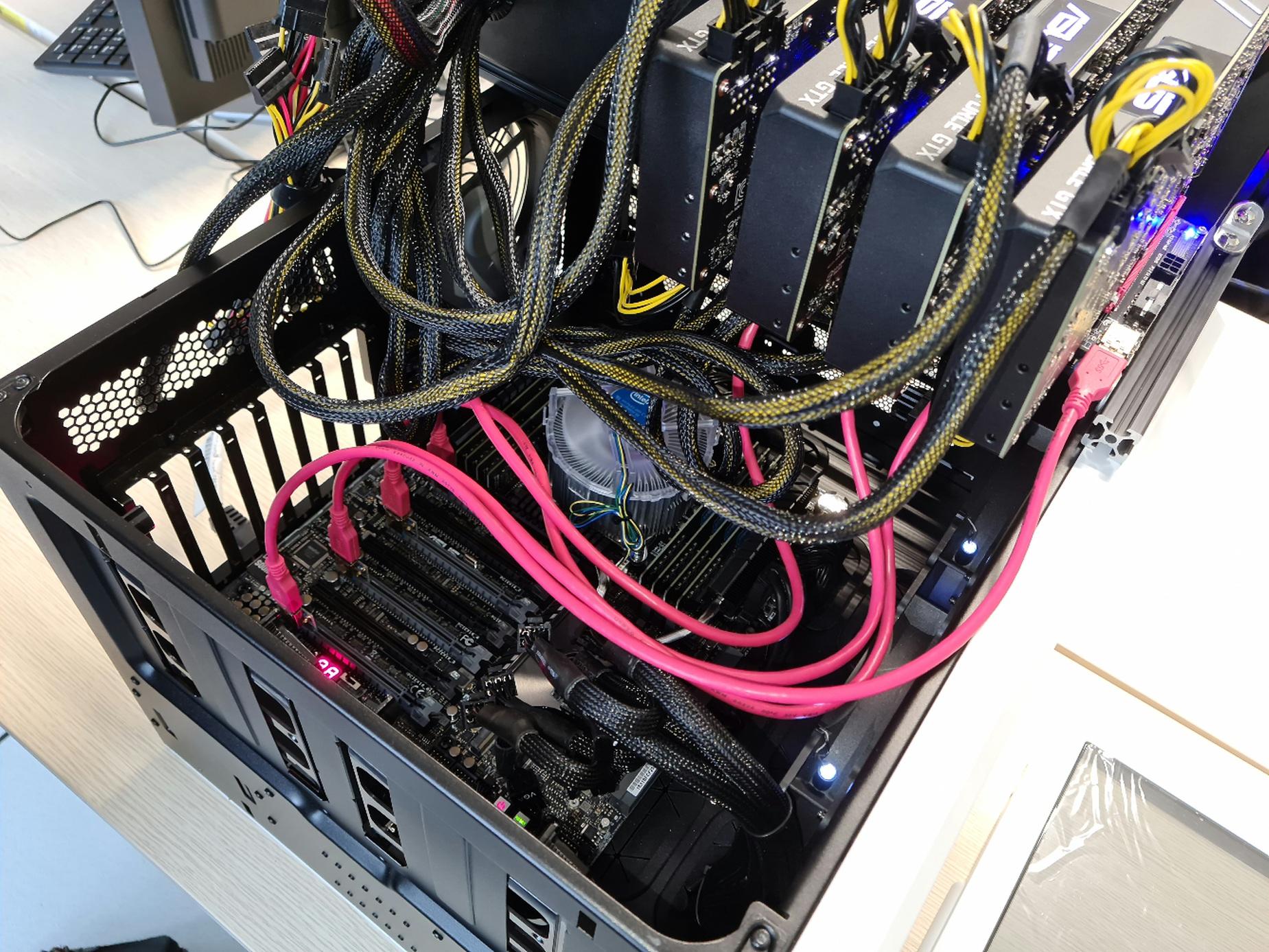


Fig. 4 Setup C

The above are the three setups that this experiment will be based on. It covers a wide range of methods where multi-GPUs can be used in a setup.

1. Benchmarks Chosen

As the scope revolves around Artificial Intelligence, Machine Learning and Information Security, the team decided to look into benchmarks that were in the aforementioned fields. After having more knowledge on benchmarking, the team felt there was a need to include an additional area to look at - Floating Point Computational Power. Most applications, including Artificial Intelligence, Machine Learning and Information Security will perform such tasks as part of their program. It is also a good gauge on the computational power outside of specific fields, which would allow this experiment to cover a broader area of the applications of IT. Thus, floating point computational power is incorporated into our scope.

The main selection criteria of the benchmark that the team will be using for this experiment is that the benchmark needs to be industrially recognized, such that this experiment can be seen as a valid attempt to simulate the tasks the benchmarks are doing. The benchmark also needs to be able to be run through multiple GPU units, as the experiment is based on setups with multiple GPUs.

After researching on the different benchmarks available in the industry, the team has decided to go with benchmarks that suited the experiment’s needs, with MLPerf covering the Artificial Intelligence and Machine Learning aspects, Hashcat, a benchmark used to calculate the amount of hashes generated under Information Security and Highly Parallel Linpack, a benchmark used to compute floating point computational power of a system.

1. Hashcat

Hashcat is a popular password cracker tool that has a built-in benchmarking tool that calculates the hashrate on each of the cryptographic algorithms. This benchmark would serve as a gauge for information security, where the speed of the hashes generated can be compared between setups. The main metric we will use to compare the setups is the hashrate, higher is better.

Table 2 shows the average amount of selected hashes that was generated by the three different setups over 10 runs individually. This is to provide the data with consistency and the results are as realistic as possible with a larger sample size. The standard deviation for the Hashcat data is considerably miniscule, where results were within range most of the time. Figure 5 would have a graphical representation of the data of Table II.

TABLE II  
Hashcat Results

| **Hash** | **Setup A** | **Setup B** | **Setup C** |
| --- | --- | --- | --- |
| sha512crypt | 770.58 kH/s | 776.85 kH/s | 769.83 kH/s |
| TrueCrypt P8KDF2-HMAC-RIPEMD160 + XTS 512 Bit | 1306.15 kH/s | 1316.01k H/s | 1314.9 kH/s |
| KeePass 1 and KeePass 2 | 727.69 kH/s | 736.24 kH/s | 729.99 kH/s |
| DPAPI masterkey file v1 | 364.67 kH/s | 374.13 kH/s | 373.04 kH/s |

