University of Victoria
Faculty of Science
Summer 2015 Work Term Report

Benchmarking the Performance of IaaS Clouds for HEP Applications

Department of Physics
University of Victoria
Victoria, BC

Matthew Murray
V00800801
Work Term 2
Physics and Astronomy
mamurray@uvic.ca

September 7, 2015

In partial fulfillment of the requirements of the
Bachelor of Science Degree in Physics

Contents

List of Figures

List of Tables

Benchmarking the Performance of IaaS Clouds for HEP Applications

Matthew Murray
mamurray@uvic.ca

September 7, 2015

Abstract

The field of High Energy Physics (HEP) has many very computationally intensive applications which are now being run on virtual machines on Infrastructure as a Service (IaaS) Clouds as an alternative to distributed grid computing. The University of Victoria HEP research group is heavily involved in using IaaS clouds to process data from the ATLAS collaboration. Due to the large workload of the HEP community, clouds used by the UVic HEP group were systemically benchmarked to determine which provide the best performance. Benchmarking data is helpful in monitoring the resources used by the UVic HEP group, in addition to appropriately distributing the workload. The benchmarks used for this report and the method of collection used are outlined. Then, the benchmark results for five Openstack clouds used by the UVic HEP group are examined and discussed for each of the benchmarks used.

1 Report Specifications

1.1 Audience

This report is intended for members and future co-op students of the University of Victoria High Energy Physics (UVic HEP) group, and to anyone else involved in cloud computing interested in the performance of IaaS clouds.

1.2 Prerequisites

A basic understanding of computer components and terminology is recommended. Knowledge of virtualization and cloud computing would also benefit the reader.

1.3 Purposes

The purposes of this report are to provide measurements of the performance of various Openstack clouds used by the UVic HEP group, to provide an overview of the methods used, and to examine the results of the data.

2 Introduction

High Energy Physics (HEP) is one of the most computationally demanding fields of science today. At the Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) in Switzerland, two proton beams rotating in opposite directions around the circumference of the accelerator are accelerated to extremely high energies. These beams contain an enormous amount of protons, and are made to collide at a few select sites around the LHC. One of these sites is the ATLAS experiment, which observes approximately one billion of these proton-proton collisions (events) per second, from which only 200 are selected to be permanently recorded[15]. Even after this, ATLAS alone generated 3.2 petabytes of raw data per year during Run 1, the operational period from 2010 - 2012[15]. After two years of upgrades, the LHC has just resumed collisions for Run 2 at a new record energy of 13TeV[10]. The collisions at this energy will be much more complex than previously observed, and will cause a significant increase in the amount of data produced by ATLAS, in addition to the time required to process it.

The data produced by ATLAS is distributed to physicists all over the world for analysis by CERN's Worldwide LHC Computing Grid (WLCG), a collaboration of computing centres across the world for processing data from the LHC. The WLCG is divided into tiers: the single Tier-0 is the CERN Data Centre, then 13 Tier-1 computing centers normally located in national scientific laboratories, and numerous Tier-2 centers located at universities or smaller research institutions[17]. The University of Victoria is one of the Tier-2 centers. Physics tasks such as the analysis, reconstruction, or simulation of events are then distributed to these centers to be performed on their computing resources.

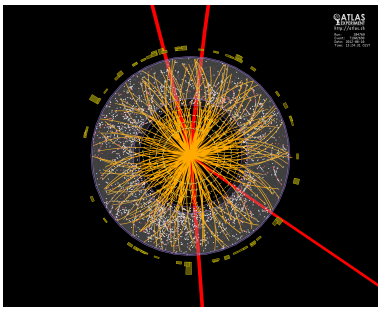


Figure 1: A reconstruction of a Higgs Boson candidate event. Copyright ATLAS Experiment ©2014 CERN.

The analysis and simulations, called jobs, that need to be performed on the LHC data number in the millions and are incredibly intensive operations. Tier-2 jobs take several hours or even days to run, and cannot be interrupted or stopped at any point without losing all progress. They also typically require a specific operating system, often outdated Linux packages, and specialized software from CERN not often available on the majority of modern operating systems. Therefore, having dedicated physical computing environments specifically for running ATLAS jobs is difficult, and scaling them even more so. As a solution, in recent years physicists have been executing jobs using virtual machines (VMs) on Infrastructure as a Service (IaaS) clouds to accommodate the computational requirements of ATLAS jobs.

A VM is an emulation of a computer existing in a layer of software on a host system. They are created, or instantiated, using an image containing a bootable operating system and any software the user needs. VMs can easily be shut down and removed altogether, and a new one can be started using a different image and configuration. This makes them easy to update and modify, and to be specifically configured for the applications they will be running. In addition, it removes the issue of needing systems dedicated specifically for running HEP jobs, and allows physicists to take advantage of computing resources that have not been preconfigured for HEP applications.

An IaaS cloud is a collection of networked servers (nodes) each running a hypervisor, a program that manages and allocates resources for VMs. IaaS clouds provide a large amount of computing resources available over the internet by allowing users to boot VMs on their hardware. They differ slightly from clouds often heard about in day to day use such as Apple iCloud®, which provide internet based services or storage. These are referred to as Platform or Software as a Service (PaaS and SaaS) clouds. Cloud computing is a much more flexible alternative to grid computing, which has been traditionally used by the WLCG.

To create VM instances, users must submit jobs to a batch job scheduling program such as HTCondor, which then will have a VM manager program such as the UVic HEP group's Cloud Scheduler send a request to a cloud to boot a VM with the requested resources. Once the request has been met, the job is run on the VM using the cloud provider's hardware. Cloud computing has proved to be very useful for physicists across the world to have access to a large amount of computing resources and a common, flexible platform for running jobs. In addition, the cloud computing model suits HEP applications very well as HEP jobs are embarrassingly parallel algorithms, meaning that with minimal effort they can be broken up into smaller problems that can run concurrently without needing to communicate with each other. This is easily visualized

in the case of HEP jobs, as the analysis and simulations consist of a large number of independent particle collisions.

The limitation, however, is that cloud providers are almost always external companies that serve many customers besides HEP physicists, and some require payment for use of their services. Given the large workload of HEP, it would be beneficial to determine the computing ability of various cloud providers, and the performance that can be expected from them. This could be used to help optimize the distribution of work and to more accurately plan job completion and deadlines. In order to accomplish this, clouds must be systemically benchmarked and compared in order to gain useful information.

3 Reasons to Benchmark IaaS Clouds

The UVic HEP group uses resources - that is, CPU cores, RAM, and storage - from a number of Openstack clouds as well as Amazon's Elastic Compute Cloud. Despite each one serving a similar purpose, each cloud is different in their hardware and configuration and are unlikely to have consistent performance. The performance of a VM on a cloud, or how quickly it can complete a job, is a complex subject with many variables.

A large factor is the hardware the cloud in question is running on. A VM running on a cloud has its CPUs emulated by a hypervisor; however, the speed and efficiency of these virtual CPUs (vCPUs) is of course limited by the hardware CPU on the node, which creates a stark difference among clouds using different CPU models. However, a single provider may not even use the same hardware throughout their service. If the cloud is using a combination of old and new nodes, a fraction of VMs booted on that cloud will be sourced on the newer hardware and the remaining fraction booted on the older hardware. This creates a performance discrepancy within a single cloud.

It is also necessary to take into account the many ways a particular cloud's configuration can differ. To name a few, it is possible the choice of hypervisor and scheduling software used on a cloud may have an effect on performance. Certain features offered by a cloud provider could have a considerable impact, in particular host-passthrough. Normally the hypervisor emulates a general vCPU that closely matches the model on the host, adding only a few of the host's model-specific features. With host-passthrough enabled, the hypervisor passes every low-level detail of the CPU model on the host to the VM's vCPU, allowing it to take full advantage of its features and increase performance[16]. Another feature sometimes offered is the use of solid-state drives to add swap space to their VM instances, which could potentially affect performance of certain applications.

In addition, a cloud may oversubscribe their CPUs, which is when the number of VM cores allocated is more than the number of physical CPU cores at the site. As such, the physical CPU time must be divided up among all the hosted VMs, which has a drastic impact on performance. This is less of an issue in most applications, since VMs normally spend much of their time idle. However, in HEP applications almost all of the available CPU time is used, and so the resulting performance hit would be akin to the VM having only a fraction of a CPU. Having hyperthreading enabled on a cloud does not alleviate the problem, as oversubscription can cause two VM cores to be sourced from two threads on a single physical core. However, without oversubscription hyperthreading does have the potential to increase performance by allowing the hypervisor to offload management tasks onto the additional threads.

The overlying message of the previous paragraphs is that there are many things that could *potentially* impact the performance of VMs on a cloud. It is not possible to know beforehand whether a particular characteristic will have an effect in practice, or by how much it will make a difference in performance. Therefore it is not useful to simply compare cloud providers with a list of their features and hardware, as such a comparison has no way of being proven or tested. It is necessary to consistently benchmark clouds in order to obtain quantitative information on their performance in practice. In addition, once this is in place, the benchmarking data would have many applications. The UVic HEP group and other computing sites could monitor changes in the performance of clouds and alert the providers, or allow the WLCG to determine which centres and countries contribute the most computing power. All things considered, benchmarking is a useful and necessary practice.

4 The Benchmarks

As mentioned above, clouds must be benchmarked in order to obtain reliable, quantifiable data on their performance. Benchmarks are usually designed to test a specific component of a system, such as the CPU performance or parallel execution performance, and are generally described as either a synthetic or an application benchmark based on their design. Synthetic benchmarks consist of doing a large number of iterations over a specific set of operations designed to stress the system being tested, and then measuring the elapsed time. Application benchmarks perform a particular, usually intensive task in an application that will be used in the operation environment and measure how long it takes to complete. The result from both is usually a single number representing either a number of operations per unit time, a rating on a predefined scale or the elapsed time.

