**Department of Electrical and Computer Engineering**

# Hybrid High Performance Computing (HPC) + Cloud for Scientific Computing

**Suresh Reuben A/L Balakrishnan**

**0000-0002-0539-1499**

**This Thesis is presented for the Degree of**

**Master of Philosophy (Electrical and Computer Engineering)**

**of**

**Curtin University**

**June 2022**

# Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made. This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Redacted Signature

Signature: ………………………………………….

Date: 5th June 2022

## Acknowledgement

I would like to express my sincere appreciation and gratitude to my Main Supervisor, Associate Professor Dr. Ling Huo Chong, and Co-Supervisor, Professor Amandeep S. Sidhu who have given me their enormous support, advice, assistance, and wisdom to help me in this research endeavour. Their inspiration in helping me start the research and then finally helping me cross the finish line has been tremendous. My sincere thanks to the following co-authors who have contributed to the research thesis through the papers that we have co-authored together: Professor Amandeep S. Sidhu, Shanmugam Veeramani, John Alan Leong, Amalina Mohamad Sabri, Sun Veer Moolye, Chung Yik Cho, Professor Sarinder K. Dhillon, and Professor lain Murray.

## Related Thesis Publications

1. S. R. Balakrishnan, S. Veeramani, J. A. Leong, I. Murray, and A. S. Sidhu, “High Performance Computing on the Cloud via HPC+Cloud software framework,” Proc. 5th Int. Conf. Eco-Friendly Comput. Commun. Syst. ICECCS 2016, pp. 48–52, 2017, doi: 10.1109/Eco-friendly.2016.7893240.
2. A. M. Sabri, S. R. Balakrishnan, S. V. Moolye, C. Y. Cho, S. K. Dhillon, and A. S. Sidhu, “Benchmarking large scale cloud computing in Asia Pacific,” Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS, pp. 693–698, 2013, doi: 10.1109/ICPADS.2013.123.
3. A. S. Sidhu, S. R. Balakrishnan and S. K. Dhillon, "HPC+Azure environment for bioinformatics applications," 2013 IEEE International Conference on Bioinformatics and Biomedicine, 2013, pp. 12-15, doi: 10.1109/BIBM.2013.6732615.

# Abstract

A clustered computing system is a High Performance Computing (HPC) system that is used to capture the combined processing power of all cluster nodes. Clusters are developed to process large data sets and programs by breaking them down into smaller tasks. These tasks are then sent for processing to individual computing nodes. Clusters are used for tasks such as processor-intensive scientific computing work. Hosting all the HPC nodes in a single premise has always been the simplest HPC implementation. It is designed using costly computer nodes that have a high number of CPU cores and plenty of computer memory. In terms of scaling and upgrading of an on-premise only HPC cluster, organizations normally just purchase more nodes, upgrade CPUs, upgrade RAM or storage for each node. However, buying more hardware is quite ineffective as it takes time to procure new hardware and upgrade the existing HPC systems, yet the demand for computing power may be immediate. New hardware could also be potentially underutilized. As HPC clusters are normally built with peak demand in mind, organizations try to anticipate demand spikes and to make sure that the HPC system can handle that maximum or peak load. However, demand spikes are normally only seasonal and as a result, the extra CPU and memory resources are left underutilized. One alternative to physically upgrading an HPC cluster is to utilize cloud computing. Cloud Computing is a new paradigm for computing infrastructure establishment. This paradigm moves the placement of the computing infrastructure to the internet to lessen the expenses related to hardware and software resources management. The HPC+Cloud framework has been implemented to enable on-premise HPC jobs to use resources from cloud computing nodes. As part of designing the software framework, Public Cloud providers: Amazon AWS, Microsoft Azure and NeCTAR have been benchmarked and Microsoft Azure has been determined to be the most suitable cloud component in the proposed HPC+Cloud software framework. Finally, an HPC+Cloud cluster was built using the HPC+Cloud software framework and then was validated by conducting HPC processing benchmarks. The HPC benchmarks, namely the most important ones being OpenMP and High Performance Linpack, have demonstrated that the HPC+Cloud cluster can support, sustain, and complete High Performance Computing workloads successfully.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

This chapter discusses the concept of HPC+Cloud and details the reason this implementation is suitable to solve the current shortcomings of on-premise HPC clusters. It introduces concepts of cloud computing and high performance clusters (HPC) and how these separate computing models are combined in the HPC+Cloud implementation framework.

## 1.1 Motivation

A clustered computing system[1] is a High-Performance Computing (HPC) system that is used to capture the combined processing power of all cluster nodes. Clusters are developed to process large data sets and programs by breaking them down into smaller tasks. These tasks are then sent for processing to individual computing nodes. Clusters are used for tasks such as modeling, data analytics, web services, data mining, and many other types of processor-intensive work. Hosting all the HPC nodes in a single premise has always been the simplest HPC implementation. It is designed using costly computer nodes that have a high number of CPU cores and plenty of computer memory. Organizations typically only buy more nodes, update CPUs, upgrade RAM or storage for each node when it comes to scaling and upgrading an on-site only HPC cluster[2]. It is very ineffective to purchase more hardware as it takes time to procure new hardware and update the current HPC systems. New hardware may also be underused theoretically. However, the demand for computing power could be immediate. As HPC clusters are typically designed with peak demand in mind, organizations are trying to predict demand requests and ensure that the HPC system can support the unique limit or peak load. Demand spikes, however, are typically only seasonal and the extra CPU and memory resources are thus left underutilized during off-peak season. Therefore, using cloud computing to physically upgrade an HPC cluster is one of the alternative solutions. Cloud computing[1] is a new model for computer technology development. This model shifts the location of the computer infrastructure to the internet to minimize the costs associated with it. The HPC+Cloud software framework[3] enables on-premise HPC systems to scale up without expensive capital investments in computer hardware. Organizations that currently have existing HPC clusters will be able to use the HPC+Cloud framework to scale and upgrade their HPC clusters at a lower cost compared to other HPC implementation architectures. Existing HPC clusters can also increase the breadth of applications supported as now additional storage and computing power are easily available on demand from the cloud. As part of this research study, the HPC+Cloud architecture will be deployed in a HPC cluster. It will have the significant impact of increasing the amount of processing power available to researchers and students for scientific computing. Scientific computing (also known as computational science) is a branch of science that deals with developing mathematical models and numerical solution approaches, as well as employing computers to assess and solve scientific and technical problems. In practice, it refers to the use of computer simulation and other forms of computation to solve issues in a variety of scientific fields. Traditional HPC clusters are built to amplify computing power and allow higher amounts of computing power to be pooled together to solve scientific computing problems. The HPC+Cloud framework will help the adoption of cloud resources for the HPC cluster and allay privacy concerns by only allowing processes that meet user privacy requirements to migrate to cloud nodes for processing. Organizations that create new HPC clusters using the HPC+Cloud architecture no longer need to make a large up-front investment in expensive high performance and high storage machines just to get started with high performance computing.

In summary:

- ✓ There is a demand for a cost-effective way to scale up and upgrade HPC clusters and the existing approach of buying new hardware to upgrade computing clusters are expensive if underutilized.
- ✓ The HPC+Cloud architecture provides a seamless way for existing HPC users to use cloud resources for scientific computing without having to learn a new method of working.

## 1.2 Research Gap

HPC+Cloud[4] is a proposed HPC software framework that allows current HPC applications to scale out into the public cloud. The local HPC application can leverage the extra computing nodes from the public cloud. In previous years, on-premise HPC scaling was carried out by either manual hardware procurement to scale on-premise hardware[5], or by grid computing[6], or by totally abandoning the on-premise cluster and moving the HPC onto the cloud[7]. The proposed HPC+Cloud framework is an intermediate solution that utilizes existing on-premise HPC hardware. Work performed by Li et al.[8] which proposes the MyCloud framework that works in concept in a similar way to the proposed HPC+Cloud framework. However, the MyCloud framework is specifically geared to the Openstack cloud platform built on GNU/Linux, meanwhile our proposed HPC+Cloud framework is geared to the Windows HPC cluster as Curtin University's HPC cluster is currently built on Windows HPC[9]. Currently, Microsoft[10] does provide tools for migrating jobs from on-premise to the cloud, however the solution is a piecemeal solution that is applied manually by the administrator and does not check the suitability of the process before migrating it to the cloud. Another similar framework that provisions cloud nodes is created by Ding et al.[11] which uses the Windows HPC cluster based on Microsoft's approach[10]. However, unlike Li et al.[8], Microsoft[10] and Ding et al.[11], the proposed HPC+Cloud adds an element of administrative control allowing administrators to make sure only suitable and allowable HPC jobs that meet privacy and legal concerns can be migrated to the Cloud. Privacy and legal concerns are an issue when adopting cloud based technology as seen in [12] and [13]. Another more recent framework that achieves similar goals as the HPC+Cloud framework is the WoBinGO software framework proposed by Simic et al.[5] in 2019. This framework allows migration from the on-premise cluster to HPC. However, it is built only for the Linux Openstack platform and does not check the suitability of the process before migrating it to the cloud as opposed to the proposed HPC+Cloud framework. Another advantage of the proposed HPC+Cloud framework is that cloud nodes are promptly decommissioned and deallocated when not in use, unlike Li et al.[8], Microsoft[10] and Ding et al.[11] approaches to migrating on-premise jobs to cloud nodes. This is an effective way to save cost as cloud resources are charged per-minute on time regardless of usage[14].