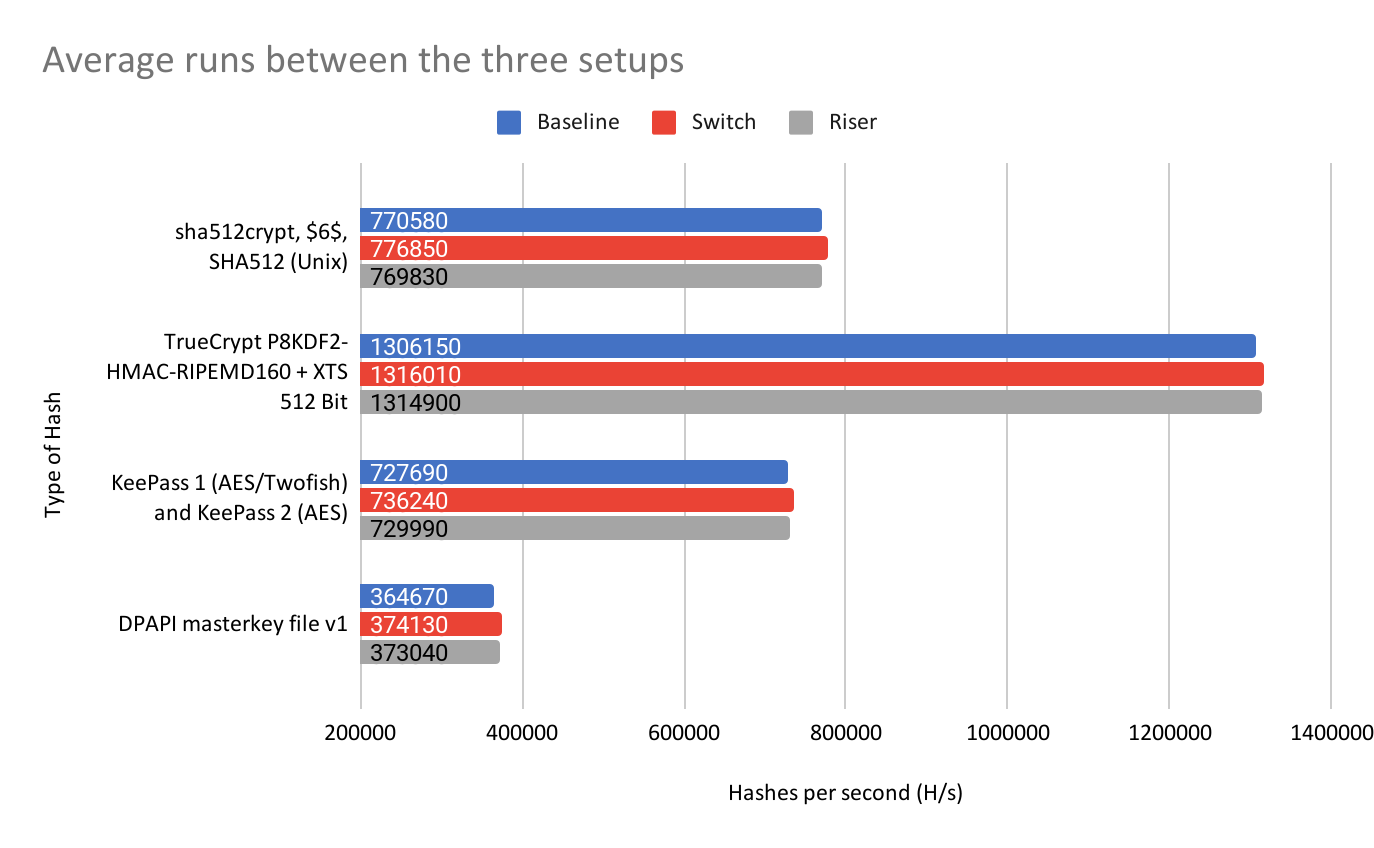
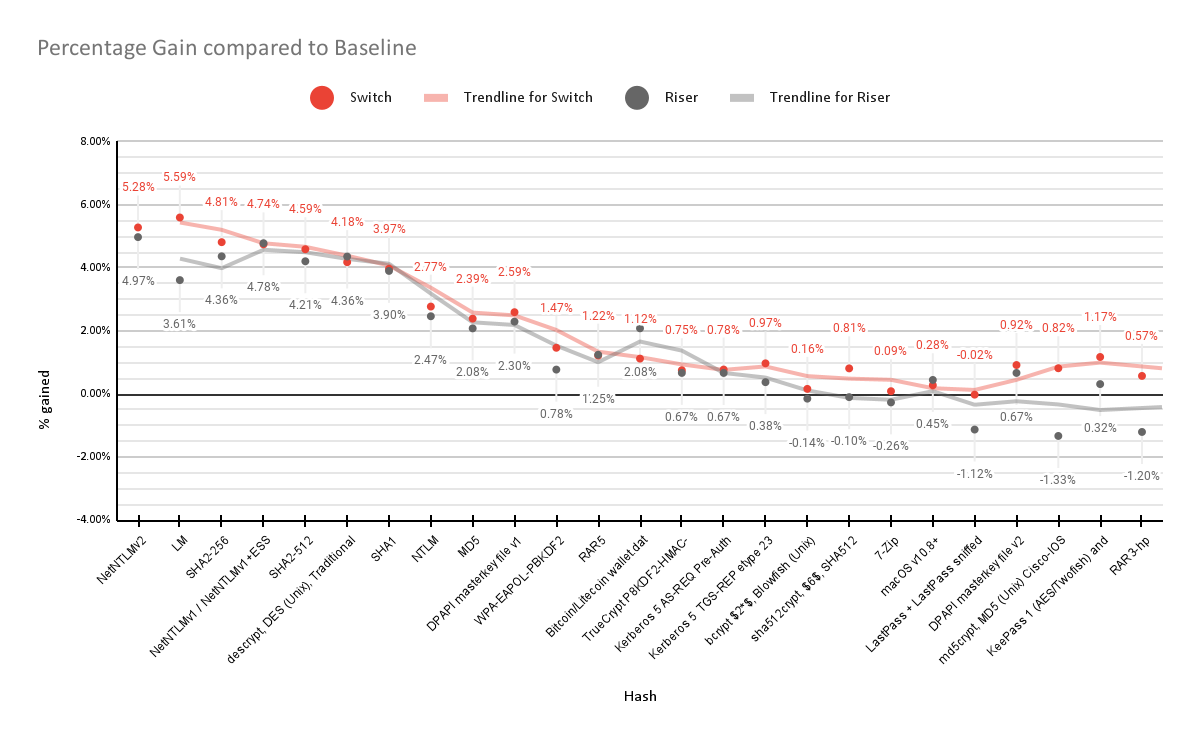


Fig. 5 Hashcat comparison between the three setups

A glance at this table would show Setup B being the most optimal setup for performing hash cracking. Out of the full hashcat benchmark test, Setup B had achieved 76% of the best results among the three different setups, Setup C had achieved 20% of the best results. whereas Setup A had achieved 4% of the best results. A clearer indicator for the full hashcat benchmark can be found in Figure 5, where Setup A sets itself up as a baseline for both Setups B and C. From Figure 6, the average performance gain of 2.08% can be seen for Setup B when compared to Setup A whereas an average performance gain of 1.61% for Setup C. We have noted that Setup C was not very stable. In some of the runs, the performance dipped a fair bit, resulting in a higher Coefficient of Variance (CV), in some cases, up to 4.98%. This will be discussed later on in the paper.

Fig. 6 Hashcat comparison with Setup A as the baseline

VII.HPL

Highly Parallel Linpack is a widely used benchmark that is suitable for parallel workloads that are core-limited and memory intensive. It is a portable and freely available implementation of the original HPC Linpack Benchmark. The original Linpack benchmark is designed single-node only, while HPL is designed for multiple nodes, utilising some sort of Message Passing Interface (MPI) like OpenMPI. It involves solving a random dense linear system in double precision (float64) arithmetic. It is used to populate the TOP500 list, a list of the world’s most powerful supercomputers, twice a year. The main metric used in the benchmark is the FLoating-point Operations Per Second, or FLOPS in short. This value is a measure of the system’s arithmetic computing capability, higher is better. We will be using NVIDIA’s CUDA accelerated xHPL for our experiments.

