Vrije Universiteit Amsterdam

Bachelor Thesis

SURF HPC Cloud

Author: Connor Daniël Kooistra (2669030)

1st supervisor: Tiziano De Matteis
2nd reader: Alexandru Iosup

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

July 13, 2024

Abstract

High-Performance Computing (HPC) workloads are traditionally executed on supercomputers (on-premise systems). These workloads use techniques such as parallel processing that are crucial to solving many complex scientific computations. However, the high cost of HPC has placed many out of reach of this solution. Cloud computing, on the other hand, allows scalable on-demand access to these resources over the Internet. Fortunately, in recent years, many major cloud providers have started designing and deploying solutions to run HPC workloads in the cloud. This shift towards the cloud is mainly driven by the idea of better scalability, cost efficiency, and application flexibility compared to traditional on-premise solutions. SURF, a Dutch IT services provider that specializes in educational and research institutions, is no exception and has started providing access to supercomputers through its HPC cloud. This represents a viable alternative and attractive opportunity for cost-effective deployment of HPC workloads in the SURF HPC cloud. However, uncertainties remain whether this platform can deliver performance comparable to traditional on-premise systems. Thus, this thesis investigates the cloud platform by evaluating the user experience, compute and network performance, and cost implications by comparing with their on-premise counterpart, the Snellius supercomputer. This study emphasizes findings which show that the HPC cloud favors the experienced user, who prefers single workspace workloads rather than clusters of workspaces that need to communicate with each other. Moreover, the costs for using HPC cloud are also significantly higher than for using Snellius, making it not cost-effective after all.

Contents

1

Introduction

High-Performance Computing (HPC) workloads are traditionally executed on supercomputers (on-premise systems). These workloads use techniques such as parallel processing that are crucial to solving many complex scientific computations (1) (2). However, the high cost of HPC has placed many out of reach of this solution. Cloud computing, on the other hand, allows scalable on-demand access to these resources over the Internet. Fortunately, in recent years, many major cloud providers have started designing and deploying solutions to run HPC workloads in the cloud (3). This shift towards the cloud is mainly driven by the idea of better scalability, cost efficiency, and application flexibility compared to traditional on-premise solutions. This represents a viable alternative and attractive opportunity for cost-effective deployment of HPC workloads on the best-fitting hardware to on-premise solutions. SURF, a Dutch IT services provider that specializes in educational and research institutions, has started providing access to supercomputers through its HPC cloud (4).

1.1 Problem Statement

While the transfer of HPC workloads to cloud environments offers flexibility in hardware deployment, it remains uncertain whether the actual performance of these HPC cloud solutions can match traditional supercomputing systems. Generally speaking, HPC cloud solutions claim to have similar computational resources. However, the architecture of the network and actual performance may vary substantially (3). So, can SURF's HPC cloud solution really provide, as a cost-effective and viable solution, an alternative to on-premise solutions?

1.2 Research Questions

This study aims to investigate the following of the SURF HPC Cloud:

- **RQ1**: What is the User Experience of the SURF HPC Cloud?

The user experience is crucial, as it determines the usability of the platform.

- **RQ2**: How does the compute and network performance compare to an on-premise counterpart?

A fair performance comparison of both platforms provides insight into whether the SURF HPC cloud can provide comparable, superior, or even inferior performance.

- **RQ3**: What are the comparative costs of using a cloud-based computing service compared to an on-premise counterpart?

Cost-effectiveness is a significant consideration for SURF HPC cloud and should thus be investigated thoroughly.

1.3 Research Methodology

- **M1**: To assess the user experience, we will evaluate the ease of use, customization, and documentation of the SURF HPC cloud. Specifically, we want to evaluate two use case scenarios: the deployment of a Virtual Machine (VM), and the deployment of a cluster of VM's.

- **M2**: We will evaluate the compute and network performance of the SURF cloud solution through benchmarks and then compare them with the performance of an on-premise system, specifically the SURF Snellius supercomputer.

- **M3**: To compare costs, we investigate the cost model of the SURF HPC cloud and of the SURF Snellius supercomputer.

1.4 Thesis Contributions

This thesis will provide proof that SURF's HPC cloud solution can be a cost-effective alternative to on-premise HPC deployments by assessing the User Experience and comparing the performance of SURF's HPC cloud solution with their own on-premise supercomputer Snellius.

1.5 Plagiarism Declaration

I hereby confirm that this thesis is my own work and has not been copied from any other source (man or machine).

1.6 Thesis Structure

Chapter 2 gives background information for this study. Chapter 3 describes the design of the SURF HPC cloud and provides insight on its platform. Chapter 4 presents the details of the implementation of the experiment. Chapter 5 represents the results, the results of related work, and its main findings. Chapter 6 concludes the study with answers to the research questions. On top of that, Chapter 6 gives some concluding remarks and makes way for future research.

2

Background

In this chapter, certain concepts are presented that are crucial for understanding this project. In the first section, the concept of High Performance Computing is described. Followed by Virtualization, and then Cloud Computing.

2.1 High Performance Computing

Where serial computing works primarily by dividing the workload into a sequence of tasks to run one after another, HPC uses the concept of massively parallel computing (5). Parallel computing is the use of multiple compute cores to solve some tasks simultaneously (6). Because of this and the extremely high-performance components of these HPC systems, very high-complexity computations can be solved much faster.

A "node" typically refers to a computing unit within a larger computing infrastructure, comprising one or more processing units (CPU or GPU) combined with memory (RAM), storage devices, and network capabilities. A node is constructed with hardware similar to that of a regular computer system.

A "cluster" is a collection of nodes that are interconnected within the HPC system by high-speed networks to work together as a single unified system. These clusters are designed to deliver high performance through the use of the combination of the power of the nodes. Clusters are highly scalable. Scalability refers to the ability to use an increasing amount of hardware resources to reduce the runtime of an application. Additional nodes can be added to the cluster to increase computational power, enabling larger and more complex workloads.