## 1.3 Research Questions

a) What is the best available public cloud that can be used to build an HPC+Cloud cluster?

b) How to model a software framework that can build an HPC+Cloud cluster using the best available public cloud and on-premise cluster, and handle legal and privacy data issues in processing HPC jobs in the cloud?

c) How to determine HPC+Cloud cluster's relative performance to existing architectures such as HPC on Premise and HPC on Cloud?

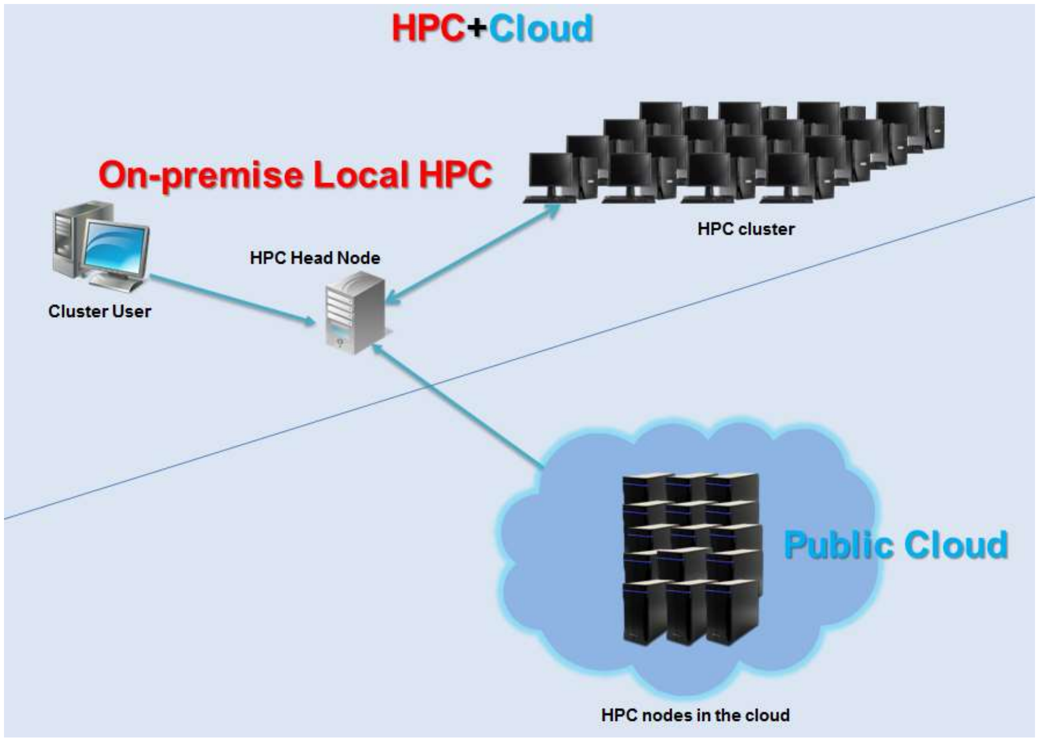


Figure 1.1 HPC+Cloud

## 1.4 Aim, Objectives, and Scope

The primary aim of this research is to design, implement and deploy high performance computing (HPC) solution that utilizes on-premise HPC compute nodes and HPC on cloud nodes as seen in Figure 1.1. The key objectives of this research are as follows:

1. To determine the right public cloud provider to combine with the HPC+Cloud software framework architecture.
2. To design and implement an HPC cluster that works seamlessly for scheduling jobs on the integrated public cloud using the HPC+Cloud software framework.
3. To ensure the proposed framework must be able to filter the HPC processes that can be scalable to the cloud as there are privacy, and legal concerns.

4. To validate the deployed HPC+Cloud software framework cluster by benchmarking its performance.

To achieve these research objectives, the scope of activities includes the following:

1. Benchmarking various public clouds to determine a suitable candidate for integrating with a local HPC cluster.
2. Configuring the local HPC cluster to use the HPC+Cloud software framework.
3. Implementing the HPC+Cloud cluster by building a test HPC+Cloud cluster.

## 1.5 Research Contributions

This research produces two major contributions as follows:

1. The proposed HPC+Cloud framework specifies and models a software framework for moving on-premise HPC jobs onto cloud nodes built on the Microsoft Azure Public Cloud platform. It further refines the work performed by Li et al.[8] that used the Openstack on GNU/Linux.
2. The proposed research also improves Ding et al.[11] ‘s work on the Microsoft Azure Public Cloud platform and Microsoft Windows HPC platform by checking the suitability of HPC processing jobs based on privacy and legal concerns before moving them to cloud nodes for processing. Privacy[12] and legal[13] concerns are main obstacles to the adoption of cloud technology for HPC computing. The HPC+Cloud software framework examines every job before it is allowed to leave the on-premise HPC cluster.

Finally, the research objective of determining the right public cloud provider to combine with the HPC+Cloud software framework architecture was achieved. To achieve this objective, HPC benchmarks were utilized and then it was concluded that Microsoft Azure Public Cloud was the best cloud node to be used with the HPC+Cloud framework. The next objective achieved is to design and implement an HPC cluster that works seamlessly for scheduling jobs on the integrated public cloud using the HPC+Cloud software framework. The framework determined the scalability of the process. Not all HPC processes could be scalable to the cloud as there are privacy, and legal concerns. The final objective that has been achieved is to validate the deployed

HPC+Cloud software framework cluster by benchmarking its performance against existing alternatives such as HPC on Premise and HPC on Cloud.

## 1.6 Thesis Structure

This chapter consists of the introduction of the research topic, where the research motivation, research gap, research aim, objectives, and scope are identified. Chapter two discusses the background information about cloud computing, high performance computing clusters, and the various existing implementation approaches that are being used currently. Discussions will be detailed on the strengths and weaknesses of current approaches and how the HPC+Cloud software framework addresses them. Chapter three then describes the algorithms and modeling of the proposed HPC+Cloud software framework in terms of flow chart and pseudocode. This is to prepare the proposed HPC+Cloud software framework for the experimental setup and discussions of experimental results in chapter four. In chapter four, the experimental results of benchmarking various public clouds to determine a suitable candidate for integrating with a local HPC cluster are discussed. Next, the HPC on-premise cluster, HPC+Cloud cluster and the HPC on Cloud clusters are benchmarked and the results are analysed and discussed. Finally, chapter five concludes the whole thesis with conclusions and recommendations for future work.

## 1.7 Summary of Chapter

In this chapter, the motivation behind this research work is presented and the proposed HPC+Cloud software framework overview is presented. However, the presented information lacks background literature support. Thus, background information and literature review shall be presented in the next chapter to clarify and identify the core information that shapes the whole idea behind the proposed HPC+Cloud software framework.

# 2 Background and Literature Review

This chapter comprises of two parts. The first part discusses the background technology needed for the HPC+Cloud, namely high-performance computing cluster and cloud computing. The second part zooms specifically on the various approaches or specifically HPC implementation architectures that have been used to scale and upgrade on-premise only HPC clusters that compete with the proposed HPC+Cloud software framework.

## 2.1 High Performance Computing

The following section is based on the author's work that has been published in the conference paper: "High Performance Computing on the Cloud via HPC+Cloud software framework," 2016 Fifth International Conference on Eco-friendly Computing and Communication Systems (ICECCS).

A networked clustered computing system can perform as a high performance computing (HPC) system[15]. The cluster Operating System is the software that enables a networked clustered computing system to harness the combined computing capability of all nodes in a cluster. Clusters are used to break down big applications and big amounts of data into smaller processing tasks. Individual computing nodes are then subsequently assigned to these tasks for processing. Despite multiple computers that are networked[16], the cluster seems to function as a single computer completing computationally heavy processing chores. Simulation, data analytics, online services, data mining, bioinformatics, and other computationally heavy processing activities[17] are solved using clusters. Clusters enhance speed and/or reliability over a single supercomputer for such workloads and are generally considerably more cost-effective than utilising a single supercomputer. Only the head node[18], which serves as a scheduler, may be accessed in an HPC cluster system. This node serves as a connection point between the user and the cluster. The cluster's head node serves as a launchpad for tasks operating in the cluster. The computing nodes[19] that are linked to the head node perform the actual processing.