It is important to note that the result produced by any benchmark test is indicative of relative performance. While many benchmark programs will exist to test the same component, they may vary in any of the number of operations in the loop body, the type and complexity of the operations used, or the number of iterations. As a result, it is not surprising for two benchmarks on the same system to give different results. This does not mean one of the benchmarks is wrong, instead that it is meaningless to compare the two results. Therefore, benchmarks should be used in comparing the performance of multiple systems, and to do so it is necessary that the same benchmark is used under the same conditions each time to have meaningful results. Furthermore, once must understand and know how to interpret the number returned by the program, including the units or if it is in reference to a particular machine.

HTCondor ships with two benchmarks, *condor_mips* and *condor_kflops*, which were used for the bulk of the data in this report. In addition, one other benchmark from the LHC beauty (LHCb) collaboration at CERN called *lhcbmc.py* was also used[14].

4.1 *condor_mips*

condor_mips is an implementation of the Dhrystone 2.1 integer benchmark[7]. Dhrystone is a benchmark that tests the system's performance when doing integer operations, as opposed to floating-point operations[11]. It consists of iterating over a loop body containing a large number of manipulations on strings and pointers, as well as integer arithmetic operations[7]. The number of iterations is dependent on the speed of the system being tested, but is normally in the range of 200 to 400 million. Dhrystone calculates the performance of the system as the number of executions of the loop body over the amount of time it took to do so, referred to as *Dhrystones per second*. This number is then divided by 1757 to give the final result in VAX MIPS, with 1757 being the score in Dhrystones per second of a VAX-11/780[1]. The VAX-11/780 was chosen to be a reference computer with a score of exactly 1 VAX MIPS. VAX MIPS are commonly referred to as MIPS, which is also how they are referred to in this report, despite the fact that the number does not represent Million Instructions Per Second.

Version 2.1 of the Dhrystone benchmark was released in 1988, and remains the current and official version today[13]. It is written in C, and the implementation in *condor_mips* contains some additional C++ code supplied by the HTCondor developers which does not interfere with the loop body. *condor_mips* is compiled with gcc 4.4.x[12] on Red Hat Enterprise Linux 6 (RHEL6) using the -O2, -fPIC, and -fstack-protector compilation flags[5]. gcc is capable of altering and optimizing programs during compilation time to increase performance, and -O2 is the second optimization level offered by gcc out of three. The -fPIC flag is used on most, if not all, modern programs, and the -fstack-protector adds additional code to protect against stack overflows. That flag is likely used on all software included in HTCondor to reduce possible attack venues. This is quite well suited for the purposes of this report, as HEP applications are also usually compiled using the -O2 and -fPIC flags, in addition to the -pthread and -m32 flags[6]. -pthread has no purpose in Dhrystone 2.1, as it enables multithreading and Dhrystone 2.1 is a single threaded program, and so the only considerable difference is the -m32 flag, which forces the program to compile as a 32-bit application.

4.2 *condor_kflops*

condor_kflops is an implementation of the Linpack floating-point benchmark. Unlike Dhrystone, Linpack performs operations almost entirely on floating-point numbers[9]. This type of performance measurement is

normally more useful in scientific computing applications, as the majority of calculations are floating-point operations. The Linpack benchmark generates and solves a dense square system of linear equations using LU factorization with partial pivoting[9]. It repeats this process many times, the exact number again dependent on the speed of the system; however, in this study the number of iterations ranged between 4000 and 12000. Linpack takes the amount of time required to perform the calculations to calculate the peak floating point performance of the system in kFLOPS (kilo FLOating Point operations per Second). However, the Linpack benchmark performs a very regular and specific task, one which can occur in practice but is not representative of the majority of scientific work. Linpack also has the goal of providing an upper bound for the system's performance. Therefore, the number returned by Linpack must be interpreted as the peak, not average, performance of a system solving a dense system of linear equations, rather than the general expected floating-point performance[9]. Nevertheless, the results are still useful and can be used to compare the floating-point performance of systems.

A version of the Linpack benchmark called Highly Parallel Linpack (HPL) is used today for generating the results of the TOP500 supercomputer list[9]. The version of Linpack implemented in *condor_kflops* is not HPL; it is the C version written in 1988 and is single threaded[7]. *condor_kflops* also only tests the performance on double-precision floating-point numbers (64 bits), as single-precision floating-point numbers are rarely used today. Like *condor_mips*, the HTCondor developers added some additional C++ code outside of the core computations. It is compiled with gcc 4.4.x[12] on RHEL6 using the -O2, -fPIC, and -fstack-protector compilation flags[5], and is compiled with loop unrolling. Like Dhrystone, this version of Linpack has no use for the -pthread flag, so again the only difference from the HEP standard is 32-bit compilation.

4.3 *lhcbmc.py*

lhcbmc.py is a benchmarking program written in the Python programming language originally produced by the LHCb-Dirac developers team. It was later revised to be a multicore benchmark, making it unlike both Dhrystone and Linpack, in order to emulate the CERN standard HEPSPEC06 benchmark. HEPSPEC06 is an adaptation of the SPEC CPU™ 2006 benchmark, with the purpose of benchmarking a system's ability to perform HEP applications[6]. However, the HEPSPEC06 benchmark can take several hours or even days to run, and as a result was not practical for use in this report.

lhcbmc.py first determines how many cores are available to the system (or logical cores on a hyperthreaded machine), then adjusts the number of threads accordingly using Python's *multiprocessing* module. Then each thread initializes two integer variables and two long variables[1], and performs 12500000 iterations over a loop body which generates a Gaussian random real number. This number is added to the contents of one integer and long variable, then the number is squared and added to the remaining two variables. The program then takes the elapsed CPU time (the sum of the elapsed user time, system time, user time of children, and system time of children as given by *os.times()*) for each thread, and returns the quotient of 360 *HS06 seconds* over the elapsed CPU time to give the result in units of HS06, the units used by the HEPSPEC06 benchmark. This is because the number of iterations done in the benchmark is stated to correspond to 360 HS06 seconds, that is, a system which takes one second to finish the *lhcbmc.py* benchmark should have an HS06 score close to 360. *lhcbmc.py* is stated to give a reasonable approximation of a system's HS06 score under the conditions that each core is running its own thread, and the system is not busy performing other tasks. This is the case in the testing environment used in this report.

4.4 Limitations

Both Dhrystone and Linpack are free, open source, widely known, quick to run, and easily obtainable. That, as well as their inclusion in HTCondor, led them to being chosen for use in this report. Despite measures in this study to make the data obtained from those benchmarks reliable and consistent, an inherent limitation of both benchmarks is that they are very old. Version 2.1 of Dhrystone and the version of Linpack used in *condor_kflops* were both released in 1988. Modern processors and compilers are significantly more advanced than the processors on which Dhrystone and Linpack were designed to run. Consequently, many believe that Dhrystone and Linpack struggle to adequately benchmark a modern processor.

[1] In Python 2.x, defining a "long" was done by appending an *L* to the end of the number being assigned. In Python 3.x, the "long" datatype was renamed "int" and the Python 2.x "int" datatype was removed, along with the *L* syntax.

In addition, the optimizing capability of modern compilers is much greater than the designers of these benchmarks could have anticipated. While Dhrystone 2.1 is more resistant to over optimization than the previous versions[13], and Linpack has sufficient utilization of pseudo-random numbers, both can be potentially over optimised by a modern version of any C compiler. Fortunately neither *condor_mips* nor *condor_kflops* are excessively optimised, but do contain some optimization; although, this is preferable since it is necessary when benchmarking to match the environment of the applications to be run as closely as possible.

5 Collecting the Data

To collect benchmarking data for a cloud, hundreds of VMs were started, benchmarked, and shutdown in succession in order to benchmark as many distinct instances as possible. In addition, each cloud provides several predefined configurations, called VM flavours, which specify the number of cores, ram, and disk space requested for the VM. To determine if any performance differences existed between flavours, each flavour had to be tested separately and have their results categorized. The data collection also had to run as quickly as possible, since there was a fairly large amount of data to be collected, and it needed to be able to run without a user being present.

The code required to accomplish this, and to make it easily portable between clouds, turned out to be quite complicated. HTCondor and Cloud Scheduler are not designed to operate on such short time scales and with such high volumes of VM startup and shutdown. Normally, a batch of VMs running HEP applications are booted and run for days rather than minutes. As a result, it is normally not a concern if a VM takes half and hour to start up or properly shut down; however, the benchmarking in this application runs for at most 7 minutes. Therefore, much of the process needed to be sped up manually.

HTCondor runs both *condor_mips* and *condor_kflops* once upon initializing the VM and makes the result available in the metadata[7]. However, it was observed upon running both benchmarks repeatedly in a VM that the return values could fluctuate considerably on even a single core. To compensate for this, a custom script was used to have the VM execute both benchmarks 15 times each and average the results. Each core is benchmarked individually, and the results are returned per core rather than averaging the results for a VM.

lhcbmc.py is run once per VM rather than per core, since it is a multithreaded benchmark. HS06 benchmarks are supposed to return the sum of the results of each thread to give an indication of the system's ability to run several single threaded applications simultaneously, a common environment for HEP applications[6]. As such, VMs with a higher number of cores score higher. This result was not very useful for the purposes of comparing VM flavours, so the program was modified slightly to return the result from each thread in the VM instead.

Lastly, the same binary was used for both *condor_mips* and *condor_kflops* on each VM that was benchmarked, and the version of Python used to run *lhcbmc.py* was Python 2.6.6 in every case, to ensure consistency. In addition, the image used for the VMs was a custom μCernVM image compiled for use by the UVic HEP group.