HPL can be configured and tuned for a system to achieve its best performance. Some of the variables will affect the performance more than the others. The main variables that affect the performance are the N problem size, NB block size, P number of process rows and Q number of process columns. N typically improves performance, given that the system has enough memory available. NB is the block size used for data distribution and computational granularity. Lastly, the product of P and Q must equal the number or MPI ranks in the run, which is determined by the number of GPUs in the system. Figure 77 shows a table with our configuration.

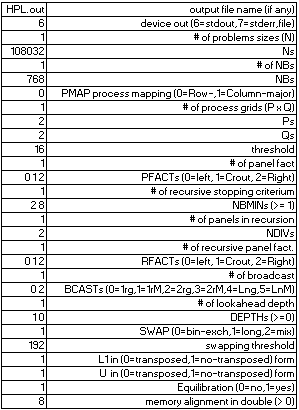


Fig. 7 HPL configuration file

With this configuration, we are able to achieve the following results over 10 runs.

TABLE III  
Benchmark Results for HPL

|  | **Average GFLOPs over 10 runs** | **Standard Deviation** | **% gain from baseline** |
| --- | --- | --- | --- |
| **Setup A** | 1,126.20 | 9.94 | 0% |
| **Setup C** | 233.25 | 33.40 | -79.29% |

We were unable to replicate the experiment on Setup B, as it would cause the entire system to become unresponsive. We have tried to run it with different N and NB values, and the highest N value we were able to run it on was N value of 16384 and NB value of 768. The N value is the problem size, and having a different value means we are unable to have a meaningful and fair comparison. We tried various methods of debugging this issue from checking system logs to live monitoring for anomalies using tools like htop and nvidia-smi but to no avail. We do not have a conclusive way of troubleshooting the issue due to limited hardware resources available to us, but we can rule out a few potential problems.

We ruled out limited bandwidth being a factor in causing the system unresponsiveness. Setup C also has each card connected electrically by PCIe x1s, so this cannot be a problem. Setup B was also able to complete the MLPerf benchmarks, which also requires a large amount of bandwidth. We also had ruled out stability issues. Setup B has proven to be rather stable from the other benchmarks as well as in the bandwithTest utility which will be expanded on later in the report.

VIII.MLPerf

For this experiment, the team has used two modules from the MLPerf, namely the Single Stage Detector, as well as ResNet. Both of these modules will be discussed in their subsection below.

1. *Single Stage Detector*

Single Stage Detector is an object detection framework, where it aims to detect multiple objects within a single image. This module in MLPerf aims to train a neural network model from a clean slate to a certain degree of accuracy. The model makes use of the COCO dataset, where it processes the images from the specified dataset and attempts to categorize them based on what the model believes the image contains. This process continues for 5 epochs, afterwhich a mean average precision (mAP) is calculated based on the predictions. The model takes the results into account and trains itself until a specified target is met, which in MLPerf’s case - 0.23 mAp. This benchmark is then repeated for 5 times to ensure the results were not skewed.

Given the target specified by MLPerf will be the same for the three different setups, the only logical way to compare them would be by time taken to complete the whole task, as well as the time taken per epoch, to further break the benchmark down.

The speed of which the setup takes to take in the image, process it and determine what objects are on the image itself will take a toll on its processing power. Table IV contains the results for the three setups on this benchmark and Figure 8 shows a graph comparing average time taken per epoch.

TABLE IV  
Benchmark Results for SSD

| **Setup** | **Average Time Taken(hh:mm:ss)** | **Average Epochs** | **Average time per Epochs(hh:mm:ss)** | **Standard Deviation** |
| --- | --- | --- | --- | --- |
| Setup A | 09:54:53 | 54 | 00:11:01 | 0:00:03 |
| Setup B | 13:14:08 | 53 | 00:14:59 | 0:00:03 |
| Setup C | 27:44:12\* | 55\* | 0:40:18\* | 0:11:58 |