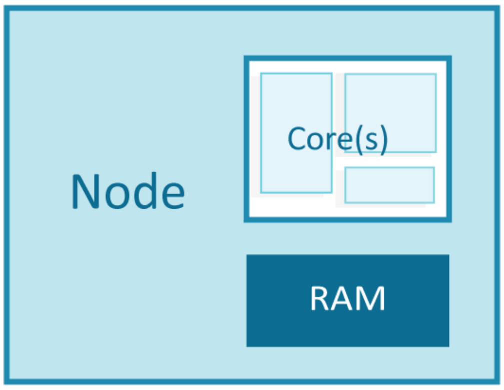


Figure 2.1: Node

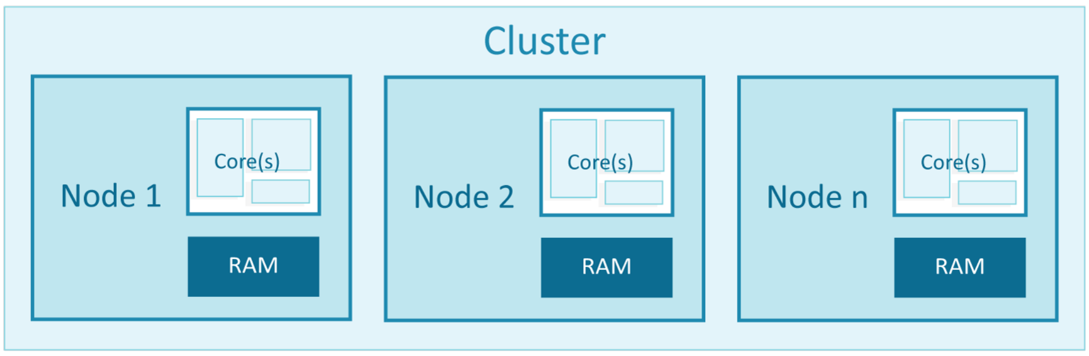


Figure 2.2: Cluster

2.2 Virtualization

Virtualization is the idea of using some software to create an abstraction layer over the system hardware. This software then reallocates the available resources from the system hardware to create multiple virtual machines (VMs), each of which functions as an independent system. This software component is called the hypervisor. The hypervisor manages the abstraction layer and allocates available system resources. Hypervisors provide flexibility and scalability: virtual machines can be quickly created, configured, and deployed to meet computational needs.

2.3 Cloud Computing

Cloud computing is the access to computing services online (7) (8). The core idea of cloud computing is to access resources without having to own or maintain those resources yourself. Some examples might be business or research related, but also consumer services like Amazon Prime. These cloud computing services are offered on a pay-per-use basis, which can optimize costs. The services have very high scalability and are very flexible for all needs. It is the cornerstone for remote work and has now turned to HPC needs.

2.4 HPC Challenge Benchmark Suite

The HPCC (9) is a benchmark that tests the system's performance on CPU, memory, network, and more related sub-components. It consists of 7 standard tests: HPL, DGEMM, STREAM, PTRANS, RandomAccess, FFT, and Communication Bandwidth and Latency. The final test is one we will dismiss because we will have ad hoc network tests explained in the next section. HPCC runs on 3 modes: single, star, and MPI. Single mode (a) means that HPCC runs on a single processor. Star mode (b) means that all processors execute separate HPCC instances without inter-process communication. MPI mode (c) means that all processors execute the HPCC in parallel with inter-process communication. Take a look at Figure 4.1.

2.4.1 PerfKitBenchmarker

PerfKitBenchmarker (PKB) is an open source community effort tool originating from Google that is used to benchmark cloud instances (10) (11). It serves as a wrapper for certain benchmark tools, automating the installation process and providing the ability to

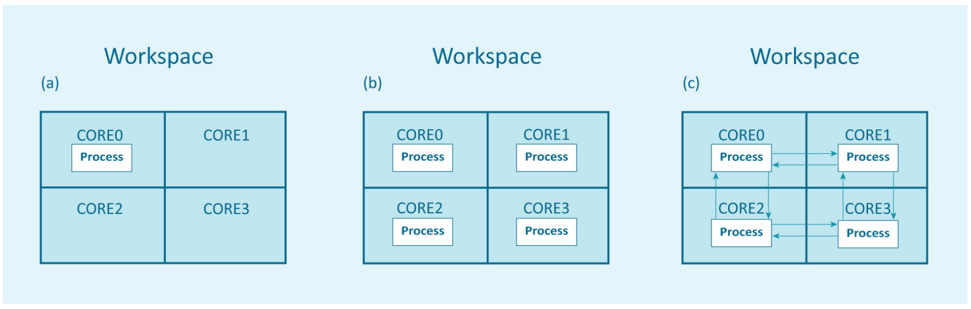


Figure 2.3: HPCC execution modes, where (a) describes the single execution mode, (b) describes the star execution mode, and (c) describes the global/MPI execution mode.

run the workload without over-complicated user interaction. The specific benchmarking tool to run in PKB will be the HPC Challenge Benchmark Suite (HPCC).

2.5 Cloud Noise

To assess the network performance of the cluster setup, we use the benchmark available in the paper Noise in the Clouds (3). This benchmark is built on Netgauge (12) and can be used to depict network performance between nodes using Open MPI. "MPI (Message-Passing Interface) is a message-passing library interface specification" (13), which is used to enable parallel programming in HPC solutions. This benchmark measures bandwidth, latency, and noise. Noise refers to network bandwidth noise or network latency noise. It generally means the fluctuation or variation over time. This is very important because it directly impacts the efficiency and reliability of the measured network rates.

3

Design of SURF HPC Cloud

This chapter covers the design of the SURF HPC cloud, specifically, the design of a workspace and a cluster (of workspaces). In SURF Research Cloud there exist flavors. A flavor is "a predefined set of quantities that describe the dimensions of a workspace" (14). Flavors are the main design concept of the SURF HPC cloud and are used to define workspaces and clusters.

3.1 Workspace

A workspace is a virtual machine on the SURF HPC cloud and can be created using flavors. One flavor is the operating system (OS). For our workspace, we can choose multiple operating systems, such as, but not limited to, Ubuntu 22.04, Ubuntu 20.04, CentOS 7, CentOS 8 Stream, and Windows Server 2019. The possibility exists to not create a workspace based on OS, but rather on specific software application to use, such as, but not limited to, Jupyter Notebook, R Studio, Docker environment, and Samba. This will deploy an OS with the software pre-installed. The options mentioned here are called catalog items. In a catalog of options for deploying a workspace, we can choose certain catalog items that best fit our needs.