Processing is completed utilizing either pipeline flows or sweep flows, depending on the kind of task delivered to the cluster. Processing workloads that execute in a sweep flow[7] in an HPC cluster are jobs that are split into processes that may be processed completely in parallel with no communication between the cluster nodes that execute each process. Sweep flow jobs are sometimes referred to as jobs that are embarrassingly parallel.

Other processing tasks in the cluster may use a pipeline flow to execute[20]. Threaded processing tasks are processed via pipeline flows. These threads are interdependent and must generally be run in a certain order. However, there are exceptions to this rule, such as when a job is partially pipelined. This occurs when a pipeline step (a processing node) completes a partial processing of the data. These partial results will then be transmitted to the next processing node, which will immediately begin processing the partial results while continuing to analyse the remaining data in parallel.

Pipeline flow tasks must communicate between nodes and processes because job threads are spread throughout nodes in the cluster. In an HPC cluster, both pipeline flow nodes and sweep flow nodes must interact with the head node to provide processing output. In an HPC cluster, latency[21] becomes a concern due to all the connections required. Parallel processing techniques and middleware frameworks[8] have been used at the software level to optimize and decrease the amount of communication overhead[22] between nodes. However, the actual network architecture still limits the majority of these software improvements. Most existing and older HPC clusters are now built utilizing Fiber Channel, which employs a combination of copper wire and high-speed fiber optic cabling[23]. InfiniBand[23], a form of communications link for data flow between CPUs and I/O devices that offers throughput of up to 54 gigabits per second, is often used in contemporary next-generation HPC clusters. InfiniBand is also scalable, having quality of service (QoS) and failover capabilities[8]. Compared to Fiber Channel, it offers a significant benefit. Fiber Channel is only a transmission channel between the HPC node's network interfaces. Meanwhile, InfiniBand skips the traditional network interface and permits direct connection between HPC cluster nodes at the CPU bus level, allowing nodes in an HPC cluster to be connected. The cost of implementing InfiniBand is the major deterrent to its use. Because the Layer 2 physical cabling in an existing HPC cluster must be removed, InfiniBand is generally employed primarily in high-end HPC clusters or cloud data centres. Using a cloud computing service to access more processing nodes and scale up the HPC cluster is one approach to take advantage of InfiniBand's low latency interconnects on a budget[24]. InfiniBand is a very expensive technology to deploy on the premise clusters. If a person wants to get the benefits of InfiniBand technology, then subscribing to a cloud service and scaling up existing on premise HPC cluster to use cloud computing nodes would be one way to get the benefits and advantages of InfiniBand at a lower cost than renovating his or her current on-premise setup and deploying InfiniBand physically on an on premise HPC cluster. Processing HPC workloads transferred to the cloud would allow most HPC jobs to benefit from the InfiniBand used in cloud data centre networks, since most Cloud Data Centres are now designed utilizing InfiniBand.

## 2.2 Cloud Computing

Cloud computing is gradually establishing itself as a new paradigm for computer infrastructure setup. This paradigm shifts the network infrastructure's location in order to reduce the costs of managing hardware and software resources. Cloud computing[1] is defined by the National Institute of Standards and Technology as an architecture that allows for ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be quickly provisioned and released with minimal management effort or service provider interaction. The usage of hosted services through the internet is referred to as cloud computing. It's also a fundamental shift in operational design, in which software is no longer bound to a single piece of physical hardware. Because of the cloud's flexibility, computing resources may be quickly transferred depending on the needs of the user.

Infrastructure as a service (IaaS), Platform as a service (PaaS), and Software as a service (SaaS) are three different types of cloud computing service models[26]. These three service models differ from traditional IT design, which often requires an enterprise to handle all IT resources on its own. When the cloud computing paradigm is chosen, IT components are handled by cloud service providers, and cost reductions are immediately realized by the enterprise[27].

There are three types of cloud computing implementation architectures: public, private, and hybrid[28]. A public cloud computing paradigm is one that delivers hosted services via the internet. Public cloud services are either free (NeCTAR)[29] or require customers to pay each time they utilize the service (Azure, Amazon EC2)[30]. Users must have their own disaster recovery and data backup procedures in place. It is well known that public cloud providers only provide the computing infrastructure such as VMs and storage, and the public using the public cloud must have their own disaster recovery and data backup procedures in place. Public cloud provider can provide those two services but there will be at extra cost on top of the cost of infrastructure. The cloud is usually multi-tenant and controlled and maintained at a data center owned by the cloud service provider. Cloud computing, which is a form of shared architecture, saves expenses, because the service provider owns the underlying infrastructure. Therefore, visibility and control are limited in a public cloud, according to Mell and Grance[1].

Behind the firewall, a private cloud is an exclusive computing paradigm that delivers services to a small number of individuals. Because certain businesses (such as banks, and hospitals) are worried about data security, they prefer private clouds to public clouds. Private clouds are among the least widely used of all cloud architectures, owing to high hardware and installation costs[31]. Although this cloud is not very cost-effective, it offers the best level of security when compared to other cloud models.

One or more private clouds plus one or more public clouds combined make up a hybrid cloud. It's a scenario in which an organization manages certain resources internally on a private cloud while the remainder is handled externally on a public cloud[32]. Because the local infrastructure interacts with the processing power of a public cloud, hybrid cloud helps companies to maximize the use of their IT infrastructure while lowering IT costs. The length of load peaks is relatively brief while utilizing the public cloud, which compensates for the high premium imposed by the public cloud provider, making the hybrid cloud more cost efficient than using the private cloud alone[33]. To address the latencies of internet connections, a hybrid cloud must be developed specifically to synchronize data and activities across the private and public clouds[34]. Hybrid cloud technology, specifically the application layer, transport layer, and session layer stack optimizations[35] that allow private and public clouds to seamlessly synchronize with each other[36], can now be used to connect a local on-premise HPC cluster with cloud-based HPC processing nodes, hence the term HPC+Cloud.

The main benefit of the proposed HPC+Cloud paradigm is the ability to grow the HPC cluster by on-demand provisioning of new virtual processing nodes from the cloud. The proposed HPC+Cloud paradigm inherits all of the benefits of the hybrid cloud, but with less administrative overhead than a hybrid cloud because there is no need to operate a private cloud on-premise. Despite the benefits of combining public and private clouds, hybrid clouds need a significant investment in network equipment to set up a private cloud on-premise. The proposed HPC+Cloud framework design, on the other hand, eliminates the need to construct a private cloud in order to use HPC+Cloud. To setup the current HPC, all that is required is to supply processing cloud nodes using the HPC+Cloud software framework, all of which will be discussed in further detail in the next section.

## 2.3 Existing HPC Implementation Architectures

HPC systems are clustered systems that are created to execute computationally heavy tasks using specialised designs. The various implementation architectures used in HPC clusters are discussed in this section.

### A. On-Premise Local HPC cluster

As illustrated in Figure 2.1, the simplest HPC deployment has traditionally been to host all of the HPC nodes in a single premise. To guarantee that the HPC cluster is powerful, costly compute nodes with a large number of CPU cores and ample computer memory are used. Furthermore, the nodes interact through high speed computer networks to guarantee high-speed, low-latency communications between HPC computing nodes and the head node[14]. Organizations often acquire extra nodes, update CPUs, RAM, or hard drive storage for each node when growing and upgrading an on-premises only HPC cluster.