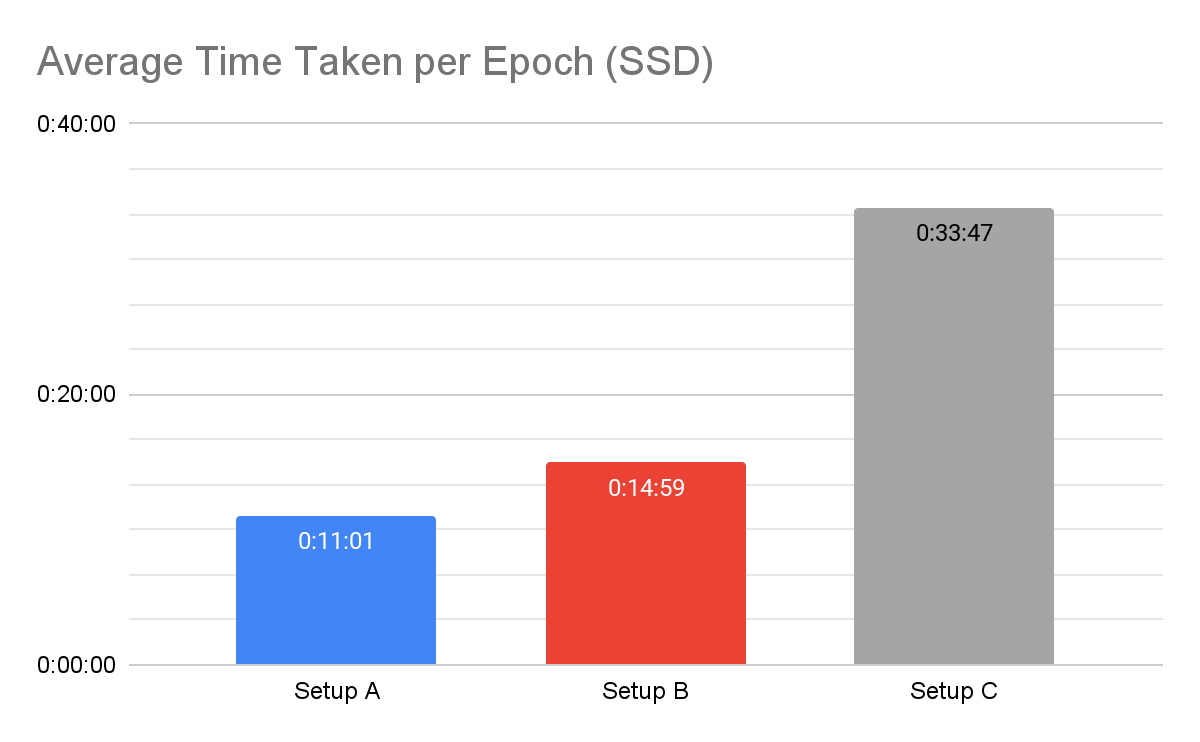


Fig. 8 Comparison of SSD timings per epoch

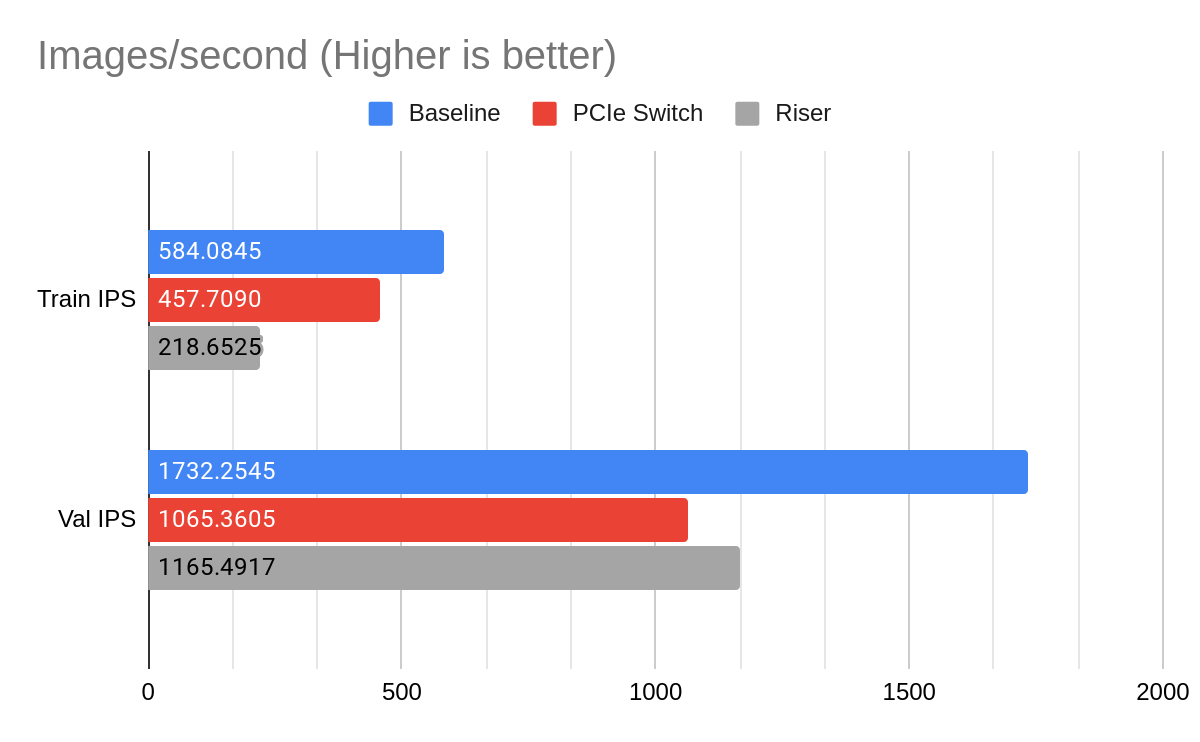
As Setup C has a high standard deviation, we will use its median of 0:33:47 for comparisons below.

The results for this benchmark for Setup C had an interesting result. The team has tried to run the same benchmark for Setup C for a total of 10 times, however, the benchmark has crashed a total of 9 times. The fact that the benchmark did not run successfully for the rest of the runs should be taken into account when determining the best setup for Artificial Intelligence and Machine Learning.

From the results on Table 4 as well as Figure 8, it is evident that Setup A is capable of performing tasks pertaining to training of a model better than Setups B and C. Setup A is estimated to be 36.01% faster than Setup B and 125.52% faster than Setup C.

1. *ResNet*

Resnet50 is a convolutional neural network model that performs image classification in computer vision. The resnet module in MLPerf’s benchmark calculates the training and inference performance using throughput (images/second) as metric, and number of epochs required to hit a top1 accuracy of 75.9% when training the model. We are using the ResNet-50 v1.5 model with MxNet in our experimentation and images per second as metric. The benchmark also uses the ImageNet1k as the dataset. The benchmark was run 3 times, with 3 epochs each, and 500 iterations per epoch. The parameters were used for all 3 of the setup to ensure consistency.

Fig. 9 Resnet50 Benchmark throughput results

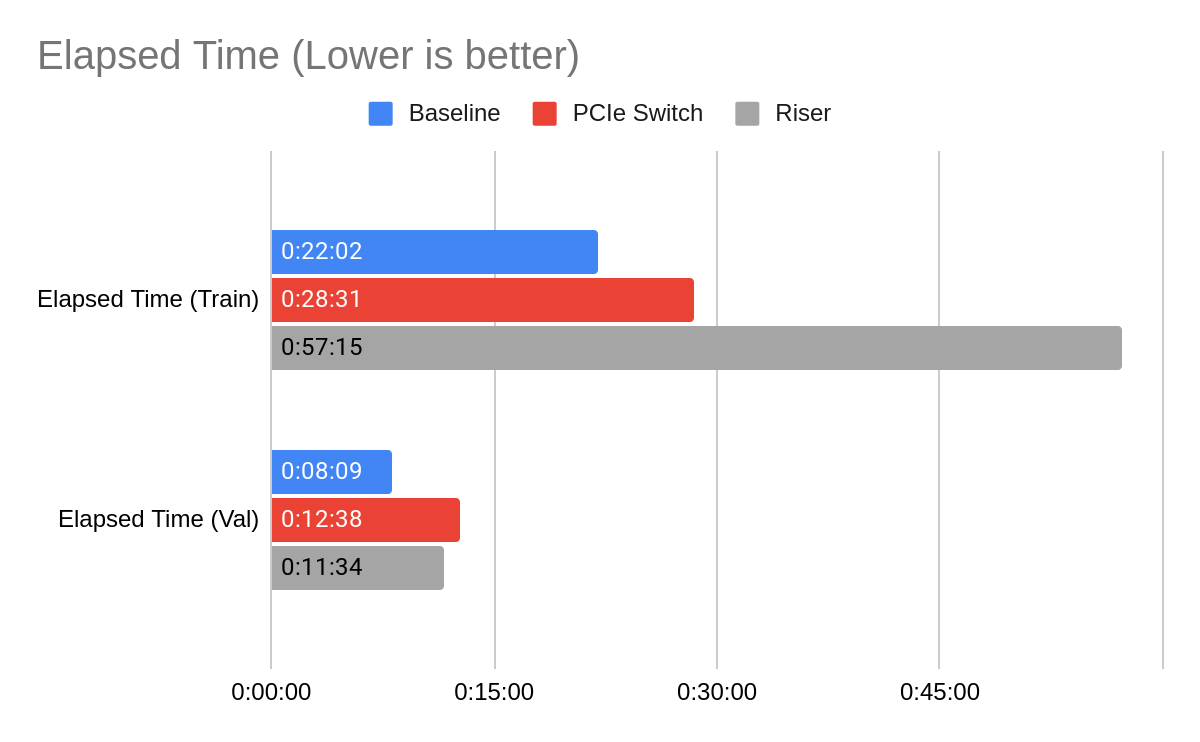
Fig. 10 Resnet50 Benchmark elapsed time results

Fig. 9 and 10 shows a 22% and 55% decrease in training throughput for Setup B and C respectively, and a 41.9% and 159.8% increase in training time for Setup B and C respectively. The inference results show a 38% and 32.7% decrease in inference throughput for Setup B and C respectively, and a 55% and 41.9% increase in inference time for Setup B and C respectively. Overall, Setup A has achieved the best results of the three setups for both training and inference as Setup B and C has a huge drop in performance. However, the results also show an interesting finding that Setup B performed better than Setup C in training, but was performing poorer in inference. This irregularity was caused by the data augmentation in the preprocessing pipeline.