After choosing a catalog item, we are prompted to choose the other flavor, the size of a workspace. Not all catalog items offer all sizes, but generally speaking these are the size options:

SURF has recently started to offer workspaces with CUDA workflows. For the sake of cohesion, please see the table below for their available flavors. Keep in mind that investigating these CUDA workspaces will be considered out of the scope of this thesis.

flavor name	CPU cores	RAM[GB]
1 core - 8 GB RAM	1	8
2 core - 16 GB RAM	2	16
4 core - 32 GB RAM	4	32
8 core - 32 GB RAM	8	32
16 core - 64 GB RAM	16	64
16 core - 250 GB RAM	16	250

Table 3.1: HPC Cloud flavors built upon an AMD EPYC 7H12 64-Core 2.6GHz CPU

flavor name	CPU cores	CPU RAM[GB]	GPU(s)	GPU RAM [GB]
RTX 2080 - 1 GPU	5	40	1	11
RTX 2080 - 2 GPU	10	80	2	22
A10 - 1 GPU	11	88	1	24
A10 - 2 GPU	22	176	2	48

Table 3.2: HPC Cloud GPU flavors: A10 CPU built upon an Intel(R) Xeon(R) Gold 6342 CPU @ 2.80 GHz, RTX2080 CPU built upon an Intel(R) Xeon(R) Gold 6126/6226 CPU @ 2.60/2.70 GHz

3.2 Cluster

A cluster consists of multiple interconnected workspaces. At the time of writing, initiating a cluster is considered an experimental feature by SURF and is not working. However, to attain the functionality of a cluster, we can create multiple workspaces and assign them to a private network manually. A private network is created using the same process as a workspace, only more simplified: we choose a flavor, and the network is created. In this case, the flavors are limited and we are only allowed to choose a name for the network and a description. Once a private network has been created, SURF allows us to put workspaces in this network. This can only be done during the workspace creation process, implying that the network must be created before any of the workspaces.

3.3 Cost Model

The cost model is designed based on the concept of "core hours", which is a common metric used in HPC cloud services. These core hours refer to the number of CPU cores used over time. For example, using one CPU core for one hour equals one core hour. Similarly, using four CPU cores for one hour equals four core hours. On top of that, using one CPU core

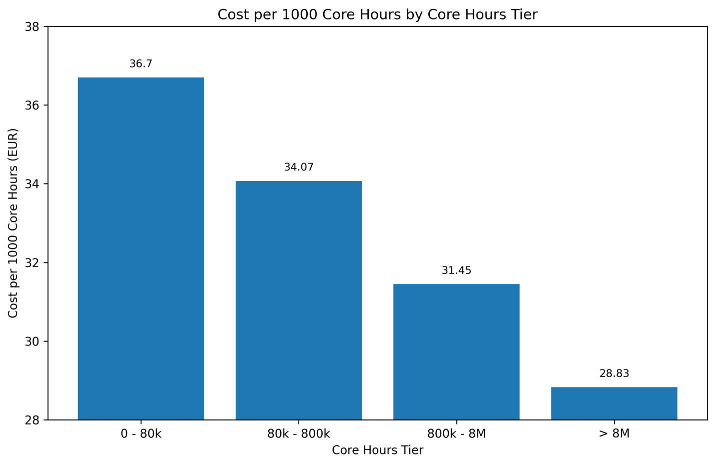


Figure 3.1: Price tiering based on Table 3.3

Number of core hours	Rate per 1,000 core hours
0-80,000	EUR 36.70
80,000-800,000	EUR 34.07
800,000-8,000,000	EUR 31.45
$>$ 8,000,000	EUR 28.83

Table 3.3: Rates of the SURF HPC cloud as of 2023-08 (15)

for four hours also equals four core hours. This model allows for flexibility and scalability. Because the price model is tiered, this approach benefits users with higher workloads, as the cost per hour decreases with increased usage. Consider Table 3.3 and Figure 3.1. Final food for thought: workspaces can be *paused*, meaning whenever a workspace is paused, there are no core hours consumed for that workspace.

3.4 Snellius

Snellius is the Dutch National supercomputer hosted at SURF. Snellius is a cluster of nodes capable of HPC. Snellius operates not through workspaces but through pre-defined computing nodes. These nodes are optimized for tasks and jobs available in the environment. These nodes also vary in size, compute power, and more. See Table 3.4. In order to create a fair comparison, we use a flavor with similar hardware that was made available for the workspaces, namely the tcn (Rome).

nodes	flavor	CPU	CPU cores per node	RAM per node
3	int	AMD EPYC 7F32	16	256 GiB
525	tcn (Rome)	AMD Rome 7H12	128	256 GiB
738	tcn (Genoa)	AMD Genoa 9654	192	384 GiB
72	fcn (Rome)	AMD Rome 7H12	128	1 TiB
48	fcn (Genoa)	AMD Genoa 9654	192	1.5 TiB
2	hcn (4 TiB)	AMD Rome 7H12	128	4 TiB
2	hcn (8 TiB)	TAMD Rome 7H12	128	8 TiB
7	srv	AMD EPYC 7F32	16	256 GiB

Table 3.4: Snellius node flavors

All Snellius nodes use the same network interconnect, which is the Infiniband HDR100 (100Gbps), based on the fat tree topology. For the cloud environment, it is unclear what kind of network interconnect is used, as it is not stated in the documentation, and SURF has never responded to our question.

3.5 Summary

In this section, we have outlined the architecture of the SURF HPC cloud and Snellius supercomputer. We have defined how to create workspaces and clusters and explained how they are made of flavors. The cost model of this cloud service is based on "core hours", a metric that scales, offering tiered pricing to optimize costs based on workload demands.

4

Experimental setup

In the experimental setup, three experiments are defined. The first experiment is the assessment of the user experience. This experiment is merged into the latter two experiments. The first experiment will be running the HPC Challenge benchmark wrapped in the PerfKitBenchmarker(PKB) tool in a single workspace. The second experiment will be to run Cloud Noise benchmarks (12) on a cluster of workspaces.