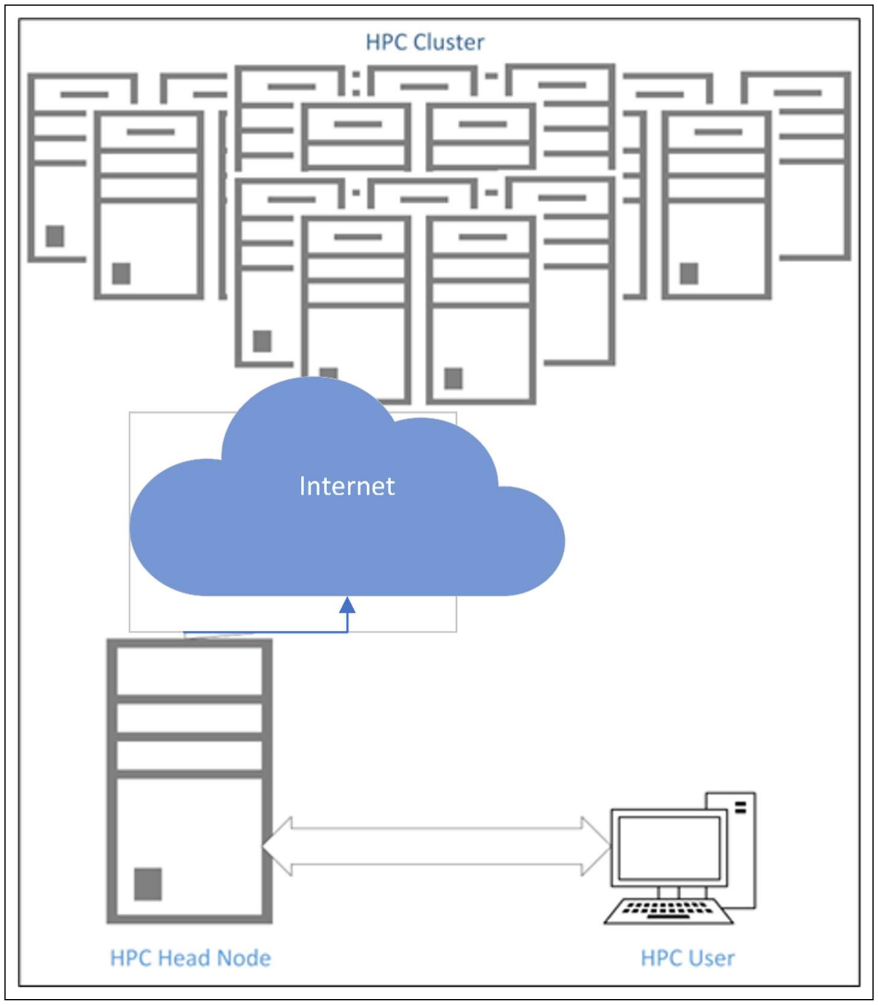


Figure 2.1 On-Premise Local HPC cluster

Physically updating hardware on an HPC cluster, on the other hand, has drawbacks. Purchasing extra hardware is useless since procuring new hardware and upgrading current HPC systems takes time[37]. This is time that researchers cannot afford to waste since the need for computer power may be urgent, and demand for processing resources is growing faster than supply[38]. On the other hand, because most high-performance computing clusters are constructed with peak demand in mind, newly procured hardware upgrades may be underused. In other words, as they strive to predict demand surges and ensure that the HPC system can manage the highest or peak load. Utilization Demand peaks, on the other hand, are usually just seasonal, and the increased CPU and memory resources are left unused during off-peak seasons [39].

**B. HPC cluster implemented on grids.**

Grid computing is a type of distributed computing technology that uses the internet to supply computing power[40]. Figure 2.2 depicts a simple implementation of an HPC cluster based on grid computing. It is made up of a grid of clusters that are dispersed around the globe in diverse geographical areas. To make HPC software compatible with the grid computing architecture, a grid computing solution necessitates rewriting HPC software and developing a middleware software. This code reworking and middleware development is not a simple task. Grid takes advantage of distant clusters located all over the world. Latency becomes an issue as a result of this. Although there is a built-in grid software middleware available to aid with latency, the grid cluster's wide geographic diversity remains a significant issue to overcome[41]. Grid does not address the issue of scaling HPC cluster hardware and software. In comparison to the proposed HPC+Cloud design, which allows HPC hardware to be updated on demand, decreasing the cost of upgrading an HPC cluster, the grid still requires significant hardware and software[41] costs to expand and upgrade. Furthermore, cloud hardware is controlled by cloud service providers with excellent fault tolerance and scalability [44]. Typical grid computing systems, on the other hand, provide no such assurances and offer services on a best-effort basis[42].

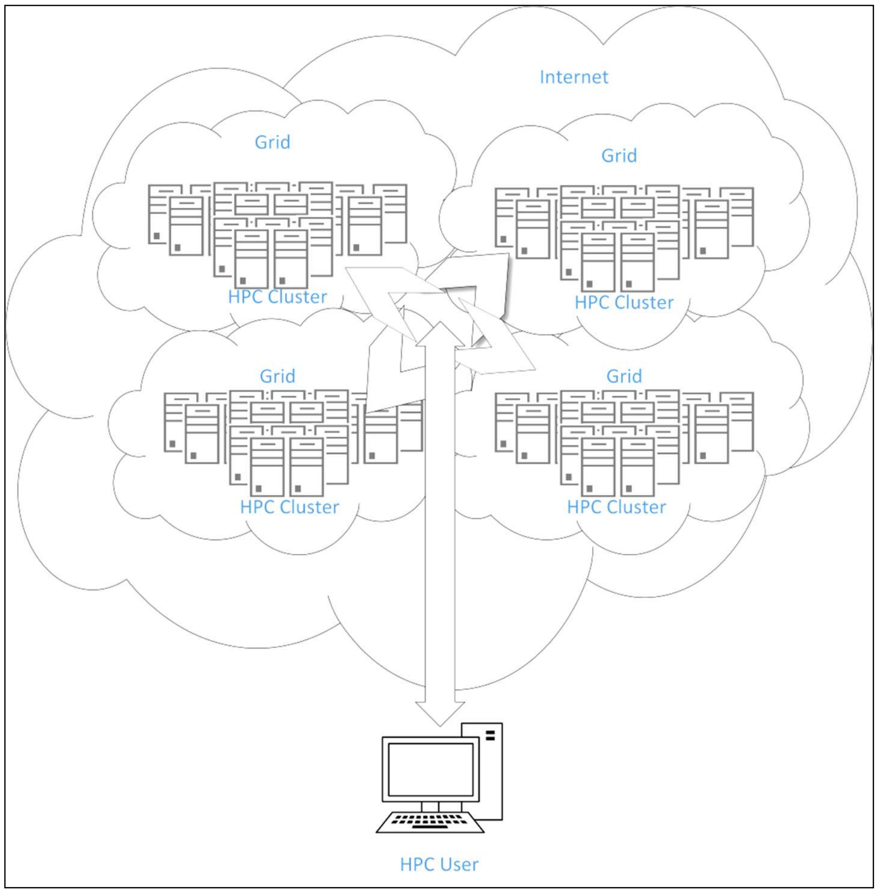


Figure 2.2 HPC cluster implemented on grid

**C. HPC cluster implemented on the Cloud**

A fully cloud-based HPC cluster is another HPC implementation design. Both the cluster head node and the processing nodes are cloud-based[43]. Figure 2.3 shows a HPC on cloud cluster setup. When an HPC cluster is deployed on a public cloud, the public cloud architecture, which is elastic in nature, takes care of the expanding and updating the capabilities of an HPC cluster[2]. The upfront expense of using the public cloud alone to replace a complete existing on-premise HPC infrastructure is the primary flaw in using HPC on Cloud to upgrade an HPC cluster. Because all current on-premise local hardware and software must be virtually replicated on the Public Cloud IaaS, add to that the cost of discarding existing hardware from the local on-premise HPC cluster, and because the entire cluster processing will take place outside of the local on-premise boundary, the cost per unit of computing for HPC on cloud would be quite high[42].