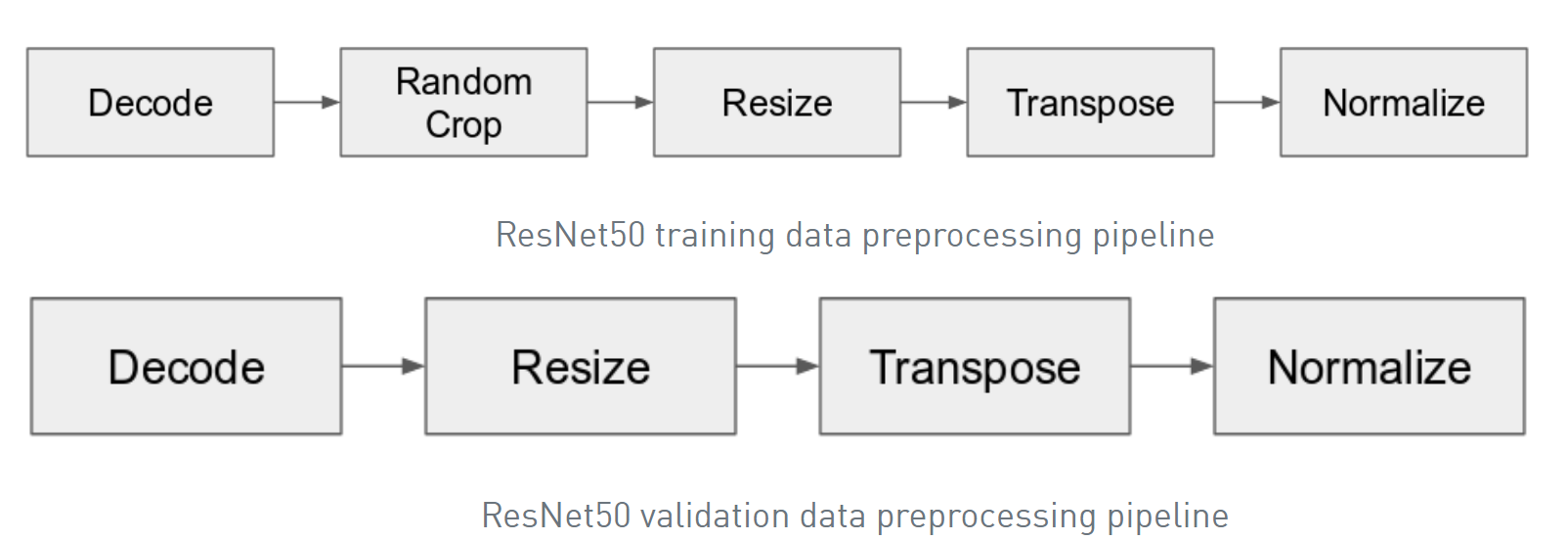
Resnet50 has two preprocessing pipelines, one for training and one for validation. The training preprocessing pipeline includes 7 data operations which are: Read raw image, Decode, Resize, Random Crop, Random horizontal flip, Transpose, Normalise. The validation preprocessing pipeline is similar to the training pipeline with the Random crop operation as the only exception; Random crop is not performed in validation. 

Fig. 11 Resnet50 data preprocessing pipeline

Data augmentation increases the amount of data and reduces overfitting on models. A reduction of one operation in data augmentation would result in faster preprocessing. This means that validation throughput would usually be higher than the training throughput. The reduced preprocessing operations in validation allows Setup C to perform better. The drastic performance improvement goes to show that Setup C was previously bottlenecked by the preprocessing operation in training which involves greater bandwidth than validation.

IX.Analysis

After obtaining the results from the benchmarks, the team began analysing the results and brainstorming on the factors that could affect the results and how we can alleviate them. The subsection below shows the factors that the team noticed and did more research and experiments onto them.

1. *Bandwidth*

After comparing the results from HPL and the MLPerf modules, we saw a big difference in performance between setups. This is attributed to the difference in the number of PCIe lanes. Setup A performed the best in these benchmarks and each card is connected via PCIe 3.0 x16, whereas setup B’s PCIe switch is connected to the motherboard via PCIe 3.0 x4 electrically and each card is connected to the switch via PCIe 3.0 x1 electrically, effectively bottlenecking the connection to PCIe 3.0 x1. Setup C is connected to the motherboard directly via PCIe 3.0 x1. The theoretical maximum throughput for PCIe 3.0 x1 is 0.965 GB/s whereas the theoretical maximum throughput for PCIe 3.0 x16 is 15.754 GB/s. This is also observed in our setups when we ran a utility by NVIDIA called bandwidthTest. The results from the utility can be seen below.