6 Results

Five Openstack clouds used by the UVic HEP group were benchmarked for this report: Compute Canada West (Nefos)[3], Compute Canada East (CC-East)[3], Cybera[2][4], Chameleon[2], and UVic's own cloud Mouse.

6.1 *condor_kflops* Benchmark Results

The following plots are histograms displaying the benchmarking results of *condor_kflops* for a given cloud and VM flavour. The results displayed are the benchmarks of individual cores, not whole VMs, and are given in

[2]Cybera provides VMs on their cloud with a swap space sourced on solid state drives in order to increase the performance of VMs when writing to disk. However, the μCernVM image used for these benchmarks does not have support for swap partitions. As a result, custom flavours were provided by Cybera for use by the UVic HEP group which do not take advantage of their solid state swap space. While this may have a performance impact in other applications, it should have minimal effect in the results provided here as the benchmarks used in this report perform minimal disk operations.

mega FLOPS (MFLOPS) instead of kFLOPS due to the size of the results. Each cloud has plots for 8, 4, 2, and 1 core flavours with the exception of Nefos (figure 2) which had insufficient 1 core data, and Chameleon (figure 6) which had no available 8 core nodes during data collection.

All of the histograms below have a fixed range of 0 to 2300 MFLOPS, and each bin is 20 MFLOPS wide with the lower limit inclusive. In addition, the plots have a variable Y-axis in order to properly emphasize the shape of the distributions.

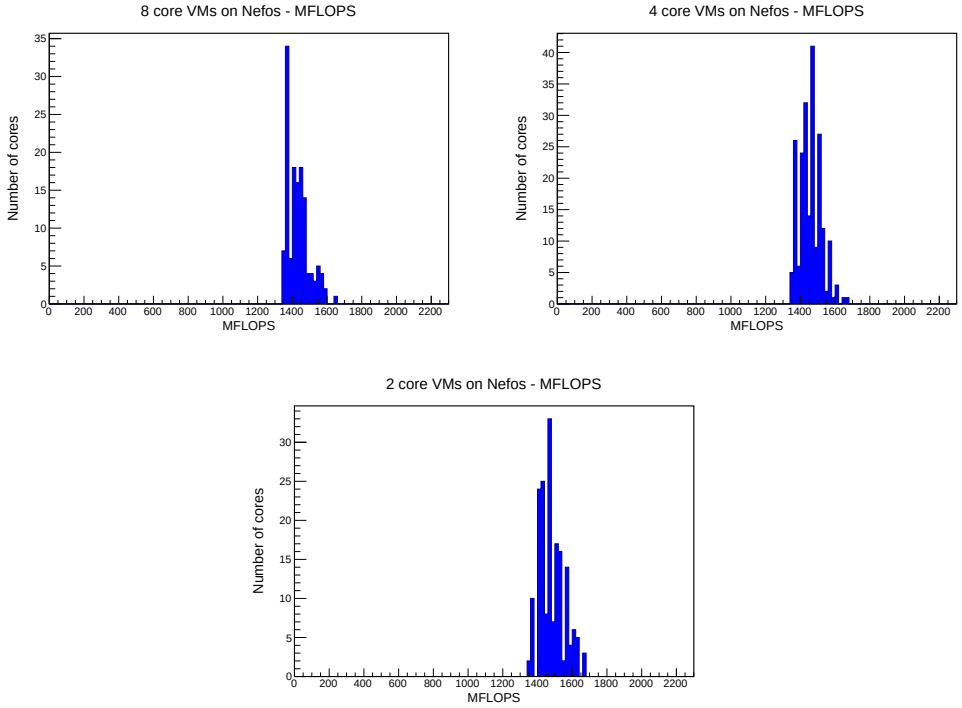


Figure 2: Nefos *condor_kflops* benchmarks; 8, 4, and 2 core flavours.

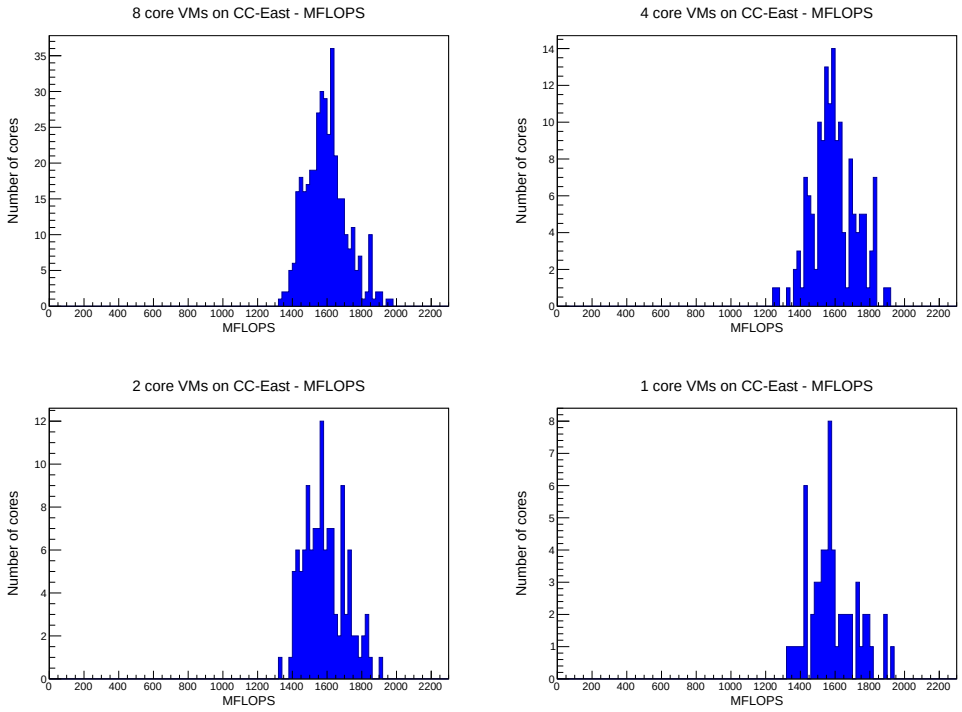


Figure 3: CC-East *condor_kflops* benchmarks; 8, 4, 2, and 1 core flavours.

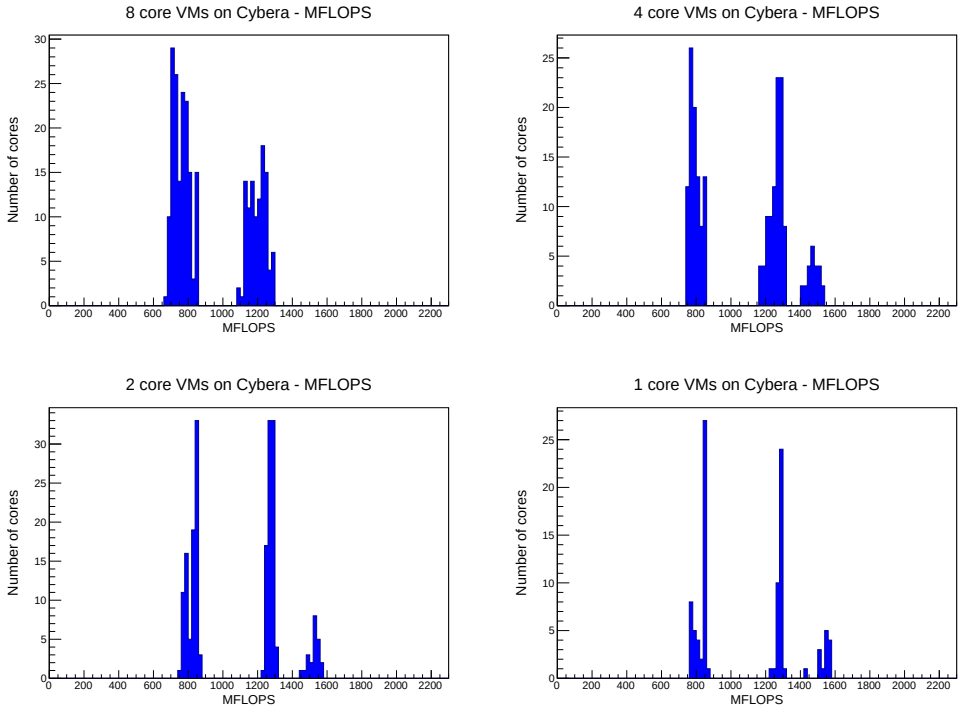


Figure 4: Cybera *condor_kflops* benchmarks; 8, 4, 2, and 1 core flavours.

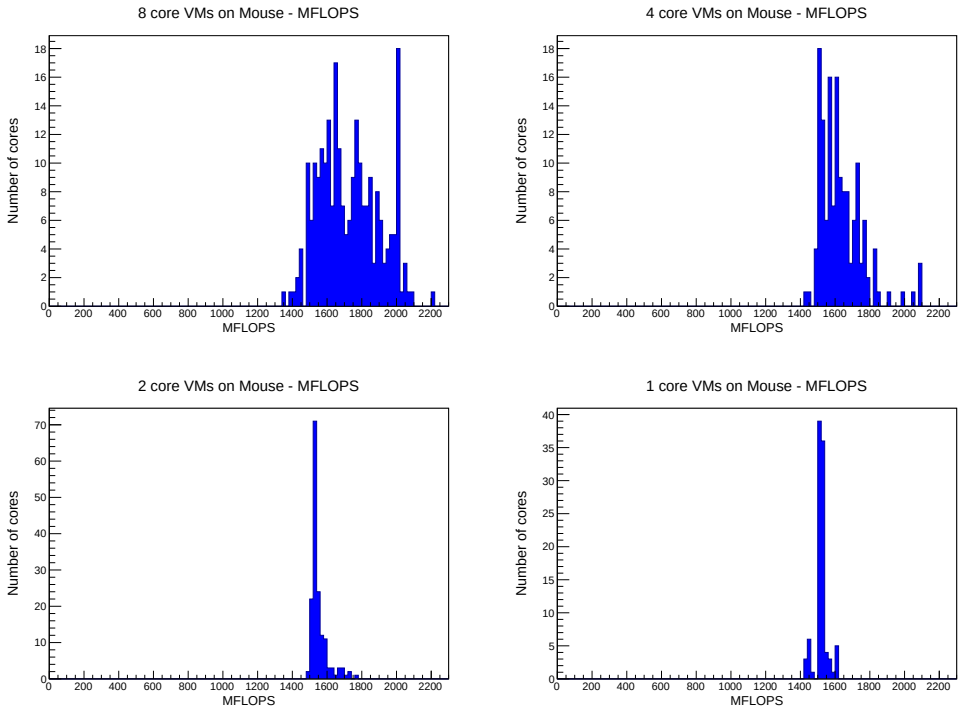


Figure 5: Mouse *condor_kflops* benchmarks; 8, 4, 2, and 1 core flavours.