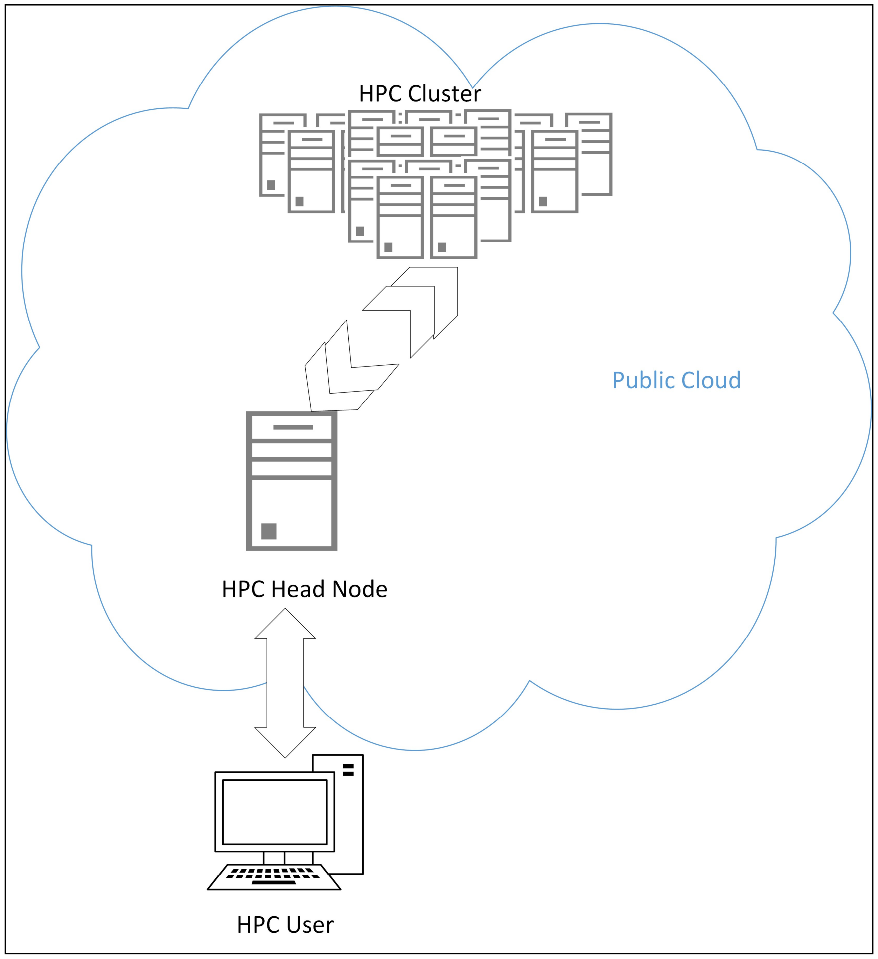


Figure 2.3. HPC cluster implemented on the cloud

### D. HPC cluster implemented on the Hybrid Cloud

As previously stated, a hybrid cloud is made up of one or more private clouds and one or more public clouds. Figure 2.4 shows how a hybrid cloud may be used to run an HPC cluster. An environment in which an organization manages some resources internally on the private cloud while the rest is handled externally on the public cloud. In the hybrid cloud, HPC clusters can be implemented[44]. The hybrid cloud appears to be a suitable option since workloads from on-premise private clouds may be effortlessly moved to the public cloud on demand, and software licenses for the private cloud are only purchased once. The primary issue is the cost of establishing a private cloud on a local level[45]. The hardware and software costs of establishing a private cloud on a company's premises are exorbitant. There's also the administrative cost of running a private cloud, as well as the public cloud's HPC cluster and virtual HPC nodes[46]. The present on-premise HPC cluster's hardware and software architecture would have to be modified to suit the private cloud.

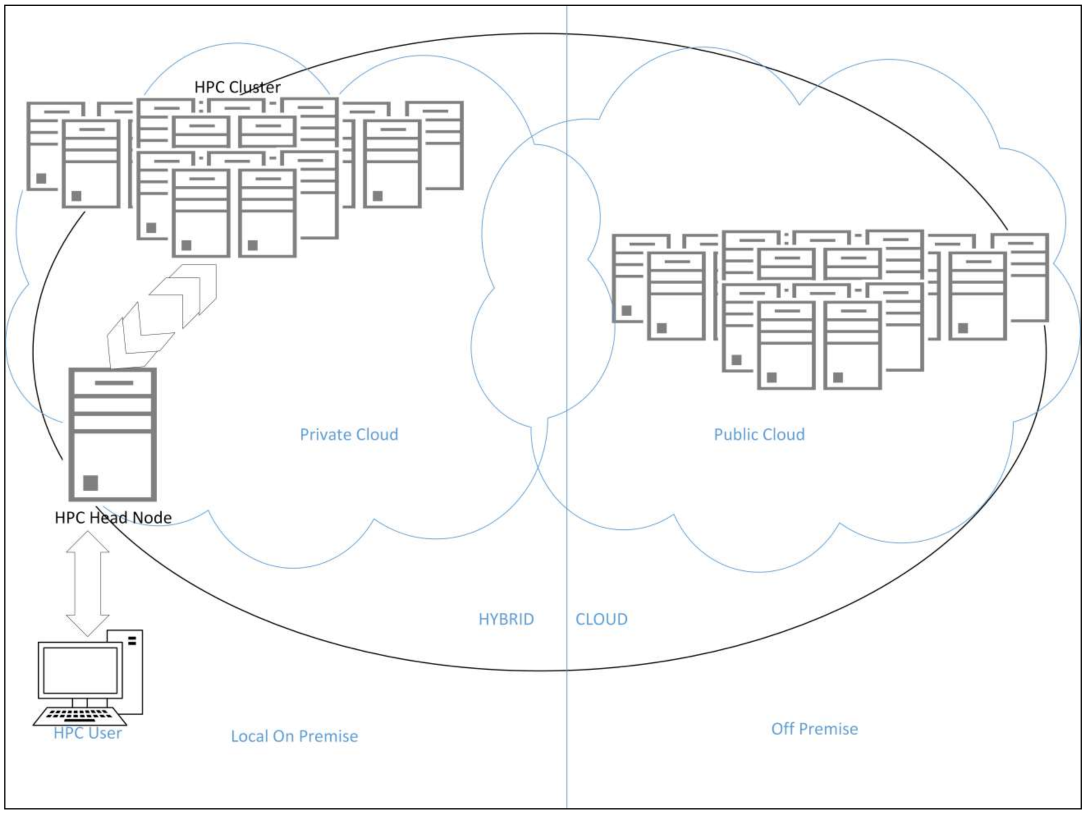


Figure 2.4. HPC cluster implemented on the hybrid cloud

## 2.4 HPC+Cloud Contrasted with Existing Implementation Architectures

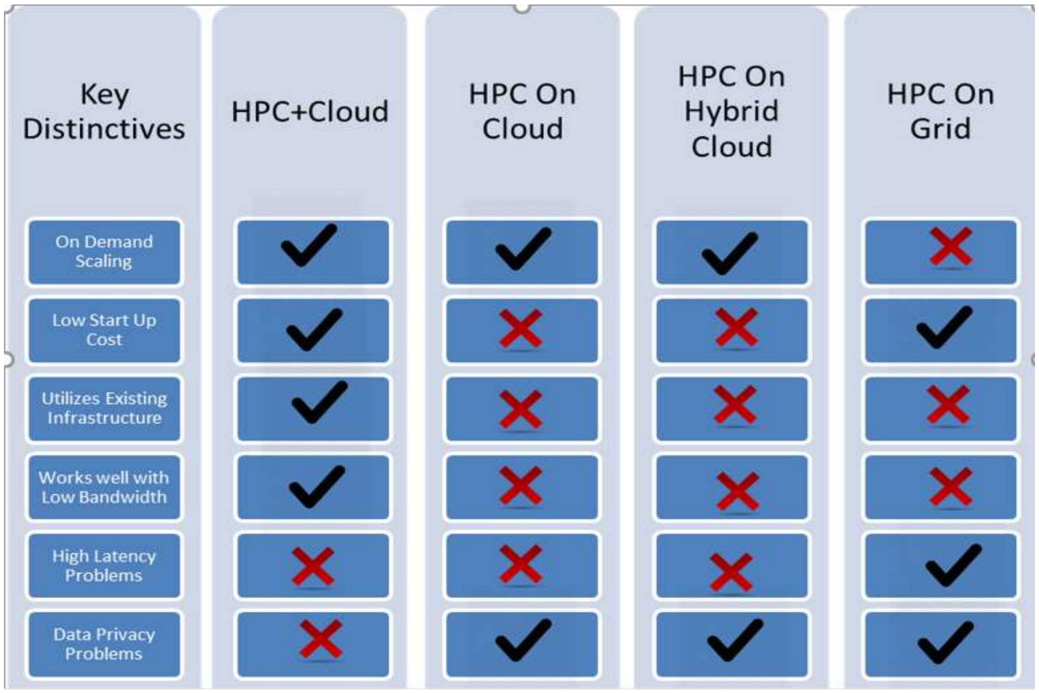
The proposed HPC+Cloud architecture's main distinguishing feature is that it allows for on-demand scalability at a low initial cost. The relatively cheaper setup cost is due to the fact that it makes use of existing on-premise HPC cluster processing nodes while also utilizing cloud processing nodes[47]. The proposed HPC+Cloud architecture utilizes existing hybrid cloud technologies to alleviate the high latency concerns that cloud technology has. The proposed HPC+Cloud also takes advantage of existing architecture application layer, transport layer, and session layer stack optimizations that allow private cloud and public cloud to synchronize flawlessly.

HPC On Cloud appears to be a fantastic option at first sight, as it also offers on-demand scalability. However, moving an existing HPC cluster fully to the cloud is too expensive, particularly in terms of start-up expenses. This is due to the fact that HPC on Cloud requires current on-premise hardware and software to be virtually duplicated in the cloud[48]. In contrast, the proposed HPC+Cloud simply uses the cloud to provide more processing nodes as needed, and therefore only has a portion of its infrastructure in the cloud.

HPC on Hybrid Cloud appears to be a viable option as well, since it allows for on-demand scaling[49]. However unlike the proposed HPC+Cloud, which uses existing on-premises infrastructure, HPC on hybrid cloud requires considerable additional hardware and software investment[3] to transform existing on-premises equipment to become a Private Cloud in order to build an Hybrid Cloud. The expense of running a Private Cloud as part of the Hybrid Cloud inside an existing organization might also be exorbitant depending on the computational and redundancy needs.

HPC on grid does enable on-demand scalability, but only to a limited extent, because processes can be spread throughout the grid cluster's numerous nodes[50]. This is a big issue because grids are generally dispersed geographically[6], resulting in latency issues that influence HPC processing time while on the grid. Grid computing platforms provide no assurances about demand scaling and deliver services with the best effort possible. In contrast to the proposed HPC+Cloud architecture, which includes service level agreements with bandwidth and uptime assurances from the cloud service provider. Figure 2.5 depicts an overview of the differences between the various implementation architectures.