4.1 System Setup

For the first experiment, in which the PKB tool will run the HPCC benchmark, a single workspace has been created. This workspace uses Ubuntu 22.04 as its OS and will be using the 16 cores - 250 GB RAM flavor. This flavor is based on the AMD EPYC 7702P 64-core CPU. Please, see the specifications in Table 4.1. For the second experiment, in which the cloud noise benchmarks return network-related metrics, a cluster has been created consisting of two workspaces. These workspaces in the cluster will use the same OS and is based on the same CPU, and will be using the same flavor, namely the 16 cores - 250GB RAM. Consider Table 4.1.

Type	OS	Flavor	CPU
Workspace	Ubuntu 22.04 server	16 cores - 250GB RAM	AMD EPYC 7702P
Cluster	Ubuntu 22.04 servers	16 cores - 250GB RAM	AMD EPYC 7702P

Table 4.1: Hardware specifications

4.2 User Experience experiment

The user experience experiment is integrated into the workspace experiment and the cluster experiment. Although difficult to measure quantitatively, the assessment was mostly combined with the setup and running of these experiments. While running the other experiments, insights can be gathered into the accessibility of the platform.

4.3 Workspace experiment

4.3.1 Benchmark configuration

In order to obtain sound results, we have to adjust the HPL of the HPCC benchmark. HPL needs to be tuned in a certain way to perform well (16). There are certain flags that we must determine before we can run the benchmarks.

- Problem size
- Block size
- Process grid ratio

The problem size is based on the memory of the system. To calculate this value, we use a formula widely used for AMD machines (17):

$$Ns = sqrt(M * ((1024 * 1024 * 1024)/8)) \tag{4.1}$$

To fill in this formula, we need to realize M, desired memory usage in GB. A value should be chosen so that the memory usage of the system will be close to 90 percent (18). Considering that we have a total 'usable' memory of 224 GB, we will take 90 percent of that: 202 GB. After we get that, we also reduce 10 percent of potential overhead.

So,

$$Ns = sqrt(202 * ((1024 * 1024 * 1024)/8)) = 148191$$

The block size should be somewhere in the [32..256] interval (17). After some empirical testing, we come to a block size (NBs) of 192.

Finally, to consider the process grid ratio mapping, we take the following rule: P and Q should be somewhat the same; however, Q can be slightly larger than P. We can use $P * Q$ and see if it somewhat matches the number of MPI processes (or ranks): $16/4 = 4$, so we determine 4 Ps and 4 Qs. These are the most important configurations to set for the HPCC benchmark.

4.3.2 HPCC

To be able to run HPCC on SURF's platform, we used a local machine with SSH permissions for the workspace set up for the experiment. On this local machine, PKB had to be installed (in a Python virtual environment). We could then configure HPCC configuration using PKB.

4.3.3 Snellius

To run HPCC in Snellius, we can use jobscripts (19). Snellius allows certain modules to be loaded through these jobscripts. Once the script defines the modules to use, it can then schedule an execution as soon as a node is available.

4.4 Cluster experiment

For the cluster experiment, we had to download and install the above mentioned cloud noise benchmarks after configuring Open MPI and Netgauge. The configuration of this benchmark did not need further adjustments. We do not have to set this experiment up for Snellius; we can use the results of another student.

4.5 Summary

This section has explained the implementation of the experiments that this research will execute. One experiment is the assessment of its compute performance through HPCC, and the other is the assessment of its network performance through Cloud Noise benchmarks. On top of these experiments, the User Experience for these use cases will be measured as quantitatively as possible, and the costs of SURF HPC Cloud and Snellius will be compared.

5

Experimental results

This section will discuss the findings obtained from the experiments and evaluate them.

The main findings are:

- The user experience comes short. The documentation is outdated and contains unintended flavors. Working with this platform requires previous experience of setting up VM's and how to install applications/packages on them.
- The computing power of the SURF HPC cloud is comparable to that of Snellius; however, the network is very underwhelming compared to its on-premise counterpart.
- SURF HPC cloud is much more expensive than its on-premise counterpart due to its difference in accounting.

5.1 Workspace experiment results

In this section, the results of the HPCC benchmark suite will be discussed. For each individual test in HPCC that we have used, there will be a plot showcasing the test result of the SURF HPC Cloud and of the Snellius supercomputer. For each plot, the y-axis will denote the specific test and the x-axis will specify the measured metric. These results have been sent to SURF in order to obtain suggestions on how to optimize these results. However, SURF has not responded, so these preliminary results will serve as the final results.

Figure 5.1 presents the HPL results, and here you can see that the cloud environment exceeds the number of Tflops that Snellius can compute. This measurement based on floating-point operations apparently favors the cloud. Here we can see that the HPC cloud performs better than Snellius, look at Figure 5.2. The DGEMM test, which is the

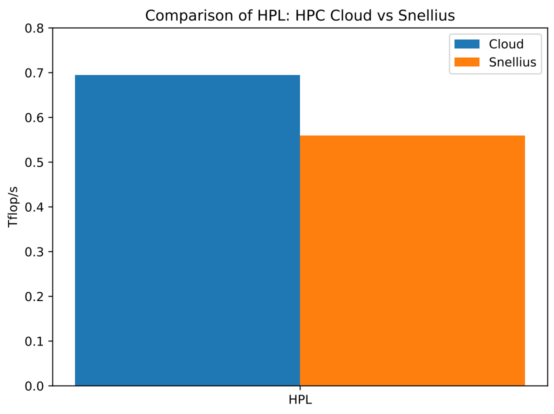


Figure 5.1: HPL results for Cloud (blue) and Snellius (orange). Higher Tflop/s indicates faster calculation capability.