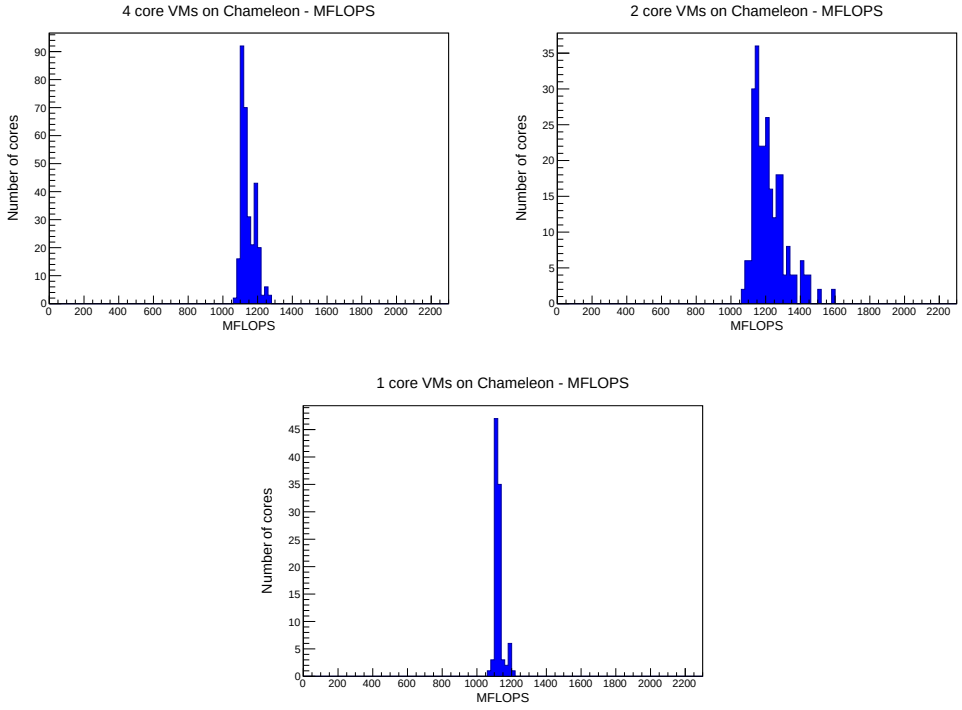


Figure 6: Chameleon *condor_kflops* benchmarks; 4, 2, and 1 core flavours.

The table below contains the median, minimum value, and maximum value of each plot, with the largest value for each column highlighted in green. Each value is obtained from the raw data, rather than specifying bins in the histogram.

Cloud \ Cores	median (MFLOPS)				minimum (MFLOPS)				maximum (MFLOPS)			
	1	2	4	8	1	2	4	8	1	2	4	8
Nefos	—	1470	1459	1422	—	1356	1346	1356	—	1667	1660	1655
CC-east	1567	1569	1584	1586	1331	1337	1255	1334	1927	1906	1907	1969
Cybera	1257	1251	1207	802	763	759	741	677	1577	1564	1523	1294
Mouse	1519	1535	1611	1713	1429	1483	1439	1350	1611	1768	2083	2200
Chameleon	1117	1200	1128	—	1076	1075	1078	—	1201	1592	1264	—

Table 1: Summary of *condor_kflops* benchmarking results.

The results for Nefos show single peaks ranging from approximately 1400 to 1600 MFLOPS, each with a median below 1500 MFLOPS. CC-East, the other Compute Canada cloud, had distributions with a larger variance than Nefos, but also with a higher range from 1400 to 1900 MFLOPS and slightly larger medians sitting above 1500 MFLOPS. Cybera exhibits a clearly defined multimodal distribution for every flavour, with tight peaks roughly centered around 800, 1200, and 1500 MFLOPS. The highest valued peak is small in the 4, 2, and 1 core flavours, and is not present in the 8 core flavour. As a result, the median is not an adequate statistic for summarizing the performance on Cybera, as it has the potential to ignore the significant proportion of results in the 800 MFLOPS peak. Mouse has extremely low variance peaks near 1500 MFLOPS for the 1 and 2 core flavours, with the 8 and 4 cores showing a larger variance. Mouse also has the highest minimum value for the 1, 2, and 4 core flavours, while being very close to Nefos for the 8 core. Lastly, Chameleon shows single peaks each of slightly different variance centered near 1100 to 1200 MFLOPS.

Mouse appears to provide the best performance according to the *condor_kflops* benchmarks, closely matched by CC-East. Nefos provided adequate performance, followed by Chameleon and Cybera which received lower results.

6.2 *condor_mips* Benchmark Results

Below are the histograms containing the results of the benchmarking using *condor_mips*. The method of data collection was identical to *condor_kflops*, therefore the description of the results is similar. The plots below display the benchmarks of individual cores instead of VMs, and the results are presented in VAX MIPS (MIPS). There are plots for 8, 4, 2, and 1 core flavours for each cloud, excluding Nefos (figure 7) and Chameleon (figure 11) for the same reasons outlined in the previous section.

The histograms below all have a fixed range of 0 to 24000 MIPS. Each bin is 200 MIPS wide with the lower limit inclusive. As in the previous section, the plots have a variable Y-axis as to not mask the shape of the distribution.

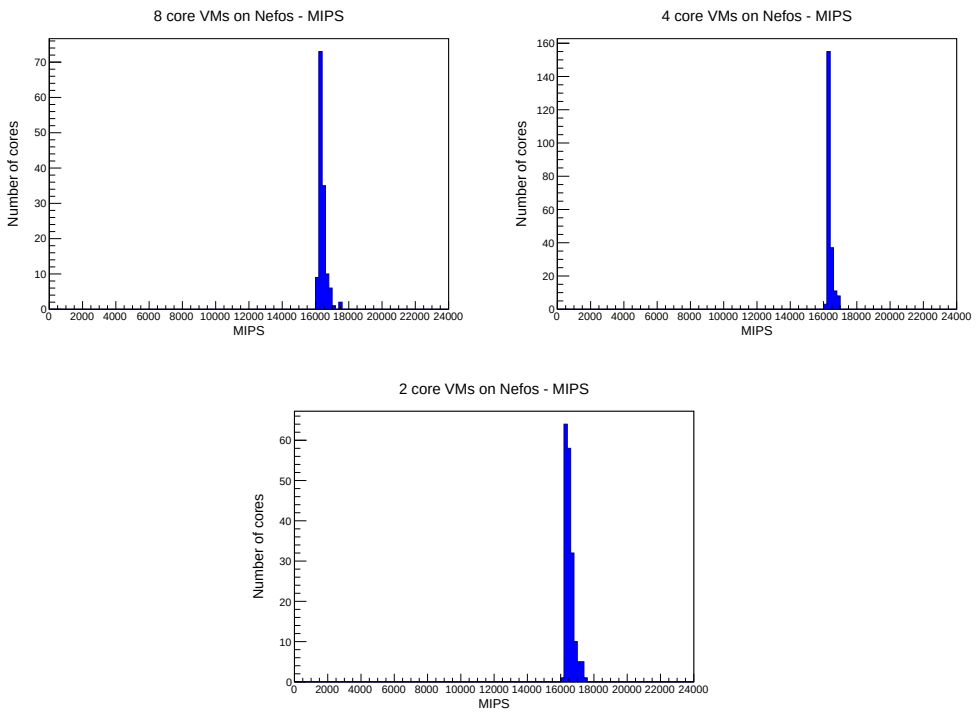


Figure 7: Nefos *condor_mips* benchmarks; 8, 4, and 2 core flavours.