When compared to alternative HPC implementation architectures, the HPC+Cloud implementation architecture offers the most benefits. Relatively lower start-up costs, ability to leverage existing infrastructure from the on-premise cluster, and ability to work effectively even with relatively lower bandwidth are three major aspects that distinguish HPC+Cloud from alternative architectures.



| Key Distinctives | HPC+Cloud | HPC On Cloud | HPC On Hybrid Cloud | HPC On Grid |
|---|---|---|---|---|
| On Demand Scaling | ✓ | ✓ | ✓ | ✗ |
| Low Start Up Cost | ✓ | ✗ | ✗ | ✓ |
| Utilizes Existing Infrastructure | ✓ | ✗ | ✗ | ✗ |
| Works well with Low Bandwidth | ✓ | ✗ | ✗ | ✗ |
| High Latency Problems | ✗ | ✗ | ✗ | ✓ |
| Data Privacy Problems | ✗ | ✓ | ✓ | ✓ |

Figure 2.5. Key Distinctives of various HPC implementation architectures

## 2.5 Summary of Chapter

This chapter is divided into two sections. The first section covers the HPC+Cloud background technologies, such as high-performance computing clusters and cloud computing. The second section focuses on the several techniques, or HPC implementation architectures, that have been utilised to grow and update on-premise only HPC clusters that compete with the proposed HPC+Cloud software framework. Different implementation architectures were compared and contrasted with the proposed HPC+Cloud.

In the next chapter, the HPC+Cloud software framework will be examined in detail by describing the algorithms and modeling the software framework in flowchart and pseudocode.

# 3 Methodology of the HPC+Cloud Software Framework

The main goal of the HPC+Cloud software framework is to allow the sending of HPC applications jobs to the cloud when the resource on the local HPC cluster exceeds a prefixed utilization threshold. Also, the framework selectively chooses which processes can be migrated to the cloud and if it is scalable according to user requirements.

HPC+Cloud is the module that enables the HPC to link up with the public cloud. The HPC+Cloud consists of the following processes as illustrated in Figure 3.1. Workload Monitoring (A) monitors the HPC cluster utilization threshold. When the utilization threshold is reached by the HPC cluster, HPC jobs are sent to Batching (B) component. Batching (B) component analyses the suitability of the job by checking that a job's Privacy Flag Variable. This variable determines the job's suitability for entering the cloud. The Sending part of (C) sends the suitable processes to the Cloud input Queue for processing in the cloud. Upon process job completion, process jobs are placed on Cloud Output Queue. Worker Output monitoring component (D) continuously monitors the Cloud Output Queue. Once there are completed jobs in the Cloud Output Queue, the Worker Output Monitoring component (D) triggers the Output Storage Component (E) to copy and store the data generated by the job in the local storage facility. Bad Request Timeout Sweeper (F) runs continuously in the background during steps A, B, C and D. It Appends the Job_Pid of jobs that are not migrated to the system log. Jobs that do not migrate to the cloud continue their processing on on-premise cluster.

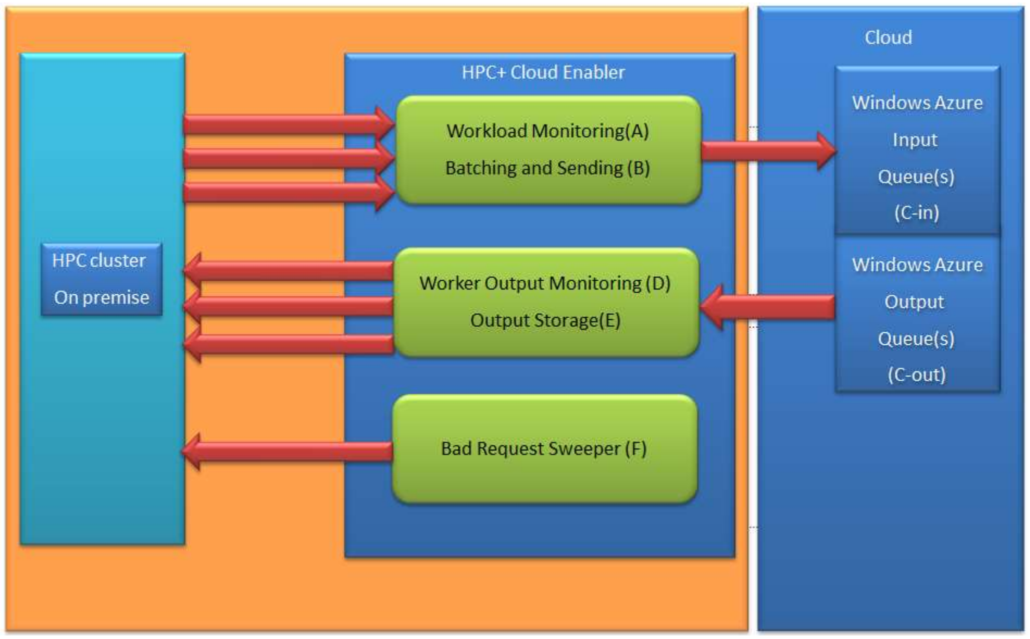


Figure 3.1. HPC+Cloud framework components

## 3.1 HPC+Cloud Software framework Algorithm

The framework HPC+Cloud framework is designed only to process one job at a time. If multi jobs are needed, another instance of the application is run in parallel. Every time a new job comes, a new instance of the application is run. If there is sharing of common resources across application instances, the operating system takes care of it.

An HPC application job consists of a compute task or a data transfer task and the job's resource requirements. An HPC job j can be represented by the tuple that describes the HPC job:

Job Description:

Desc(j) = ⟨$\mathbf{Res}_j$, $\mathbf{Exec}_j$, $\mathbf{Env}_j$, $\mathbf{D}_j$, $\mathbf{S}_j$, $\mathbf{Job\_Pid}_j$ ⟩

Where:

- $\mathbf{Res}_j$ are the resources such as compute nodes and memory allocated to the job.
- $\mathbf{Exec}_j$ is the application to be executed,
- $\mathbf{Env}_j$ is the software stack (e.g., operating system and libraries),
- $\mathbf{D}_j$ is the Data of the job.

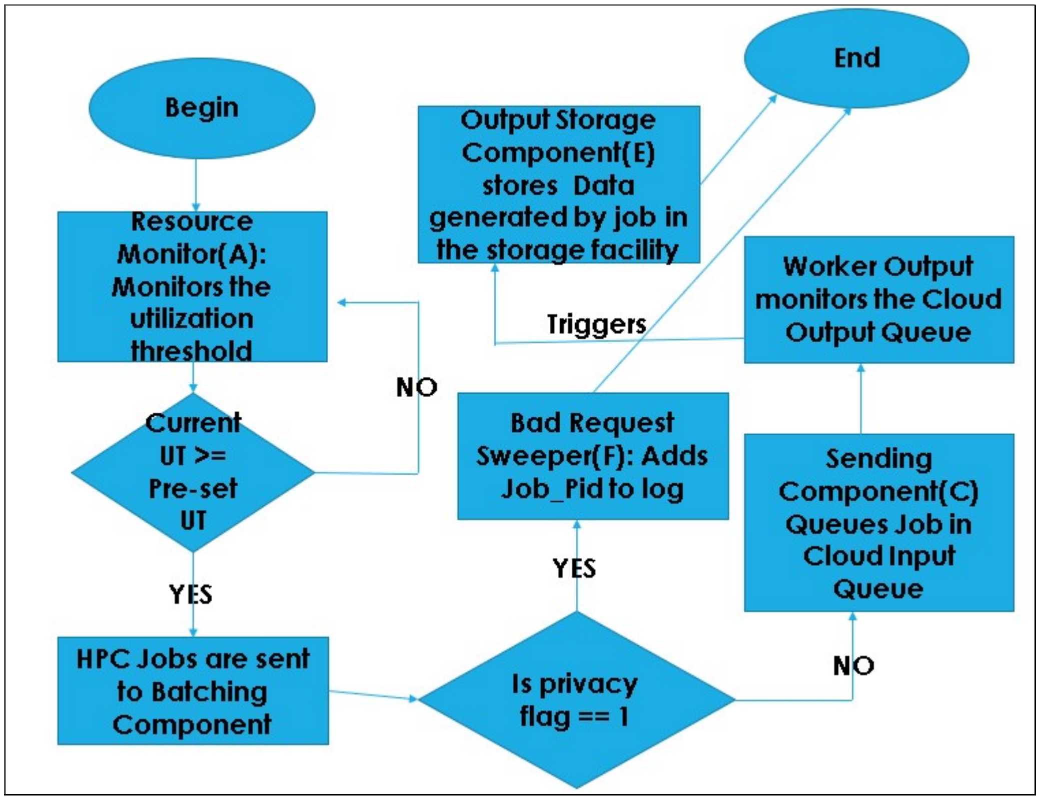
- **S**$_j$ is the suitability of the job as determined by the user, application, or the HPC+Cloud is stored into the Privacy_Flag variable.
  if a job is not allowed to migrate to the cloud for privacy, or legal reasons the status is stored here in this flag variable.
- **Job_Pid**$_j$ is the process Identifier of the job.