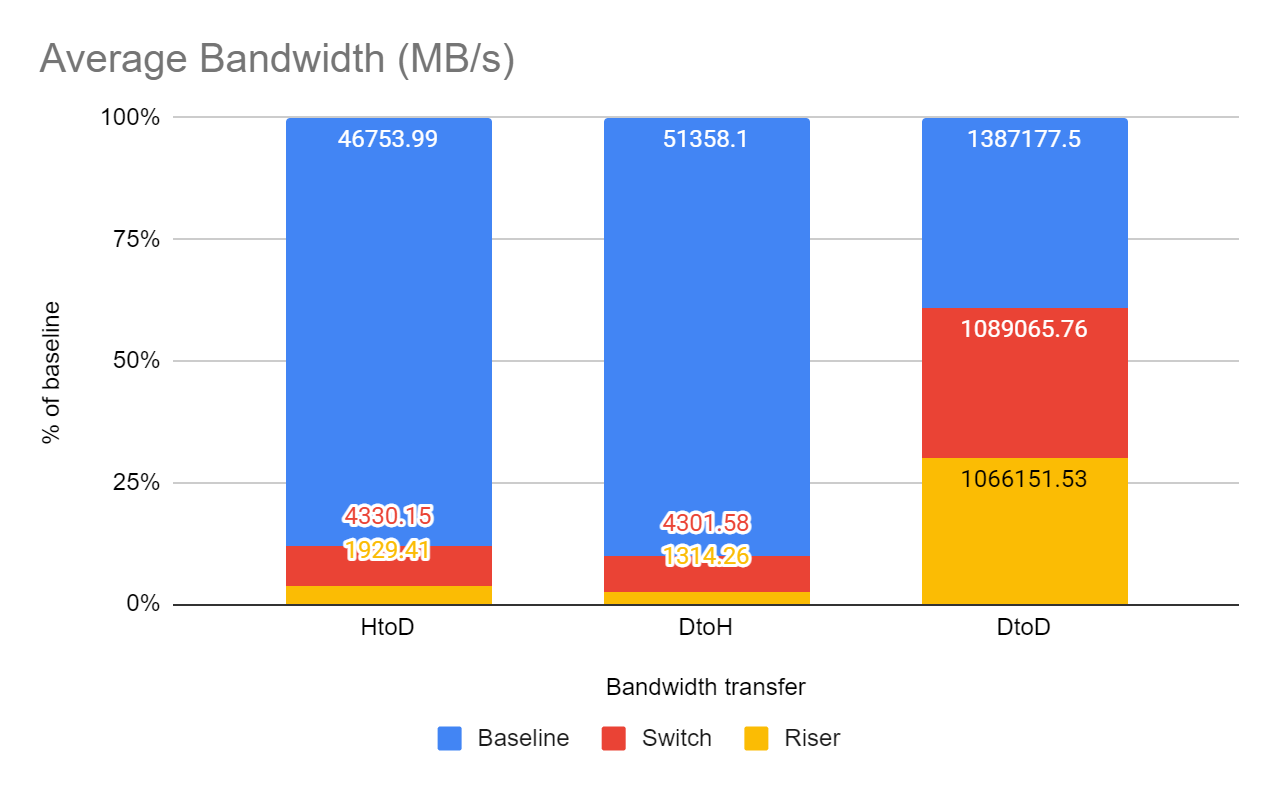


Fig. 12 Effective Bandwidth Between Setups

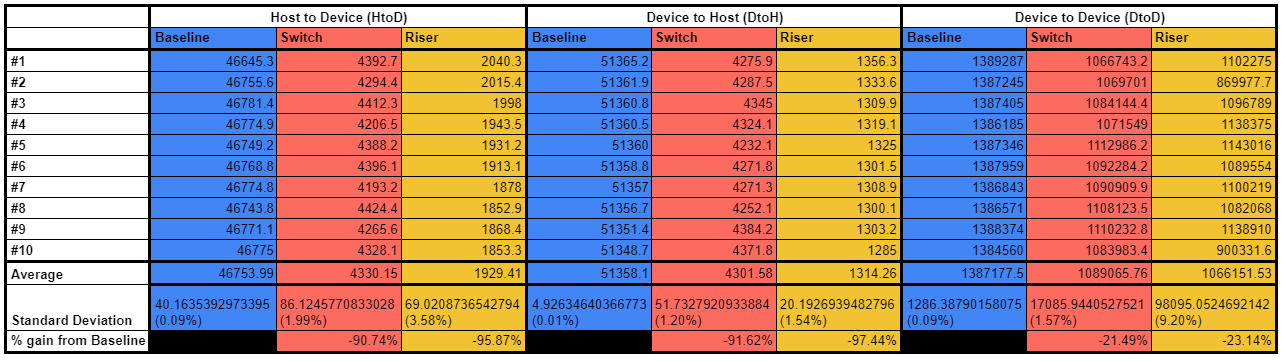


Fig. 13 Overview of three setups

We observed that both setup B & C had lower effective bandwidth than setup A which is expected. We also observed that setup C had lower effective bandwidth compared to setup B, which was odd as both are effectively PCIe 3.0 x1. In order to find out why, we ran the bandwidthTest again while monitoring using nvtop. Nvtop stands for NVidia TOP, a htop like task monitor for NVIDIA GPUs.

We observed that in setup C, the transfer speed would fluctuate a lot during the test, as compared to setup B where it is more stable. This leads us to suspect that setup C is unstable, which would also explain the higher standard deviations in the benchmark results, as well as the crashes that would occur during the single stage detector benchmark. We tried reseating the riser cards, using different slots and changing GPU positions, but to no avail.

1. *Instability of Setup C:* We theorised that the problem might be coming from the way the riser cards are connected to the motherboard. As the riser cards are physically PCIe x1 connected to the motherboard’s physical PCIe x16 slot, there is a possibility of a bad connection. There isn’t a way for us to lock and hold the riser cards in position. The connection between the riser cards and the motherboard was weak enough that a light tug would fully disconnect them. During system boot, we would also occasionally get a BIOS error code 96, which is “PCI Bus Assign Resources”, which could mean it has a flaky connection with the GPUs. However, we do not have the hardware necessary to conclusively determine if this is truly the case.

We did further research and also realised the possibility of the difference in the way the chipset and switch handles data distribution, which would affect the effective bandwidth of the setups. Subsection C will cover this in further detail.

1. *Temperature*

After obtaining from hashcat, we realised that temperature of the GPUs can be a factor affecting the results. This is because we expected Setup A, our baseline, to perform the best in all the experiments, as we expected the other setups to be limited by theoretical bandwidth by going from PCIe x16 to x1. This was not the case in hashcat as our results have shown. We re-ran hashcat on all setups and closely monitored the runs from all setups with nvtop this time.

We noticed that the temperatures in Setup A were a lot higher than the other setups, ~79-85°C as compared to ~55-79°C. The difference in temperature within each setup was also noticed to be due to the position of the card relative to the others. We confirmed this by swapping the card positions with one another. This led us to think that the temperature was affecting the performance in setup A, due to the case limiting the airflow to the GPUs. We did further research and found evidence that our GPUs which were NVIDIA Pascal cards, had a technology called GPU Boost 3.0 by NVIDIA.

Temperature plays a huge part in GPU performance, due to NVIDIA’s method of dynamically boosting clock speeds until the cards hit power or temperature limits. This allows the GPUs to boost much higher than the advertised boost clocks. For the cards we are using, the boost clock is advertised at 1582MHz, and under sufficient cooling, we observed a core clock of about 1822MHz. This number fluctuates quite a bit as the boosting algorithm makes split second changes to a large number of parameters in order to keep the GPU at maximum possible boost frequencies. In GPU Boost 3.0, the target temperature is set to 84°C, which lines up with our observed temperatures.

GPU Boost has multiple conditions to provide the highest possible boost clocks, namely: power headroom, voltage, thermal headroom, boost binning, and thermal throttling. [3]

1. *Power headroom:* GPU boost will automatically overclock the card if there is enough power headroom available to achieve higher boost clocks, thus it is important that enough power is supplied to the cards so GPU boost can work as intended. It is also the most common limiting factor as the card will use up all the available power to hit higher clocks.

Power limit was not a concern for our setups as the maximum power draw of each of our 1080 Ti is 250W, and we have a 1600W PSU.

1. *Voltage:* The GPU’s power delivery system needs to be able to provide the additional voltage that is needed to hit and sustain the higher clock speeds. Voltage is a direct contributor to temperature as well so it ties into the thermal headroom condition as well. There is a hard limit to how much voltage the card can use and that limit is set by the card’s BIOS, usually around 1.0V+-. GPU Boost takes advantage of any voltage headroom to try and sustain the highest clock speed it possibly can.
2. *Thermal Headroom:* Thermal Headroom is the third major condition that has to be fulfilled in order to get effective operation of GPU boost. The boosting algorithm is extremely sensitive to the temperature of the GPU and will increase or decrease the clock speed based on the slightest change in temperature, thus the need of keeping the temperature of the cards as low as possible in order to achieve the highest clock speeds. Temperatures higher than 75°C will cause the clock speed to drop noticeably. Even though the clock speeds will still be higher than the advertised boost clocks, it’s not desirable to leave performance on the table. Our cards often drop clock speeds in order to keep it’s temperature in check, especially in the baseline setup where the temperature is high.
3. *Boost Binning:* GPU boost changes the clock speeds depending on all the above-mentioned factors in blocks of 15MHz. One block of 15MHz is known as a boost bin. It can be observed that the clock speeds only move in blocks of 15MHz, although it can drop multiple bins at a time.
4. *Thermal Throttling:* Pascal architecture cards and later like the 1080Ti we are using in the experiments usually do not undergo thermal throttling (TT). The cards only start to thermal throttle at a fixed temperature known as Junction Temperature Max (TJMax). For our card, that temperature would be 93°C. When the card reaches this temperature, the clock speeds will gradually drop until it falls below the base clock or until the temperature is below TJMax. The reason Pascal cards and later do not usually thermal throttle is due to GPU boost, which regulates the temperature by increasing and decreasing the clock speeds through boost binning to hit the target 80°C.

The main difference between TT and boost binning is that boost binning alters the maximum clock speed achieved by GPU boost using temperature data, while TT occurs at or below the base clock. As our cards do not hit TJMax, highest observed was 89°C during ResNet50 training, we do not undergo TT, instead we are dropping boost bins.

In order to confirm our theory that temperature was causing the drop in performance in the baseline setup and not some other factor that we missed, we ran 10 runs of hashcat with the fan speed manually set to the maximum. The results below are quite conclusive.