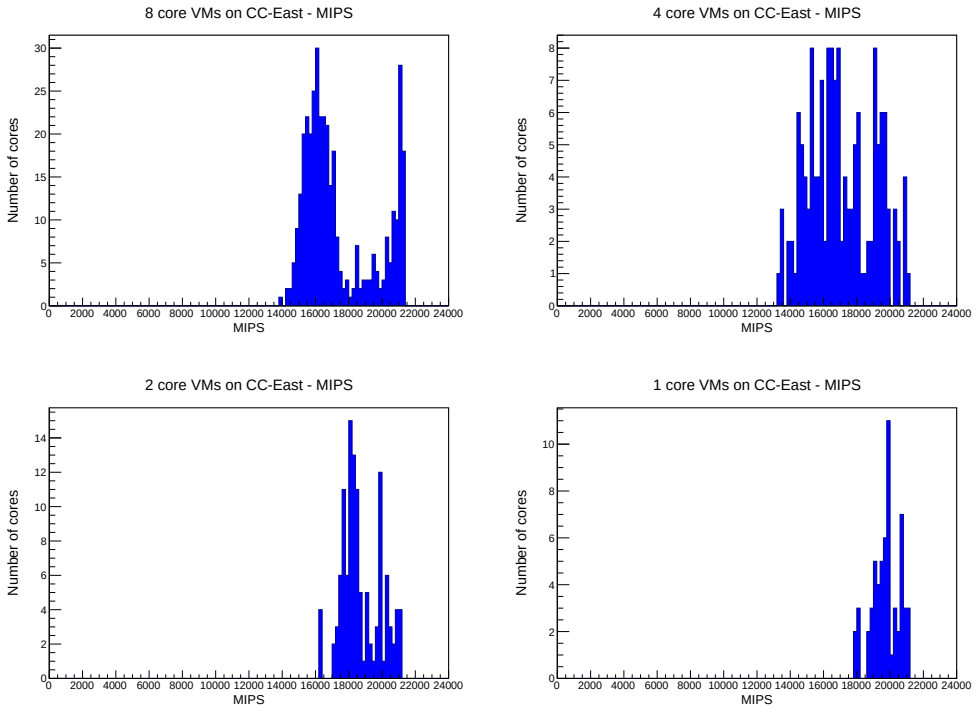


Figure 8: CC-East *condor_mips* benchmarks; 8, 4, 2, and 1 core flavours.

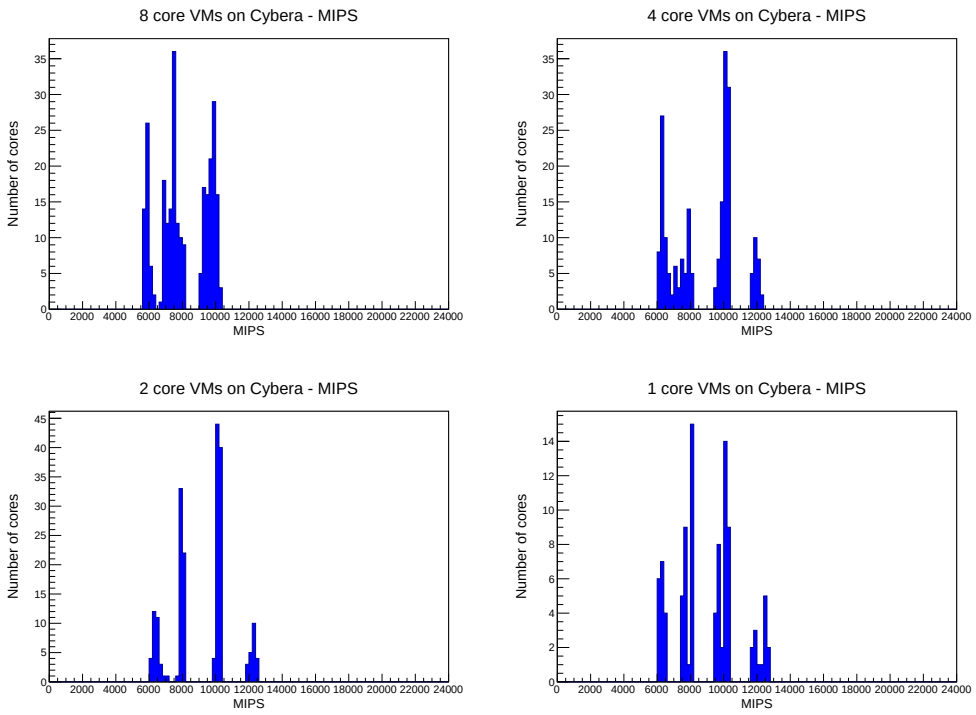


Figure 9: Cybera *condor_mips* benchmarks; 8, 4, 2, and 1 core flavours.

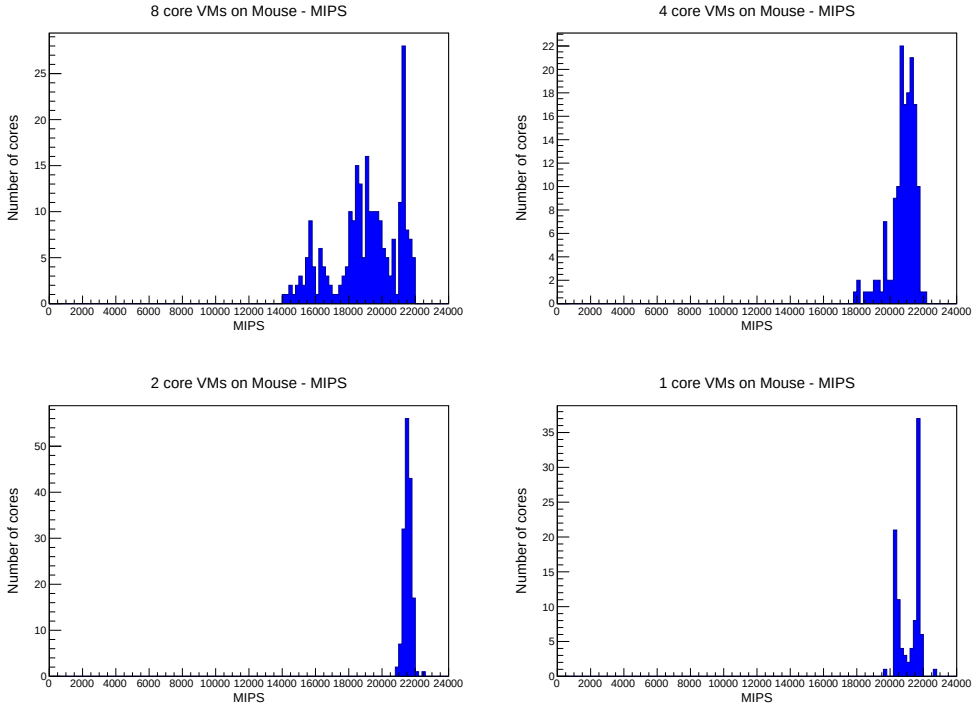


Figure 10: Mouse *condor_mips* benchmarks; 8, 4, 2, and 1 core flavours.

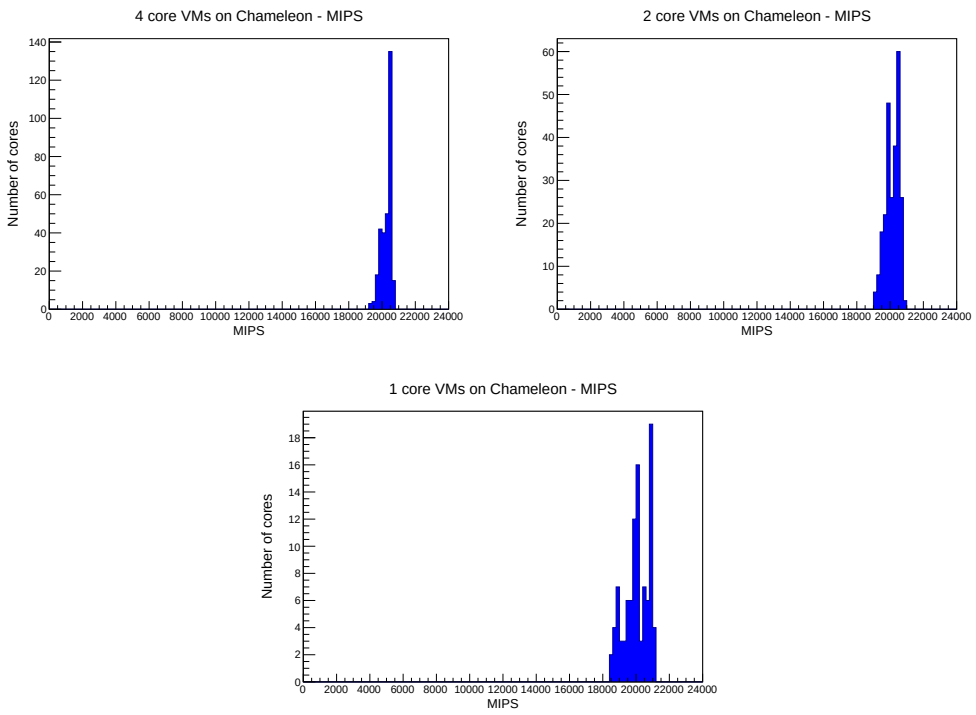


Figure 11: Chameleon *condor_mips* benchmarks; 4, 2, and 1 core flavours.

The table below contains the median, minimum, and maximum values for each plot, obtained from the raw data. The maximum value for each column is highlighted in green.

Cloud \ Cores	median (MIPS)				minimum (MIPS)				maximum (MIPS)			
	1	2	4	8	1	2	4	8	1	2	4	8
Nefos	—	16432	16355	16360	—	16019	16044	16047	—	17519	16888	17485
CC-east	19791	18374	16713	16583	17944	16229	13349	13939	21176	21084	21089	21383
Cybera	9565	10055	9864	7651	6060	6090	6102	5684	12723	12475	12256	10308
Mouse	21493	21520	20912	19186	19751	20885	17807	14136	22780	22515	22118	21886
Chameleon	20043	20199	20393	—	18423	19087	19324	—	21096	20808	20640	—

Table 2: Summary of *condor_mips* benchmarking results.