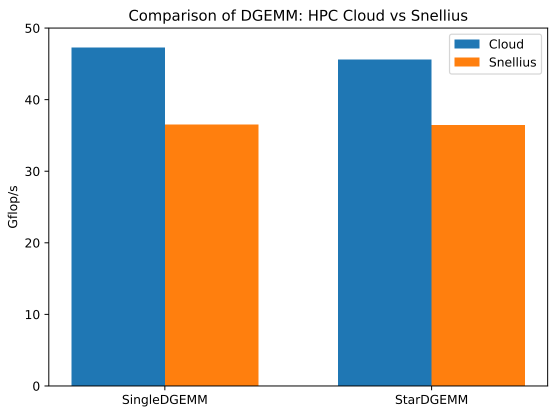


Figure 5.2: DGEMM results for Cloud (blue) and Snellius (orange). Higher Gflop/s indicates faster calculation capability.

measurement of matrix-matrix multiplications, is performing better in the cloud. Figures 5.3 and 5.4 continue this trend in which the cloud outperforms Snellius in tests based on raw computing power. If we, however, start looking at the tests that are more leaning towards RAM power, we can see a clear shift. For Figure 5.5, which measures the rate of 64-bit updates to a random selection of arbitrary elements in a large table, Snellius significantly outperforms the cloud. The STREAM tests are also much more RAM-focused tests. Here, in Figure 5.6, we can see that for a single execution mode, Snellius and the cloud are somewhat on par with each other. But when we look at the star execution mode, Snellius is again on top.

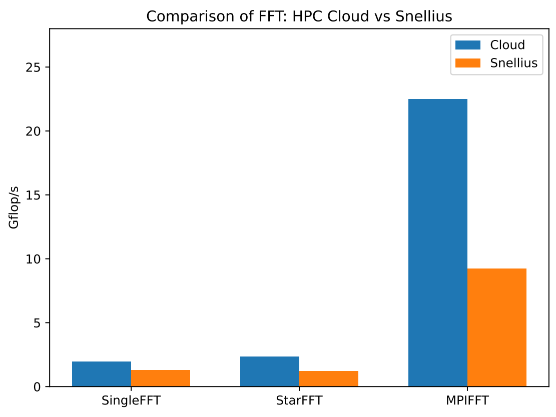


Figure 5.3: FFT results for Cloud (blue) and Snellius (orange). Higher Gflop/s indicates faster calculation capability.

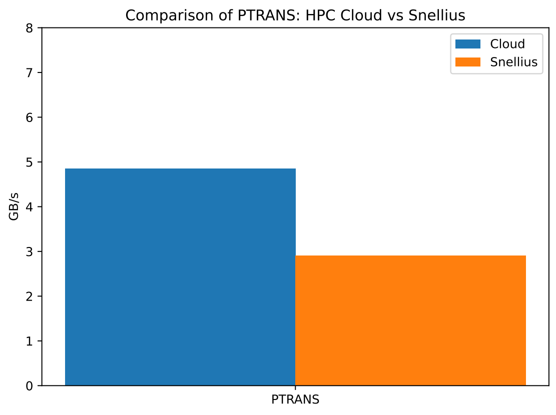


Figure 5.4: PTRANS results for Cloud (blue) and Snellius (orange). Higher GB/s indicates greater data transfer capacity.

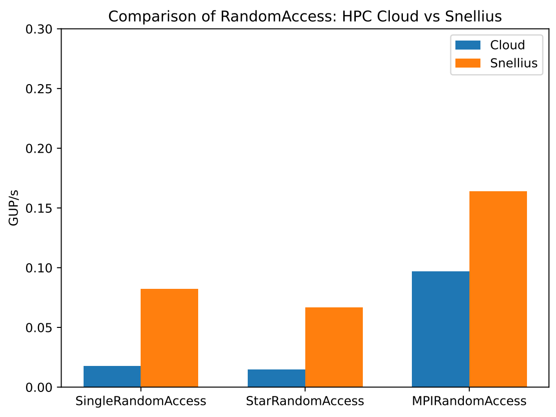


Figure 5.5: RandomAccess results for Cloud (blue) and Snellius (orange). Higher GUP/s indicates faster processing speed for updates.

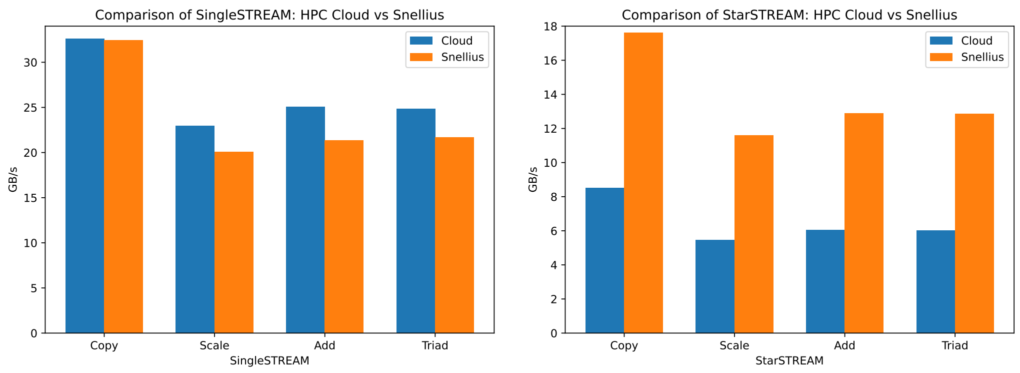


Figure 5.6: STREAM results for Cloud (blue) and Snellius (orange). Higher GB/s indicates greater data transfer capacity.

5.2 Cluster experiment results

In this section we discuss the results of the cluster experiment. We will be looking at four different results: the latency, bandwidth, and their respective noise graphs. This will give a complete overview of the network performance of both systems.

Figure 5.7 shows the measured latency of the cloud and the Snellius system. For these plots, the y-axis shows the measured latency as RTT/2 (us). RTT/2 here means round trip time / 2. On the x-axis you can see that the message size is 1 byte. Looking at the results, there is a huge difference in performance. The cloud environment suffers from very high latency compared to Snellius. The cloud has a latency of around 45, whereas if you look at the zoomed-in frame of Snellius' results, there is approximately 1.3 us latency. Figure 5.8 shows the measured bandwidth of the cloud and the Snellius system. Here, we can see on the y-axis the measured bandwidth as Gb/s, gigabit per second. The x-axis again shows the message size used for the test. Again, we can see a huge difference. For the cloud environment, there is only a measured bandwidth of 15 Gb/s, and the Snellius supercomputer tops around 99 Gb/s. We can clearly see the dominating network performance of the Snellius supercomputer.