TABLE V  
Max Fan Speed Performance

| **Average Performance Gain of Max Fan v.s. Baseline** | |
| --- | --- |
| **Min Gain** | -0.07% |
| **Max Gain** | 9.70% |
| **Average Gain** | 5.51% |
| **% of cases improved** | 96.00% |

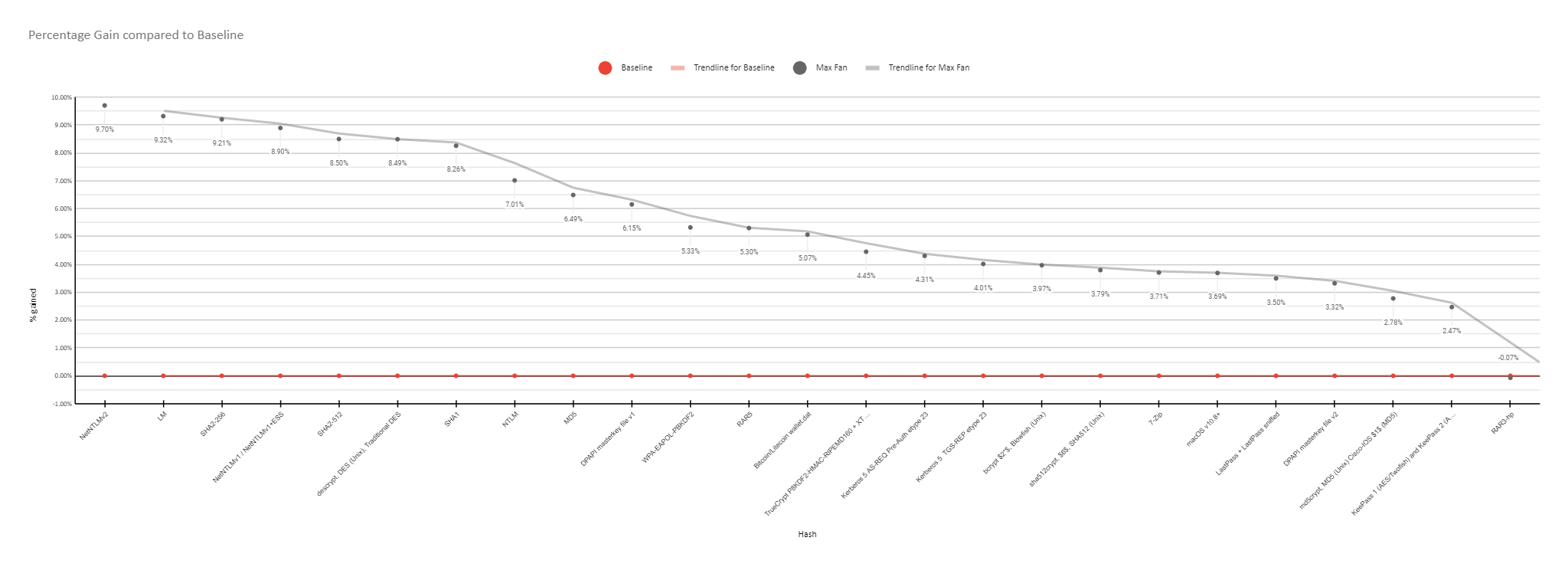


Fig. 14 Percentage gain compared to baseline

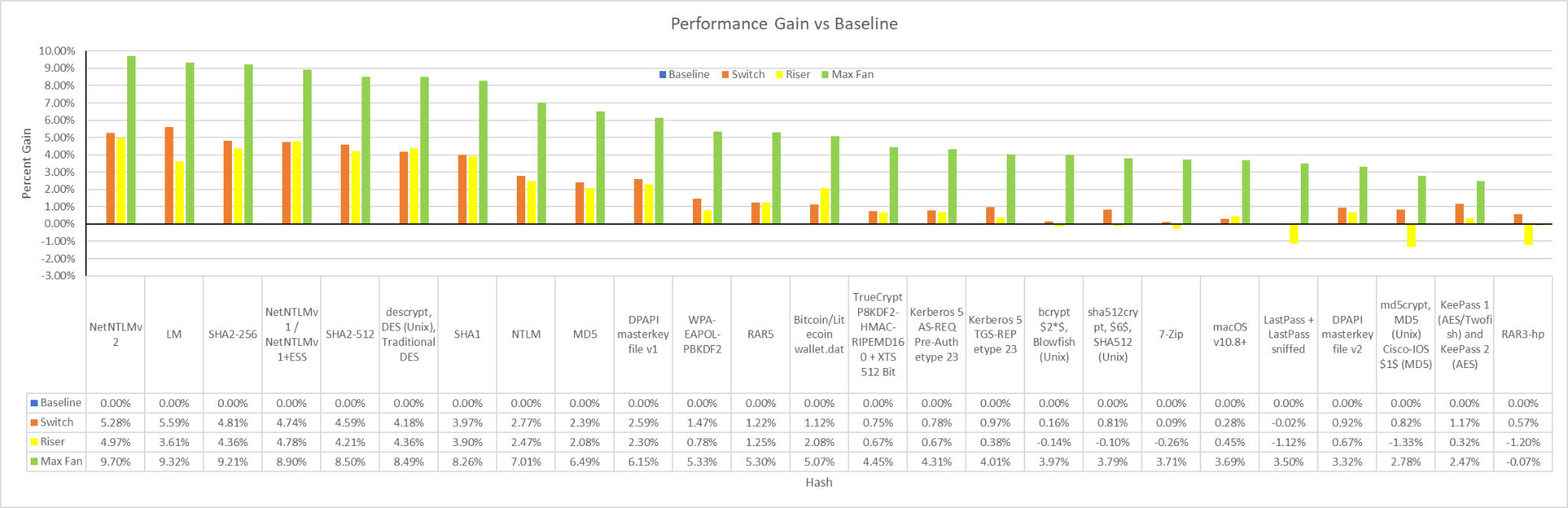


Fig 15 Combined chart with all setups

1. *PCIe Switching*

We observed that setup B would outperform setup C in most cases. We expected setup B to perform worse due to the data having to pass through more components as compared to setup C, thus theoretically should be slower due to latency and communication overhead, despite both being connected via PCIe 3.0 x1. Other than setup C’s instability discussed above, we realised that PCIe switching could also be a factor.

1. *How PCIe lanes are controlled:* There are multiple PCIe slots on a motherboard, but only the top-most one is controlled directly by the northbridge or host bridge. Historically, the northbridge was a separate chip connected to the CPU via a high-speed interface called the system bus or front-side bus (FSB). The northbridge controls the performance-critical components like the RAM and first PCIe slot [4], usually dedicated to the graphics card. Nowadays, the northbridge is integrated on the CPU. [5]

The other PCIe slots, also known as PCIe expansion slots, are controlled by the southbridge. The northbridge and the southbridge are the two chips that make up the chipset that we know of. The southbridge mainly implements the slower capabilities of the motherboard, like the BIOS, USB, PCIe expansion slots and others. The southbridge is connected via the northbridge, hence the other PCIe slots are usually slower than the first slot. [6]

Below is an illustration for how the slots are connected to the chipset.