Nefos displays single low variance peaks, with medians near 16000 MIPS. Similarly to *condor_kflops*, CC-East has much higher variance than Nefos, but has only slightly higher medians with the exception of the 2 core flavour. In contrast to the previous section, the 8 core flavour on CC-East displays a very prominent bimodal distribution centered around 16000 and 21000 MIPS. This is most likely due to CC-East having heterogeneous hardware in their cloud; however, Nefos interestingly does not display a clear bimodal distribution despite also having heterogeneous hardware. Unfortunately, this reduces the usefulness of the median as CC-East will often outperform Nefos in the 8 core flavour despite their similar medians. Cybera again demonstrates a multimodal distribution, with four peaks centered around 6000, 8000, 10000, and 12000 MIPS. As in the *condor_kflops* results, the highest valued peak is not present in the 8 core distribution. However, only three peaks were observed in the results of *condor_kflops* on Cybera rather than the four observed here. It is likely that Cybera has four different types of nodes in their cloud, two of which must have

very similar floating-point performance. Mouse also demonstrates a multimodal distribution in its 8 and 1 core flavours, with all four flavours having a mode near 21000 MIPS. Chameleon has single peaks near 20000 MIPS for each of its flavours, a very large increase in its performance compared to the previous section.

Overall, Mouse and Chameleon seem to demonstrate the best integer performance, according to *condor_mips*. CC-East has the next best performance followed by Nefos, however CC-East lacks consistency. Cybera again displayed lower results compared to the other clouds.

6.3 *lhcbmc.py* Benchmark Results

The plots below contain the results of the *lhcbmc.py* benchmarks. The results are given per VM rather than for individual cores as in the *condor_kflops* and *condor_mips* results, as the goal of the benchmark is to examine multicore performance. However, as stated above, the results for each thread on the VM are averaged rather than summed in order to more effectively compare the differences in performance between VM flavours. The results are presented in HS06 units. Only Mouse has plots for each of the 8, 4, 2, and 1 core flavours. Chameleon still had no available 8 core nodes during the entirety of the data collection, and the *lhcbmc.py* benchmark would not run on a 1 core VM on Cybera for unknown reasons. It will also be noted here that the results for Cybera have approximately half as many data points as Mouse and Chameleon. It was also not possible to obtain data on either Nefos or CC-East with this benchmark.

Each histogram has a fixed range of 0 to 18 HS06, and the bins are 0.25 HS06 wide with the lower limit inclusive. The results of the *lhcbmc.py* benchmark are double precision floating-point numbers rather than integers. As in the previous sections, the plots have a variable Y-axis.

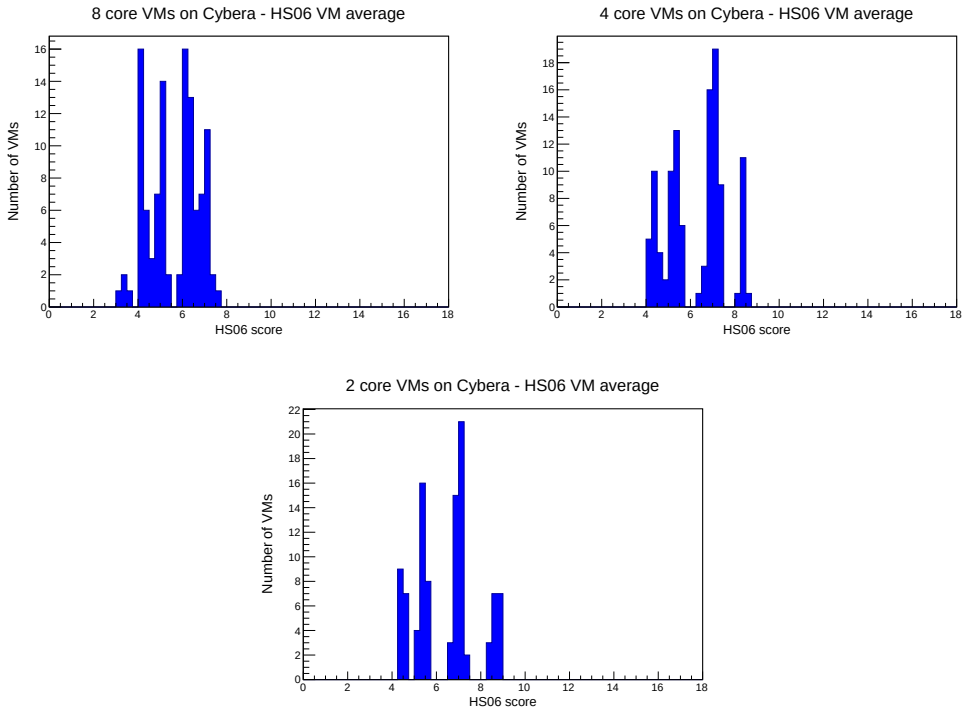


Figure 12: Cybera *lhcbmc.py* benchmarks averaged per VM; 8, 4, and 2 core flavours.

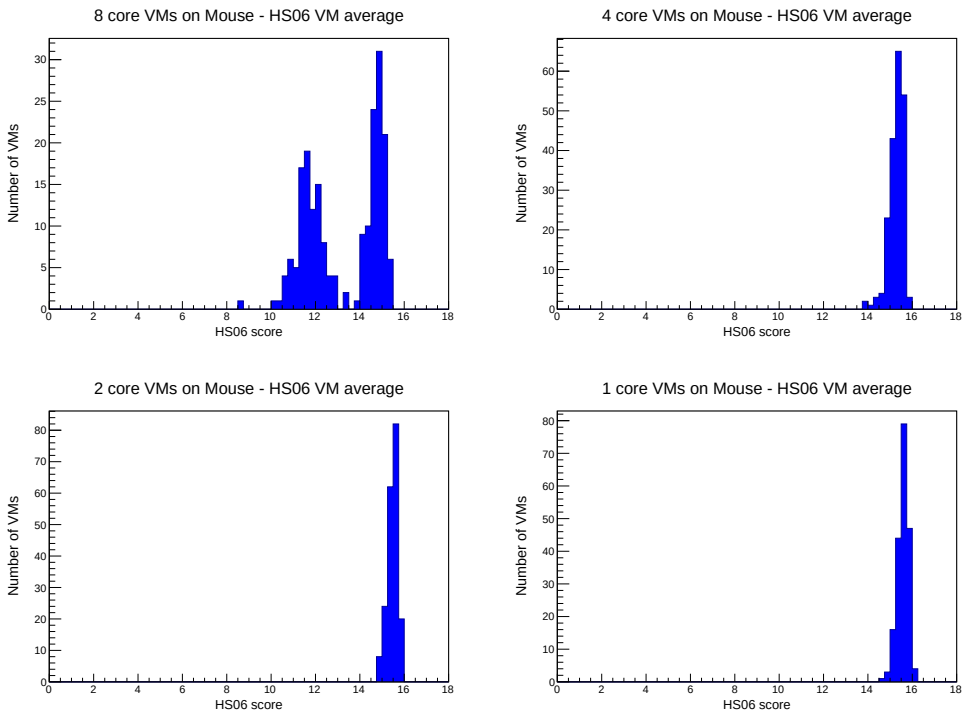


Figure 13: Mouse *lhcbmc.py* benchmarks averaged per VM; 8, 4, 2, and 1 core flavours.

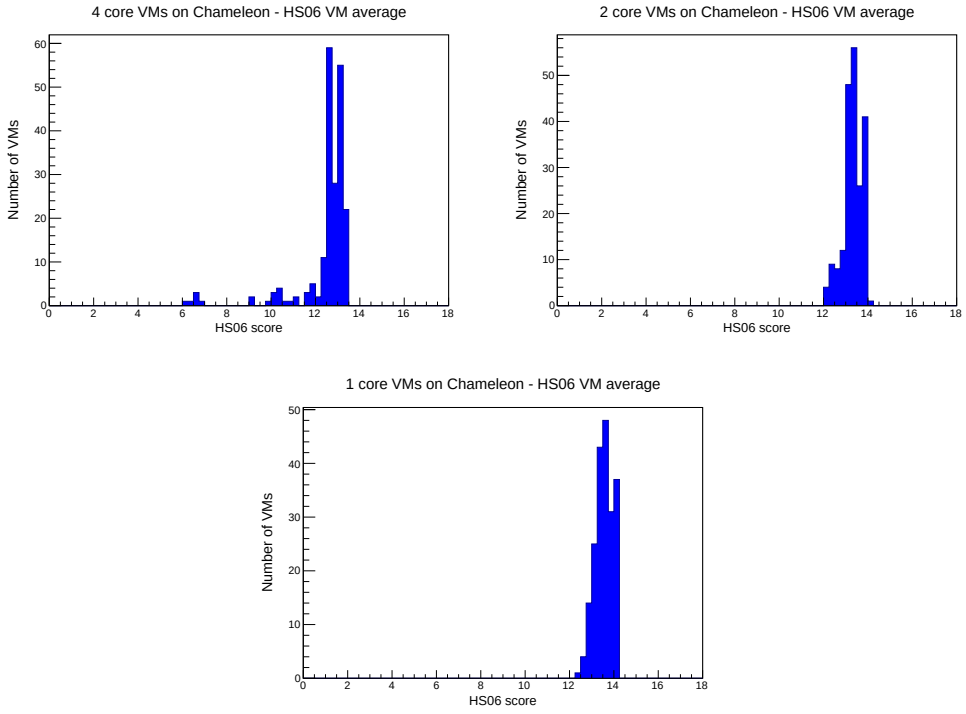


Figure 14: Chameleon *lhcbmc.py* benchmarks averaged per VM; 4, 2, and 1 core flavours.

The table below contains the median, minimum and maximum for each distribution, obtained from the raw data. The values were truncated to two decimal places in order to make the data easily readable. The maximum value for each column is highlighted in green.

Cloud \ Cores	median (HS06)				minimum (HS06)				maximum (HS06)			
	1	2	4	8	1	2	4	8	1	2	4	8
Cybera	—	6.85	6.81	6.04	—	4.33	4.10	3.06	—	8.91	8.67	7.58
Mouse	15.62	15.52	15.38	14.05	14.75	14.79	13.80	8.73	16.22	16.00	15.83	15.45
Chameleon	13.55	13.34	12.77	—	12.29	12.04	6.14	—	14.22	14.01	13.40	—

Table 3: Summary of *lhcbmc.py* benchmarking results.