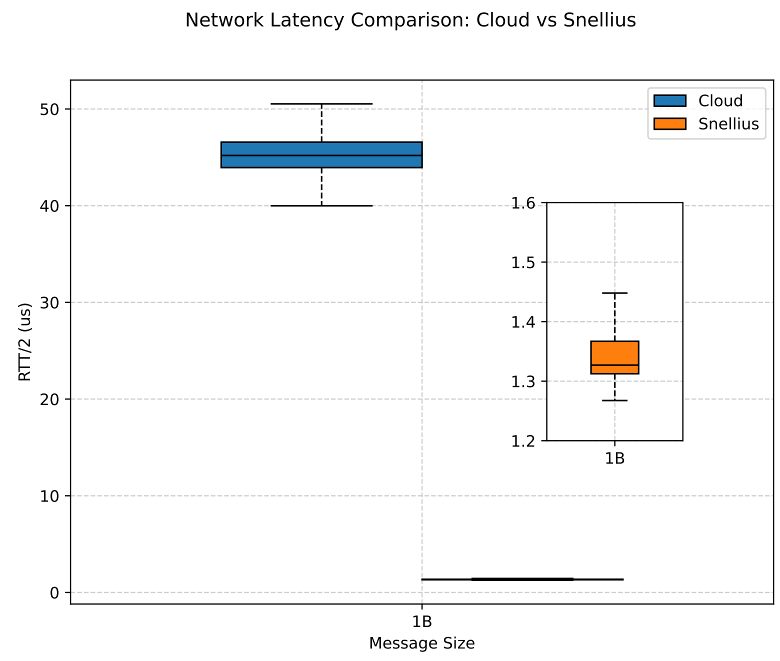


Figure 5.7: Measured Latency (us) for Cloud and Snellius. Less latency indicates quicker response times.

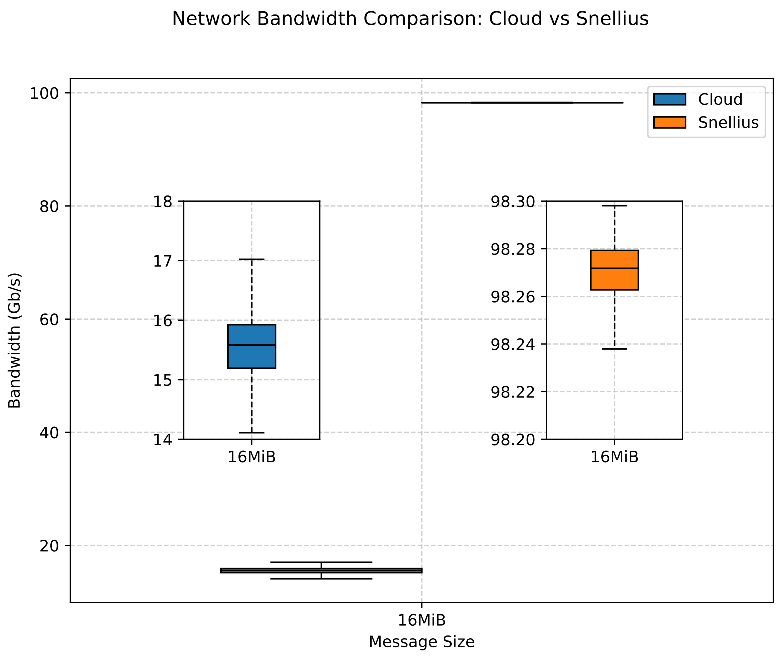
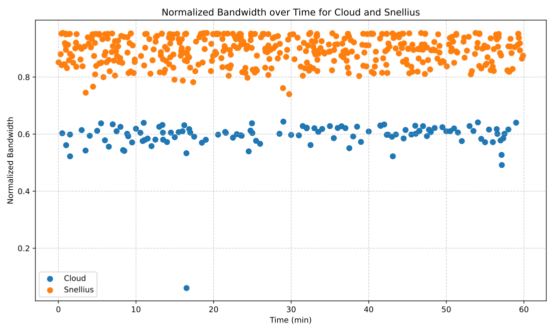


Figure 5.8: Measured Bandwidth (Gb/s) for Cloud and Snellius. Higher bandwidth indicates greater data transfer capability.



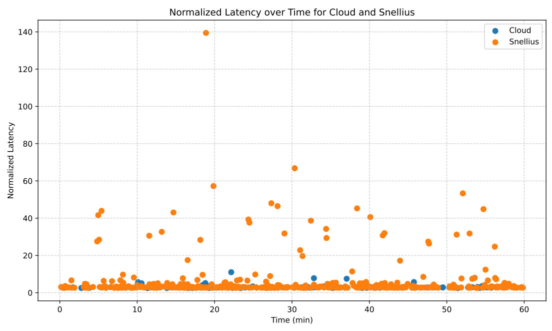


Figure 5.9: Bandwidth (left) and latency (right) noise for both Cloud and Snellius platform.

If we go even further, we can take a look at the noise graphs for each measurement. Figure 5.9 shows the noise for both measurements. We can clearly see that, for the latency, there is barely any noise captured. However, if we look at the Bandwidth noise we can see that a lot of noise accumulates on the cloud environment. This means that the cloud performs only on a 0.6 factor, which means that it can only use 60 percent of its power. For Snellius, this is only a 0.85 factor, which results in an efficiency of 85 percent. These figures show quite clearly how the network performance in Snellius is much superior than in the SURF HPC cloud.