The Resource Monitor (A) runs a service monitoring the HPC system utilization threshold (UT).

1. When UT is reached by the HPC system, HPC jobs are sent to Batching(B) and Sending (C) components.
   a. Batching component analyzes the suitability of the job (from the information found in S$j$) according to the following constraint: Job meets user requirements of privacy as stated in privacy flag variable.
   b. The Sending component (C) queues and sends the job to the Cloud input Queue (C-In) based on the process id (Job_Pid) of the HPC job.
2. Upon Job completion, jobs are placed on Cloud Output Queue (C-Out).
3. Worker Output monitoring component (D) continuously monitors the Cloud Output Queue(C-Out).
   a. Once there are completed jobs in the Cloud Output Queue (C-Out), Worker Output Monitoring component (D) triggers the Output Storage Component (E) to copy and store the Dj generated by the job in the local storage facility.
4. Bad Request Timeout Sweeper (F) runs continuously during steps 1,2 and 3. It appends the Job_Pid of jobs that are not migrated to the system log. Jobs that do not migrate to the cloud continue their processing on on-premise cluster.

## 3.2 HPC+Cloud Software Framework Flowchart

Figure 3.2 illustrates the HPC+Cloud software framework algorithm described in Section 3.1, in the form of flowchart.



Begin
Resource Monitor(A): Monitors the utilization threshold
Current UT >= Pre-set UT
NO
YES
HPC Jobs are sent to Batching Component
Is privacy flag == 1
YES
NO
Bad Request Sweeper(F): Adds Job_Pid to log
Sending Component(C) Queues Job in Cloud Input Queue
Worker Output monitors the Cloud Output Queue
Triggers
Output Storage Component(E) stores Data generated by job in the storage facility
End

Figure 3.2. Software framework flowchart

## 3.3 HPC+Cloud Software Framework Pseudocode

The following tables contain the HPC+Cloud software framework that is further refined into detailed pseudo code.

Table 3.1. HPC+Cloud Main Function Pseudocode

```
Pseudo-code: HPC+Cloud Main Function

Input: Job Desc($_j$) = ⟨$Res_j$, $Exec_j$, $Env_j$, $D_j$, $S_j$, $Job\_Pid_j$ ⟩

Output: Select Jobs suitable for migration to the cloud.

START MAIN

SET Utililization_Threshold to User_Defined_Preset

CALL ResourceMonitor FUNCTION Find Current Utilization Value RETURN Current_Utilization

    IF Current_Utilization is bigger than or equal to Utililization_Threshold

        THEN

        CALL Batching FUNCTION Select a suitable Job RETURN Job_Pid

        CALL Sending   FUNCTION with INPUT: Job_Pid

        CALL WorkerOutPutMonitor FUNCTION

    ENDIF

END MAIN
```

Table 3.1 describes the main function of the HPC+Cloud framework. The main function accepts user input of an HPC processing job Desc(j). The utilization pre-set is set according to user defined pre-set. Meaning the user can determine ahead of time at what level of on-premise cluster utilization should be reached before processes can be migrated to the cloud for processing. The ResourceMonitor function checks the on-premise utilization, if the on-premise cluster current utilization is bigger than or equal to the utilization pre-set utilization threshold, the process of migrating jobs to the cloud is begun. Next the Batching function is called to determine if the process is suitable for migration, the Batching function returns the process id if the job is allowed to migrate. Next the Sending function uses this process id and migrates the job to the cloud. Next the WorkerOutPutMonitor function returns the state of the job. Whether the job is completed or still processing.

Table 3.2. HPC+Cloud Resource Monitor (A) Pseudocode

```
Pseudo-code: Resource Monitor (A)

Input: None.

Output: Current Utilization Value of the on-premise cluster.

START FUNCTION

CALL HPC_Cluster_Utilization_Library FUNCTION

IF Current Utilization is NULL Return Error Terminate Proc with
Utilization_Library_ERRMSG

RETURN Current_Utilization

END FUNCTION
```

In Table 3.2, the resource Monitor function is called by the main function. The function then uses the built-in library function for the on-premise HPC to get the given cluster utilization.

Table 3.3. HPC+Cloud Batching (B) Pseudocode

```
Pseudo-code: Batching (B)

Data Processed: Job Desc($_j$) = ⟨$Res_j$, $Exec_j$, $Env_j$, $D_j$, $S_j$, $Job\_Pid_j$ ⟩

Output: Returns Process id (Job_Pid) of the job suitable for cloud migration

START FUNCTION

While (Not End of Jobs List in Cluster)

    CALL SystemLibrary FUNCTION Read_cluster_Job_data Return Desc(j)
    CALL SystemLibrary Extract Job Data FUNCTION with INPUT: Desc(j) RETURN Job_Pid, Privacy_Flag
IF Job_Pid is NULL OR Privacy_Flag is NULL `THEN Return Error Terminate Proc with SystemLibraryExtract_ERRMSG
IF PrivacyFlag equal to 0
THEN
    RETURN Job_Pid
ELSE
    RETURN ERROR: Job is unsuitable for Migration. Trigger: Inform HPC Head node
    CALL Bad Request Timeout Sweeper FUNCTION INPUT: Job_Pid
END WHILE LOOP
END FUNCTION
```

Table 3.3 highlights the Batching function, which checks the job for its suitability to be migrated to the cloud from the list of jobs currently on the cluster. It achieves this by checking the privacy flag variable. SystemLibrary Extract function is used to extract the flag variable from Desc(j).

This ensures only HPC jobs that meet user privacy requirements be migrated to the cloud. If the Job is suitable the process id of the job that is returned. The way this mechanism works is that users can determine ahead of time which jobs are exclusively meant to stay within the organization. The Privacy flag variable is initialised by the user when the HPC job is created and stored in the data structure Desc(j).

Table 3.4 HPC+Cloud Sending (C) Pseudocode

```
Pseudo-code: Sending (C)
Input: Job_Pid
```

```
Output: Sends suitable Jobs to Cloud Input Queue (C-In)
START FUNCTION
SET INPUT=Job_Pid
    CALL Cloud library Cloud_Input.Enqueue FUNCTION with INPUT: Job_Pid
If Process_Status is Failed to Enqueue Return Error Terminate Proc with Cloud_Input.Enqueue_ERRMSG
RETURNING Process_Status
END FUNCTION
```

Table 3.4 highlights the Sending function which sends HPC processing jobs to the cloud for processing via Cloud library Cloud_Input.Enqueue function which queues the job for processing on the cloud nodes. Once This process is completed a process status is returned denoting the job has been successfully migrated to cloud.

Table 3.5. HPC+Cloud WorkerOutPutMonitor (D) Pseudocode

```
Pseudo-code: WorkerOutPutMonitor (D)
Output: Monitors Cloud Output Queue (C-Out)
START FUNCTION
Flag set to 1
WHILE Flag == 1
    CALL Cloud library Cloud_OutPut.Monitor FUNCTION RETURNING Process_Status
If Process_Status is 1 THEN Flag is set 0
ENDWHILE
    IF Process_Status equal to 1 // Completed Job is on Queue.
    THEN
        CALL Output Storage FUNCTION
    ELSE Return Error Terminate Proc with Cloud_OutPut.MonitorERRMSG
END FUNCTION
```

Table 3.5 highlights the WorkerOutPutMonitor which continuously monitors the Cloud Output Queue by utilizing Cloud library Cloud_OutPut.Monitor function which returns status of the job via process status variable. If the job is completed the Job is put on Cloud Output Queue and the Output Storage is triggered to write Completed job to the local storage.

Table 3.6. HPC+Cloud Output Storage (E) Pseudocode

```
Pseudo-code: Output Storage (E)
Output: Sends Completed Jobs to Local Storage
START FUNCTION
CALL Cloud library Cloud Output Dequeue FUNCTION`RETURNING HPC_Job
    IF HPC_Job equal to 1 AND NOT NULL
    THEN
        CALL SystemLibrary WriteIO FUNCTION with Input:HPC_Job
RETURN WriteSuccessStatus
        IF WriteSuccessStatus is 0
        THEN Return Error Terminate Proc with WriteIO_ERRMSG
        IF WriteSuccessStatus is 1
        THEN CALL Cloud_Nodes_DeAllocate RETURN DeallocateSuccess
        IF DeallocateSuccess is 0
        THEN Return Error Terminate Proc with DeAllocate_ERRMSG
        ELSE IF HPC_Job is NULL Return Error Terminate Proc with Cloud_Output_Dequeue_ERRMSG
END FUNCTION
```

Table 3.6 highlights the Output Storage function which Sends Completed Jobs to Local Storage. A series of validations are conducted to ensure data is dequeued from the cloud and written to local storage medium on the on-premise cluster accurately.