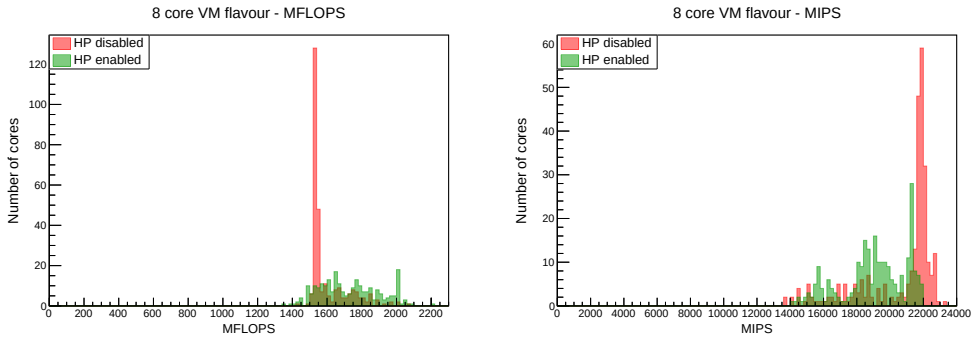
Cybera once again displays a multimodal distribution, with peaks around 4.5, 5.5, 7, and 8.5 HS06. Like the previous two benchmarks, the highest valued peak is not present in the 8 core distribution. However, it is less clear in these results if the two peaks between 4 and 5.5 HS06 are truly distinct peaks or not. In addition, a very small peak near 3 HS06 consisting of only four VMs is present in the 8 core distribution. The distribution on Mouse is similar in shape to the *condor_mips* results, a notable difference being that the 8 core distribution has only two peaks instead of three. The largest valued peak is centered around 15.5 HS06, with the second peak in the 8 core distribution centered near 11.5 HS06. Chameleon displays unimodal distributions centered near 13 HS06. However, the 4 core distribution is slightly different in that it has a few outlying results scattered between 6 and 12 HS06.

Overall, Mouse appears to provide the best performance using the *lhcbmc.py* benchmark, followed by Chameleon, then Cybera.

6.4 Enabling host-passthrough on Mouse

The results of all the benchmarks shown so far for Mouse were taken with host-passthrough enabled. However, enabling host-passthrough on Mouse was only done recently. Since Mouse is an in-house cloud, this provided an opportunity to investigate the difference host-passthrough can have on VM performance. Below are histograms comparing *condor_kflops* and *condor_mips* benchmarks on Mouse before and after host-passthrough was enabled, with the MFLOPS and MIPS results on the left and right, respectively. The only data for the 4 core flavour was taken with host-passthrough enabled, and so it is absent from this comparison.

The only other cloud in this report that is known to surely have host-passthrough enabled is Cybera[8]. The other clouds may also have it enabled, but it is not known with certainty.



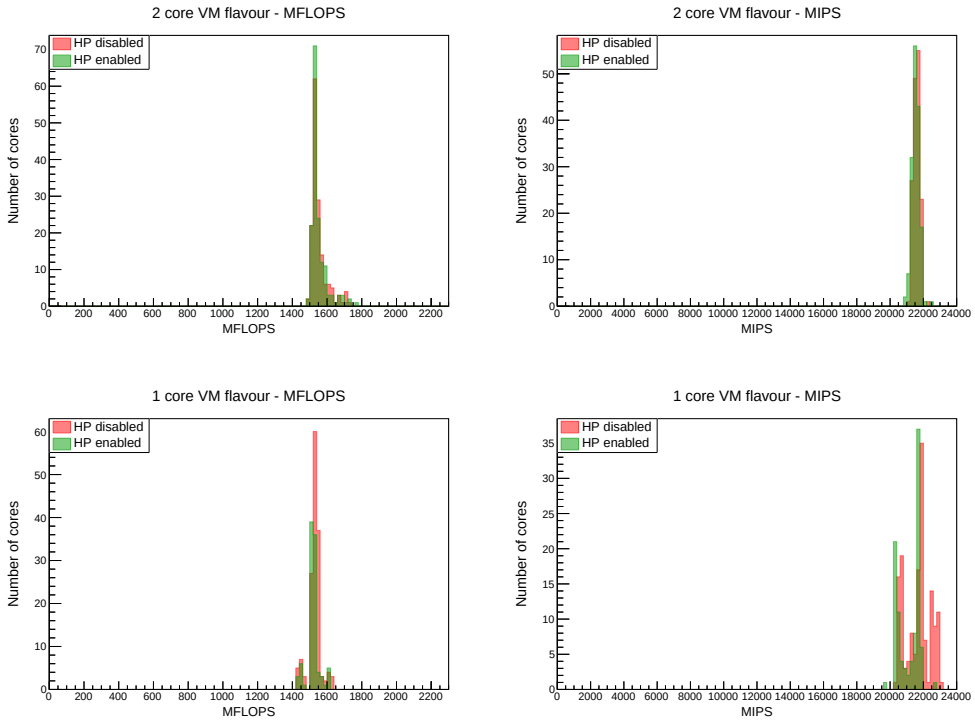


Figure 15: Before and after comparison of enabling host-passthrough on Mouse; 8, 2, and 1 core flavours.

The table below contains the median, minimum and maximum of each of the distributions obtained from the raw data. The higher value between host-passthrough disabled and enabled is highlighted in green.

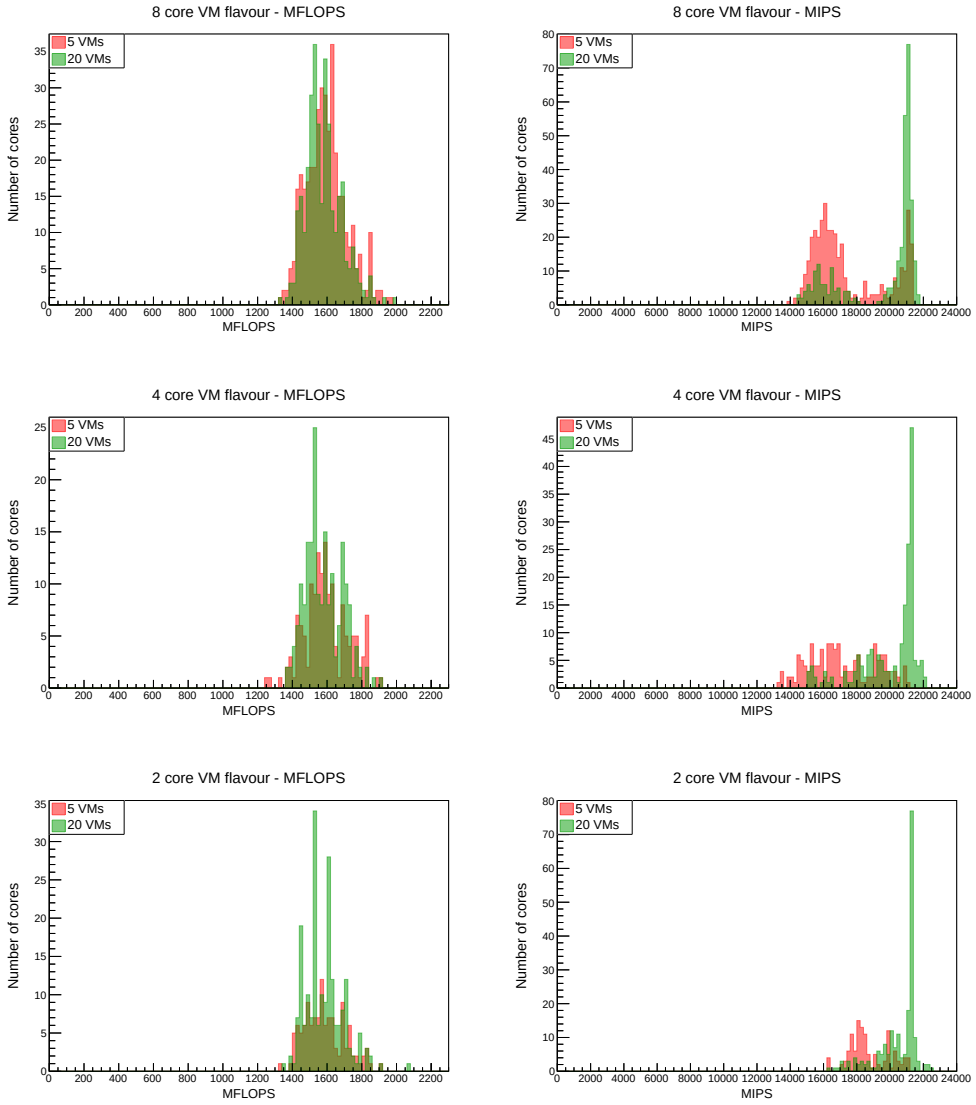
	median (MFLOPS)			minimum (MFLOPS)			maximum (MFLOPS)		
Cores	1	2	8	1	2	8	1	2	8
HP enabled	1519	1535	1713	1429	1483	1350	1611	1768	2200
HP disabled	1527	1536	1540	1432	1497	1428	1623	1733	2068
	median (MIPS)			minimum (MIPS)			maximum (MIPS)		
Cores	1	2	8	1	2	8	1	2	8
HP enabled	21493	21520	19186	19751	20885	14136	22780	22515	21886
HP disabled	21805	21611	21754	20353	21198	13722	23012	22229	23215

Table 4: Summary of benchmarking results comparing host-passthrough performance difference on Mouse.