Number of SBU	Rate per 1,000 SBUs
0-10,000,000	EUR 15

Table 5.1: Rates of National Supercomputer Snellius

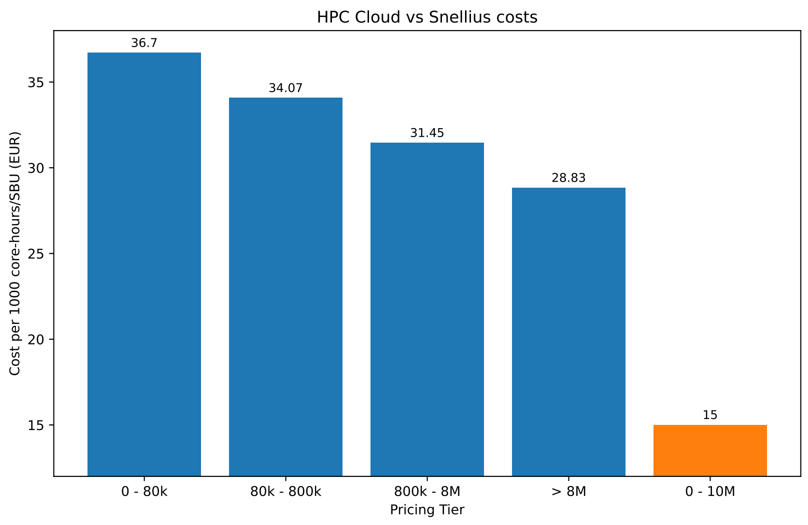


Figure 5.10: Costs for HPC Cloud and Snellius

5.3 Costs

We have seen the costs of the HPC cloud before in Table 3.3. Now we take a look at the costs for Snellius. Look at Table 5.1. Snellius measures the price slightly differently. Instead of core-hours, they measure in System Billing Units (SBU's). Although they are named differently, they are basically the same. An SBU is basically a core-hour, so we can compare Snellius rates directly with the SURF HPC cloud. Take a look at Figure 5.15. In this figure, we can clearly see the difference in pricing. Snellius is less expensive than the SURF HPC cloud. Not only in the actual cost price, but also in its accounting. For Snellius, you pay for the runtime of your issued job. For HPC cloud, you pay for the uptime of your workspace. Should a script finish on your workspace but it remains active, billing will continue. Of course, there is the option to pause a workspace, but for the scenario where a workspace has to remain active, these costs will skyrocket.

5.4 User experience

Alongside the mentioned experiments in the previous chapter, the User Experience has been measured. This experiment is much less quantifiable; however, it has been measured while setting up the workspaces and experiments. For the two use cases, which are the setting up of a workspace and of a cluster of workspaces, the latter is immediately not possible. Apparently, the formation of a cluster is an experimental feature for SURF HPC cloud, and at the time was not working. Performing this manually took significantly more effort than setting up a single workspace. To set up a workspace is relatively straightforward: choose OS + package flavor and the core-to-ram composition. The setup of the experiments was not without difficulties. Many of which were reported to SURF support and unfortunately not resolved.

5.5 Summary

This chapter discusses the main findings of the experiments. Then each section discusses these findings in more detail. For performance experiments, there is a notable difference in network performance. The compute performance is somewhat similar, mainly favoring the SURF HPC cloud instance, but the network performance is significantly worse. The costs are also discussed, showing how Snellius is actually far cheaper than the cloud environment. And finally, the user experience is discussed.

6

Conclusion

This thesis investigated the feasibility of the SURF HPC cloud platform as a scalable, cost-effective alternative to its on-premise counterparts. Focusing on the user experience, performance, and costs, the study aims to provide insights in the possibilities and limitations of cloud environment.

6.1 Answering Research Questions

This study aimed to answer the following research questions:

- **RQ1**: What is the User Experience of the SURF HPC Cloud?

The user experience is difficult to measure quantitatively. However, it is clear from this research that the SURF HPC cloud requires more effort to be fully understood than expected. This study specifically looked at two use cases that were objectively more difficult than stated in the documentation of SURF. The goal of SURF is to provide an easy-to-use alternative to on-premise supercomputers, this has not been met.

- **RQ2**: How does the compute and network performance compare to an on-premise counterpart?

After various benchmarks, this study shows that the compute performance of the SURF HPC cloud performs really well. However, its network performance is very poor compared to its on-premise counterpart. The network used for workspaces to communicate to each other is much slower and suffers from more latency than its on-premise counterpart. In addition, the network present is subsceptible to bandwidth noise.

- **RQ3**: What are the comparative costs of using a cloud-based computing service compared to an on-premise counterpart?

When comparing the costs of the SURF HPC Cloud and the Snellius supercomputer, this study concludes that the cost of the SURF HPC cloud, which is accounted for through the earlier explained notion of uptime, is much higher than its on-premise counterpart.

6.2 Concluding Remarks

Taking into account the achievements and failures of the SURF HPC Cloud, there are definitely benefits to use over an on-premise counterpart. On the SURF HPC cloud platform, there is much more freedom to deploy specific HPC workloads. For Snellius, one is limited to the available modules on the system, whereas on SURF HPC Cloud the user is free to choose its own application (given that it is supported by the OS used on the workspace). However, because of this freedom, it is not user-friendly. The target audience of the SURF HPC Cloud is users who require an easy-to-set-up workspace to conduct scientific research. These users should not be busy with managing and researching how to install a certain package or application that they need. In fact, it should be as plug and play as possible. On top of that, these users could not be knowledgeable in setting up VM's, making it even more time demanding. Therefore, this study shows that the SURF HPC cloud is not a go-to pick for everyone. However, instead shows that SURF HPC cloud can be a viable alternative for the experienced user who desires a single workspace setup capable of HPC computations.

6.3 Future Work

For the future work of this thesis, if possible, these results should be reviewed by SURF themselves. They should be able to confirm the results or explain how some optimizations will clarify the results even more. For the SURF HPC cloud, there is more to it than just workspaces based on CPU and RAM power. The platform also supports the capabilities of implementing GPU's in workspaces, allowing for more use-cases, such as AI computations or other computations that require a GPU. There also exists the possibility to extend the research into storage capabilities for these workspaces (and perhaps clusters should they be a working feature).