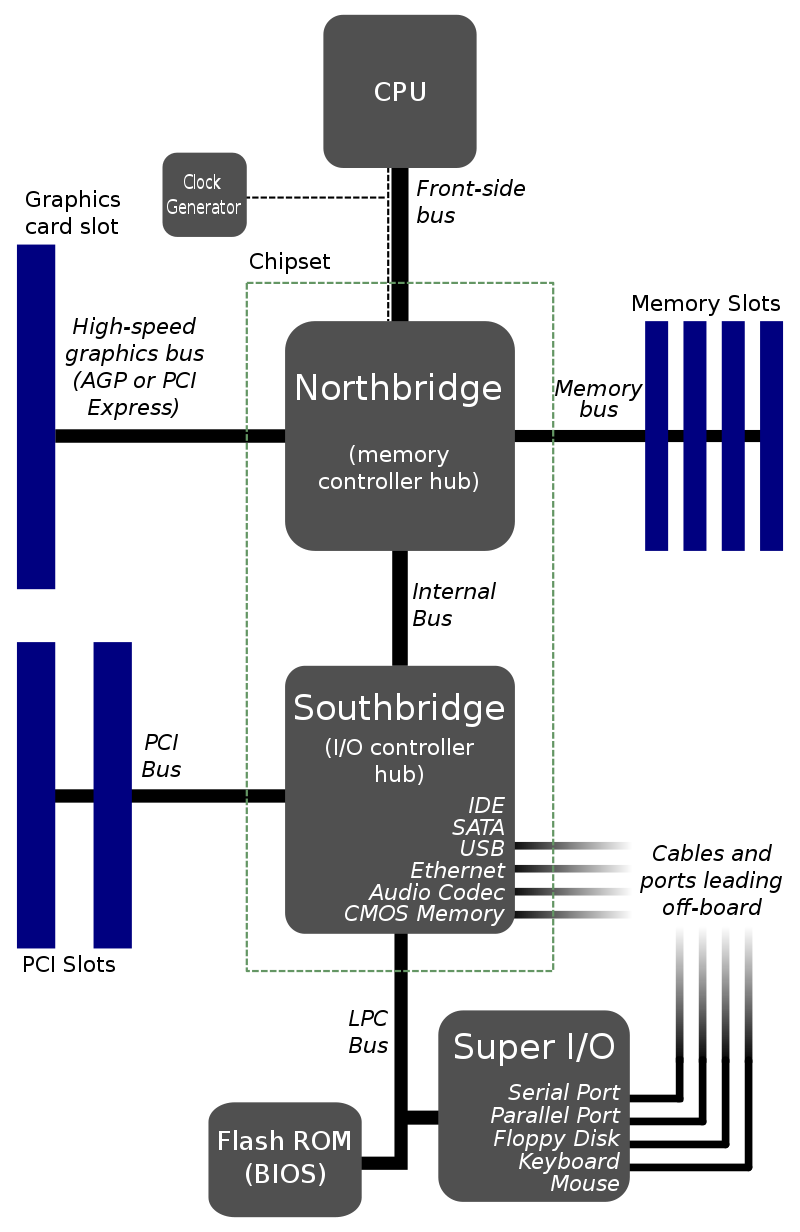


Fig 16. Diagram of chipset[7]

1. *Effect on our setups:* By moving the cards from the PCIe expansion slots to the switch, we are also moving the cards from the southbridge’s control to the northbridge and PCIe switch’s control. This might have caused the difference in performance between setup B and C, even though all the cards are effectively connected via PCIe 3.0 x1 in both cases.

However, we do not have the hardware necessary to conclusively determine if this is truly the case.

1. *Setup B inability to run HPL*

Regarding setup B, we attributed its inability to run the benchmark with the same configuration to hardware. To pinpoint the problem with setup B, we ran the benchmark on both setup A & C again while monitoring with nvtop. In setup A, we observed the peak transmit and receive can both reach about 11 GB/s per card across all cards.

We had a theory that the switch was being overwhelmed by the sheer amount of data being transferred. That was quickly debunked as we observed that the data transmitted and received during the run for setup C was scaled down to about 900 MB/s per card.

Setup B was also able to run and complete the MLPerf modules later on, which also utilizes a large amount of bandwidth. It is possible that the data distribution of the switch was incapable of high speed switching, as we observed the transmit and receive speeds would spike to peak then drop to 0 again within less than a second. However, this should not be the case as in the switch technical specification it is mentioned that the switch is capable of 5.0 GT/s Serialization/Deserialization (SerDes), which translates to 40 GB/s.

The problem is not with HPL either as the switch was able to run the benchmark, but only with N problem size 16384, while the others could run HPL at 108032 N problem size. N problem size represents the difficulty of the benchmark executed, thus it cannot be compared fairly against the other setups.

We can narrow down the problem to the switch and not the GPUs, as we tried to change the card positions such that different cards took the duty of displaying to the monitor. All of the attempts resulted in the system freezing up and black screen with no errors in the system logs or on nvtop when we used it to monitor for anomalies.

This leads us to believe that the switch was failing some way or another, resulting in no video output. Despite all our theories, we do not have a conclusive way of determining the problem, as we lack more hardware and time to test more scenarios to further narrow down the problem.

IX.Conclusion

.

We noticed that connecting to GPUs via riser cards and PCIe switches has a lower bandwidth than GPUs being directly connected to the motherboard. Based on our findings, lower bandwidth workloads can perform on externally connected GPUs with negligible performance impact due to cooler operating temperature. This limitation affects high bandwidth requirement tasks like AI/ML. As such, ‘mining-rig-esque’ setups are not ideal for such tasks.

With regards to bandwidth limitation, there are factors that can affect the bandwidth which in turn affects the throughput and latency of data transmission. We feel that using an incorrectly sized PCIe connector can cause instability. The inconsistent PCIe connections is an important factor to note which can affect the final outcome of this project.

As a result, further experimentation could be conducted to identify more variables that affect performance on external GPU setups. However, the concluding point to be made from our findings based on this project is that bandwidth limitation imposes a great deal on AI/ML tasks, though not so much on low-bandwidth tasks like hash-cracking. Therefore, based on our experiments, we cannot recommend externally connected GPUs for high performance computing and AI/ML tasks.

References

1. P. Staff, “Inside the gpu shortage: Why you still can’t buy a graphics card,” *PCMag Asia*, 10-Jun-2021. [Online] Available: https://sea.pcmag.com/graphics-cards/44196/inside-the-gpu-shortage-why-you-still-cant-buy-a-graphics-card. [Accessed: 28-Jul-2021]
2. D. Patterson, “For better or worse, benchmarks shape a field,” Communications of the ACM, vol. 55, 2012.
3. U. Saleem, "GPU Boost – Nvidia’s Self Boosting Algorithm Explained - Appuals.com", *Appuals.com*, 2021. [Online]. Available: https://appuals.com/gpu-boost-nvidias-self-boosting-algorithm-explained/. [Accessed: 28- Jul- 2021].
4. "Northbridge (computing) - Wikipedia", *En.wikipedia.org*, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Northbridge\_(computing). [Accessed: 28- Jul- 2021].
5. M. Kelvin, "PCIe lanes explained - how many do you need on a motherboard?", *Cots*, 2021. [Online]. Available: https://cotscomputers.com/blog/pcie-lanes/. [Accessed: 28- Jul- 2021].
6. "Southbridge (computing) - Wikipedia", *En.wikipedia.org*, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Southbridge\_(computing). [Accessed: 28- Jul- 2021].
7. "File:Motherboard diagram.svg - Wikimedia Commons", *Commons.wikimedia.org*, 2021. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=3789066. [Accessed: 28- Jul- 2021].