The 1 core flavour showed a slight decrease in performance with host-passthrough enabled, reflected in the median, minimum and maximum. The performance for the 2 core flavour remained almost exactly the same. The 8 core flavour showed an increase in MFLOPS performance and a decrease in MIPS performance. The 8 core MFLOPS plot may look confusing, but note that the red peak at 15500 MIPS extends unusually high – nearly to 130 cores. This makes the enabled distribution look deceivingly flat. The results are certainly interesting; although, it should be noted that this is only a comparison of host-passthrough performance on a single cloud. Thus it cannot be used to make sweeping conclusions about the effect of host-passthrough in general.

6.5 Benchmarking an Increased Number of VMs Simultaneously on CC-East

In all of the previous results shown in this report, the data was obtained while running only 5 or occasionally 4 VMs at once. However, given that it is not uncommon for nodes to have 24 or even 32 cores in clouds like these, it was a possibility that a benchmarking run on a given cloud could be mostly or completely localized to a single free node in the cloud. In order to test this hypothesis, benchmarks were collected on CC-East while running 20 VMs simultaneously. The histograms below show a comparison of the results from *condor_kflops* and *condor_mips* while running 5 VMs and 20 VMs simultaneously, with MFLOPS and MIPS on the left and right, respectively.



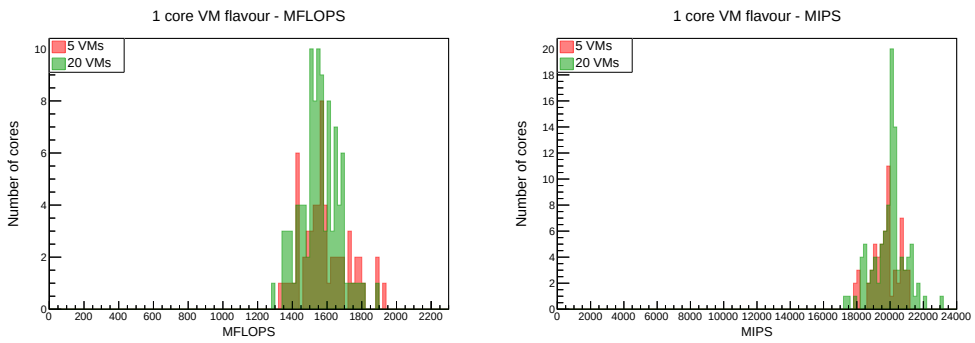


Figure 16: Performance while running 5 VMs vs 20 VMs simultaneously; 8, 4, 2, and 1 core flavours.

The table below contains the median, minimum and maximum of each of the distributions obtained from the raw data. The higher value is highlighted in green.

	median (MFLOPS)				minimum (MFLOPS)				maximum (MFLOPS)			
Cores	1	2	4	8	1	2	4	8	1	2	4	8
5 VMs	1567	1569	1584	1586	1331	1337	1255	1334	1927	1906	1907	1969
20 VMs	1550	1575	1559	1566	1283	1356	1372	1331	1897	2060	1906	1995
	median (MIPS)				minimum (MIPS)				maximum (MIPS)			
Cores	1	2	4	8	1	2	4	8	1	2	4	8
5 VMs	19791	18374	16713	16583	17944	16229	13349	13939	21176	21084	21089	21383
20 VMs	20072	21149	20979	20878	17367	16296	15014	14430	23057	22466	22146	21668

Table 5: Summary of comparing benchmarking results from 5 VMs and 20 VMs at a time.

Little significant change was observed in the MFLOPS results. However, there was a significant increase in the *condor_mips* results for the 8, 4, and 2 core flavours and a more modest increase for the 1 core flavour. Each distribution has a large tight peak near 21000 MIPS except for the 1 core. The second mode observed in the 8 core flavour with *condor_mips* is still present in the 20 VM distribution, but is smaller. While the 20 VM results are interesting, they seem to exhibit a lower variance than the 5 VM results, which is the opposite of the expected result if the 5 VM distributions were taken from only a single node. As a result, these results weaken the plausibility of the earlier hypothesis.

7 Future Work

It would be beneficial to perform a similar benchmarking suite using the HEPSPEC06 benchmark, and compare the relative performance obtained using that benchmark to the results in this report. This would help conclude if the benchmarks used in this report are adequate for doing large-scale data analysis of clouds, or if they are suitable for use as pilot benchmarks. In particular, it would be useful to compare actual HEPSPEC06 results with the *lhcbmc.py* results to determine the margin of error when approximating an HS06 score using

In addition, it would be useful to compare the performance difference of host-passthrough on other clouds, if possible. With additional data it would be easier to make a statement of the performance effect host-passthrough has on a cloud.

8 Conclusion

With the heightened computational requirements soon to come out of Run 2, a proper monitoring system and knowledge of the available resources will be necessary for the HEP community to take full advantage of the scientific data to come out of the LHC. The benchmarks of the clouds examined in this report indicate that there is a measurable difference in the performance of the clouds used by the UVic HEP group. The results obtained here can assist with HEP workload management in the coming future, and can also be used to collaborate with cloud providers to improve their services.

9 Acknowledgements

I would like to thank Ron Desmarais for his assistance and guidance throughout the work term, as well as providing me with several opportunities and making me feel welcome here. I also extend my thanks to Randall Sobie for providing me with this work term opportunity, and to the National Sciences and Engineering Research Council of Canada for providing funding for me during this work term. I would also like to thank Martin Conklin for his helpfulness during my work term, and for sharing his knowledge of previously working for the UVic HEP group with me. Lastly, I would like to thank everyone involved in the UVic HEP group: Colin Leavett-Brown, Micheal Paterson, Kevin Casteels, Ryan Taylor, and Ian Gable.

Glossary

compilation flag An argument that can be passed to a compiler in order to make specific changes to the current program being compiled. This can include having the compiler add certain performance optimizations, or additional security.

hyperthreading A feature on modern high-end CPUs which allows a single core to execute two threads at once, taking advantage of when one thread is sleeping or waiting on an I/O operation to execute the other thread. This attempts to reduce or eliminate the amount of time a core spends idle.

hypervisor A software that either runs in an operating system or directly on the hardware of a computer that manages and provides resources for VMs.

image A file containing all of the necessary components and file structure of a functioning computer system. They typically contain an operating system, boot information, and basic system utilities, but can also contain additional user applications. Also referred to as a boot image.

loop unrolling A code optimization feature, normally done during compilation, which expands for loops into series of statements. This removes the overhead of the loop logic (incrementing and testing the index variable, branching to the beginning of the loop) at the cost of increasing the size of the program.

Openstack An open source cloud computing platform that manages and controls the various components of a cloud. All of the clouds examined in this report are Openstack clouds.

petabyte A thousand terabytes, or 10^{15} bytes.

References

[1] ARM Technical Support Knowledge Articles. *Dhrystone and MIPs performance of ARM processors*. ARM Limited. July 28, 2010. URL: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/4160.html (visited on 08/25/2015).

[2] *Chameleon Cloud*. URL: https://www.chameleoncloud.org/.

[3] *Compute Canada - Calcul Canada*. URL: https://www.computecanada.ca.

[4] *Cybera*. URL: http://www.cybera.ca/.

[5] Thain G. *[htcondor-admin #28581] condor_kflops Linpack version*. Personal e-mail. Aug. 17, 2015.

[6] *How to run the HEP-SPEC06 benchmark*. URL: https://w3.hepix.org/benchmarks/doku.php?id=bench:howto (visited on 09/03/2015).

[7] HTCondor Team. *HTCondor Source Code*. Version 8.2.8. Computer Sciences Department, University of Wisconsin-Madison. URL: http://research.cs.wisc.edu/htcondor/downloads/?state=select_from_mirror_page&version=8.2.8&mirror=UW%20Madison&optional_organization_url=http:// (visited on 08/25/2015).

[8] Topjian J. *Cybera CC-east comparison*. Personal e-mail. July 15, 2015.

[9] Dongarra JJ, Luszczek P, and Petitet A. "The LINPACK Benchmark: past, present and future". In: *Concurrency and Computation: Practice and Experience* 15 (9 July 14, 2003). DOI: 10.1002/cpe.728. URL: http://www.netlib.org/utk/people/JackDongarra/PAPERS/146_2003_the-linpack-benchmark-past-present-and-future.pdf.

[10] CERN press office. *LHC Season 2: facts & figures*. URL: http://press.web.cern.ch/backgrounders/lhc-season-2-facts-figures (visited on 09/04/2015).

[11] Longbottom R. *Dhrystone Benchmark Results - Roy Longbottom's PC benchmark Collection*. URL: http://www.roylongbottom.org.uk/dhrystone%20results.htm (visited on 08/25/2015).

[12] *Red Hat Enterprise Linux 6.0 Release Notes. 13. Compiler and Tools*. URL: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/6.0_Release_Notes/compiler.html#idp7309288 (visited on 09/03/2015).

[13] Weicker RP. "Dhrystone Benchmark: Rationale for Version 2 and Measurement Rules". In: *ACM SIGPLAN Notices* 23 (8 Aug. 1, 1988). DOI: 10.1145/47907.47911. URL: http://dl.acm.org/citation.cfm?id=47911.

[14] The LHCb-Dirac developers team. *The lhcbmc.py Benchmark*. Private Communication. Revised by Wiebalck, A. Aug. 3, 2015.

[15] The ATLAS Organization. *ATLAS Fact Sheet*. URL: http://www.atlas.ch/fact-sheets-view.html (visited on 08/24/2015).

[16] The Openstack Project. *KVM - Openstack Configuration Reference - juno*. URL: http://docs.openstack.org/juno/config-reference/content/kvm.html (visited on 08/21/2015).

[17] *Tier centres | Worldwide LHC Computing Grid*. URL: http://wlcg-public.web.cern.ch/tier-centres (visited on 09/04/2015).