References

[1] P. Ved M. Chintalapti J. Pal S.N. (2021). Sukeshini, Sharma. **Big Data Analytics and Machine Learning Technologies for HPC Applications**. 2020. 1

[2] Mattson et al. **The Landscape and Challenges of HPC Research and LLMs**. 2024. 1

[3] Konstantin Taranov Salvatore Di Girolamo Tobias Rahn Daniele De Sensi, Tiziano De Matteis and Torsten Hoefler. **Noise in the Clouds: Influence of Network Performance Variability on Application Scalability**. 2022. 1, 8

[4] SURF. **SURF HPC cloud**. 1

[5] IBM. **What is high-performance computing (HPC)?** 5

[6] HPC at LLNL. **Introduction Parallel Computing**. 5

[7] Microsoft. **What is Cloud Computing?** 7

[8] IBM. **Cloud Computing**. 7

[9] Pietr Luszczek Jack Dongarra. **Overview of the HPC Challenge Benchmark Suite**. 2006. 7

[10] Google. **New Open Source Tools for Measuring Cloud Performance**. 7

[11] Google. **Performance Kit Benchmarker, PKB**. 7

[12] Andrew Lumsdaine Wolfgang Rehm Torsten Hoefler, Torsten MEhlan. *Netgauge: A Network Performance Measurement Framework*. Springer, Berlin, Heidelberg, 2007. 8, 13

[13] MPI Forum. **MPI: A Message-Passing Interface Standard**. 2023. 8

[14] SURF. **SRC Available Flavors**. 9

[15] SURF. **Services and rates**. 2023. 11

[16] J. Dongarra A. Cleary A. Petitet, R.C. Whaley. **HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers**. 2018. 14

[17] AMD. **NETLIB-HPL**. ? 14

[18] AMD. **AMD Zen HPL**$_AVX2$. ? 14

[19] SURF. **Writing a job script**. 2024. 15

Appendix A

Reproducibility

A.1 Abstract

This appendix will cover the instructions on how to set up the program used to run the benchmarks that have been discussed in this thesis. This section will describe the necessary dependencies and how to install the software.

A.2 Artifact check-list (meta-information)

- **Program:** PerfKitBenchmarker.
- **Run-time environment:** SURF HPC cloud platform (virtual environments), SURF Snellius supercomputer (computing nodes)
- **Output:** json, plain text
- **Experiments:** HPCC, Cloud Noise, User Experience
- **Code licenses (if publicly available)?:** Apache v2, Original BSD license, BSD 3-clause

A.3 Installation and Methodology

A.3.1 HPCC

To run the HPCC benchmark through PKB or Snellius, we have described the steps in this subsection.

A.3.1.1 PerfKitBenchmarker

Set up a ssh config file to start making connection to the SURF workspace.

```
Host surf.singleworkspace
    Hostname wanip.singleworkspace
    User username
    Port 22
    IdentityFile /path/to/key
```

Using the command "ssh surf.singlenode", the access to the workspace has been verified. Then test whether Python is installed and create the virtual environment.

```
$ python --version
$ python -m venv $HOME/path/to/venv
```

Now, to install PKB on the local machine (not the SURF workspace):

```
$ cd $HOME/path/to/venv
$ git clone https://github.com/GoogleCloudPlatform/PerfKitBenchmarker.git
$ pip3 install -r $HOME/PerfKitBenchmarker/requirements.txt
```

Set up two config files: single-vm-config.yaml and single-vm.json, and properly address the HPCC configuration in these files.

```
static_vms:
- ip_address: 'wanip.singleworkspace'
    user_name: 'username'
    ssh_private_key: '/path/to/key'
    ssh_port: 22
    zone: 'single-vm'
    os_type: 'ubuntu'
    flags:
      hpcc_block_size: 192
      hpcc_problem_size: 148191
      hpcc_dimensions: "4,4"
      hpcc_benchmarks: "HPL, MPI RandomAccess, MPI RandomAccess LCG, MPIFFT, PTRANS, Sing
```

```
[
    {
    "ip_address": "wanip.singleworkspace",
     "user_name": "username",
     "keyfile_path": "/path/to/key",
     "internal_ip": "lanip.singleworkspace"
    }
]
```

Finally, to run:

```
./PerfKitBenchmarker/pkb.py
--benchmarks=hpcc
--benchmark_config_file=node.yaml
--static_vm_file=node.json
--accept_licenses
```

A.3.1.2 Snellius

For Snellius, we do not have to install anything, instead we are able to load modules and execute the HPCC benchmark set through setting up a jobscript.sh.

```
    #!/bin/bash
#SBATCH -J hpcc-snellius
#SBATCH --nodes=1                   # number of nodes needed
#SBATCH --exclusive
#SBATCH -p rome                     # ensure rome compute node
#SBATCH -t 5:00:00
#SBATCH --mail-type=BEGIN,END
#SBATCH --mail-user=mail@domain.any

#Loading modules
module load 2022
module load foss/2022a
module load HPCC/1.5.0-foss-2022a

timestamp=$(date +"%y%m%d_%H%M%S")
output_dir="$TMPDIR"/output_dir_$timestamp

mkdir -p $output_dir

cp _hpccinf.txt "$output_dir/hpccinf.txt"

cd $output_dir

mpirun -np 16 hpcc

cp -r $output_dir $HOME
```

```
rm -rf $output_dir
```

A.3.2 Cloud Noise benchmarks

To run the Cloud Noise benchmarks on the cluster of workspaces in the SURF platform, we will install openmpi first on both workspaces (workspace01 and workspace02):

```
    sudo apt-get install libpmix2 libpmix-dev
download openmpi
tar -xzvf openmpi
./configure --enable-mpi1-compatibility
make all
sudo make install
mpirun --version -> v5.0.3
```

hostfile: workspace01 slots=16 workspace02 slots=16

after above is confirmed working, move on to netgauge:

```
    tar -xzvf netgauge
./configure MPICC=mpicc MPICXX=mpicxx
make
```

Make sure to check the netgauge installation with mpirun ./netgauge

Move on to downloading the git (cloud noise benchmarks) and install accordingly

sudo apt-get install intel-mkl

set benchmarks/gemm/Makefile naar gcc ipv icc

adjust conf.sh SYSTEM to surf and have following config:

```
MPIRUN="mpirun"
MPIRUN_MAP_BY_NODE_FLAG="--map-by node"
MPIRUN_HOSTFILE_FLAG="--hostfile conf/surf_hostfile"
MPIRUN_ADDITIONAL_FLAGS=""
INTERFACE_MASK=""
RUN_IB=true
RUN_IBV=false
NET_NOISE_CONC=1
BW_SATURATING_SIZE=16777216
MPI_COMPILER=mpicc
NG_CONFIGURE_FLAGS=""
```

then

```
./compile.sh
popd
./run.sh -k "AWS,Normal,Same Rack,Day"
```

A.4 Evaluation and expected results

Python has been used to evaluate the results. These Python files are available in the GitHub repository linked to this thesis.