Table 3.7 HPC+Cloud Bad Request Sweeper (F)

```
Pseudo-code: Bad Request Sweeper (F)
Input:Job_Pid
Output: Writes data System logger.
START FUNCTION
    CALL SystemLibrary WriteIO FUNCTION Input: Job_Pid
        // Appends Job_Pid to System Logger.
END FUNCTION
```

Table 3.7 highlights The Bad Request Sweeper Component. This function appends the Job_Pid of jobs that are not migrated to the system log. Jobs that do not migrate to the cloud continue their processing on on-premise cluster.

## 3.4 Implementation Method

First, an on-premise Windows HPC cluster was built, and Microsoft Azure Cloud nodes were allocated to the cluster. The HPC+Cloud framework code was implemented and installed on the head node of the cluster so that it can interface the Windows HPC scheduler. Then users submit jobs to the HPC using the job submission script. Users can edit the privacy flag to flag the job so that the HPC+Cloud framework is allowed to filter the job and stop it from migrating to the cloud. The rest of the jobs can be migrated to the cloud to use HPC resources on the cloud.

## 3.5 Summary of Chapter

In this chapter, the HPC+Cloud software framework is modeled using algorithm descriptions, flowchart, and pseudo code. With this modelling completed, the HPC+Cloud framework was implemented and then an HPC+Cloud cluster was created by using the framework.

# 4 Experimental Results and Discussions

In this chapter, benchmarking was first performed to choose a public cloud provider that will be used with the HPC+Cloud cluster. Upon choosing the public cloud component, the proposed HPC+Cloud framework was implemented and then an HPC+Cloud cluster was built based on the framework. Next, the functionality of the HPC+Cloud cluster was benchmarked to show that the HPC+Cloud cluster can support, sustain, and complete High Performance Computing workloads successfully.

## 4.1 Choosing a Public Cloud provider to be the Cloud component of the HPC+Cloud through benchmarking

The purpose of benchmarking cloud platforms is to determine which cloud platform is suitable for implementing the HPC+Cloud algorithm proposed in Chapter 3. After the HPC+Cloud framework is implemented, the next step is to build an HPC+Cloud cluster. This HPC+Cloud cluster is benchmarked to test its functionality. The success of the implementation is measured by observing whether the algorithm can migrate jobs to the cloud for processing and successfully generate and store results or not. A traditional HPC benchmark is not suitable due to inherent latency in on-premise to on cloud communication. In a traditional HPC, professional HPC nodes are connected via Fiber Optic network. If two similarly specified HPC clusters with one purely on-premise networked via Fiber and another uses a mixture of on-premise and cloud nodes, naturally the on-premise would outperform the HPC+Cloud implementation. If additional HPC resources are only needed on a non-regular basis, therefore some latency affected by transferring data to and from cloud nodes for processing can be tolerated when using the HPC+Cloud cluster. Also, the cost of upgrading physical resources outweighs the cost using resources from the cloud.

The following section is based on the author's work that has been published in the conference paper: "Benchmarking large scale cloud computing in Asia Pacific," 2013 19th IEEE International Conference on Parallel and Distributed System (ICPADS): Curtin University Malaysia campus currently uses Windows HPC software platform for its high-performance computing needs. So, the Public Cloud partner chosen to form the Cloud component of the HPC+Cloud software framework has to integrate well with the existing Microsoft Windows HPC software platform. The platform of choice favored by Curtin Malaysia is the Windows HPC software platform[9]. Therefore, the Public Cloud provider of choice is Microsoft's Azure Cloud[10]. However, there are two other platforms that are commonly used as public cloud providers by academic institutions. Australian institutions utilize the Australian National eResearch Collaboration Tools and Resources Cloud (NeCTAR)[29]. NeCTAR is an Australian Government project to provide public cloud resources to Australian universities. And the other player of choice among both industry and academic institutions is the Amazon Public Cloud or commonly known as Amazon EC2 which is a subsidiary of retail giant Amazon.com[16]. The main reason that the Amazon EC2 is popular with both academia and industry is that Amazon was an early pioneer in providing public cloud services. Compared to both NeCTAR and Amazon Public, Microsoft Azure is a relative newcomer that only started to provide its services in 2012.

These three public clouds, namely Microsoft Azure, Amazon EC2, and NECTAR are benchmarked against one another to determine the best performing public cloud among the three. The software used to benchmark the cloud is Roy Longbottom's Linux benchmarking tools [11].

## 4.2 Cloud Benchmarking Instance Specifications and Assumptions

For cloud computing instances (an instance is a unit of computing resource provided by a cloud provider), the cloud provider provides a fixed computing instance specification meaning there is no way to adjust the CPU option and memory option to ensure parity between the different providers. The specifications as seen in Table 4.1 are all in the medium instance for each provider at the time of running the benchmark.

Table 4.1. Cloud Benchmarking Instance Specifications.

| | **Windows Azure** | **Amazon** | **NECTAR** |
|---|---|---|---|
| **Processor** | AMD Opteron Processor 4171 HE | Intel Xeon@ CPU E5-2650 0 @ 2.OO GHz | Intel@ Core 2 Duo CPU T7700 @ 2.40GHz |
| **Clock Rate** | Minimum 2095 MHz, Maximum 2095 MHz | Minimum 1800 MHz, Maximum 1800 MHz | Minimum 2600 MHz, Maximum 2600 MHz |
| **CPUs** | 2 | 2 | 2 |
| **RAM Size** | 3.36GB | 3.66GB | 7.80GB |

## 4.3 Classic Benchmark Test Categories.

*A. Dhrystone Benchmark*

It is a benchmark[51] used to assess integer processing performance. Table 4.2 shows a comparison between the 3 platforms that the benchmark tests were run on, namely Amazon, NeCTAR, and Azure. The ratings obtained are that of VAX MIPS where VAX stands for Virtual Address Extension and MIPS means Million Instruction per Second.

*B. Linpack Benchmark*

Linpack Benchmark[52] measures the floating-point computing power of a system. Floating point shows a way of representing the approximation of a real number in such a way that it can support a wide range of values. The Millions Floating-point Operations per Second (MFLOPS) is the unit by which the benchmark test is measured.

*C. Livermore Loops*

Livermore loops[53] is a benchmark test that is usually run for parallel computers. Produced for supercomputers consisting of numerous kernels, three specific types of data sizes are run, and the results obtained are in MFLOPS.

The results generated for overall ratings consist of Maximum, Average, Geometric mean (Geomean), Harmonic mean (Harmean) and Minimum, whereby Geomean is the official overall rating. All tests for Livermore loops were completed over 24 loops and the geometric mean was the one recorded.

*D. Whetstone Benchmark*

The Whetstone Single Precision C Benchmark[54] is related to CPU performance and is meant to check speed ratings in Millions of Whetstone Instructions per Second (MWIPS).

Table 4.2. Classic Benchmark Test Results (higher is better).

| | **Windows Azure** | **Amazon** | **NECTAR** |
|---|---|---|---|
| **Dhrystone Benchmark (VAX MIPS rating)** | 8155 | 10455.5 | 10752.28 |
| **Linpack Double Precision Benchmark (MFLOPS)** | 1317.95 | 1603.02 | 1609.31 |
| **Livermore Loops Benchmark Maximum Rating (MFLOPS)** | 2588.9 | 2733.8 | 2634.1 |
| **Whetstone Single Precision C Benchmark MWIPS (MFLOPS)** | 2135.854 | 2111.706 | 2644.834 |

### 4.3.1 Classic Benchmark Overall Comments

Results for the classic benchmark can be viewed in Table 4.2 and Figure 4.1.
In the Dhrystone Benchmark performance, NECTAR scores 1.3x better than Windows Azure. However, between NECTAR and Amazon, NECTAR scores 1.03x better. NECTAR is the best for this benchmark.

In the Linpack Benchmark performance, NECTAR scores 1.22x better than Windows Azure. However, between NECTAR and Amazon, NECTAR scores 1.004x better. NECTAR is the best for this benchmark.

In the Livermore Loops Benchmark performance, Amazon scores 1.056x better than Windows Azure and between Amazon and NECTAR, Amazon scores 1.038x better. Amazon is the best for this benchmark.

In the Whetstone Single Precision C Benchmark performance, NECTAR scores 1.24x better than Windows Azure. However, between NECTAR and Amazon, NECTAR scores 1.25x better. NECTAR is the best for this benchmark.

Overall, NECTAR is the best in the classic benchmarks category followed by Amazon and Windows Azure since NECTAR is superior in 3 benchmark results compared to others.

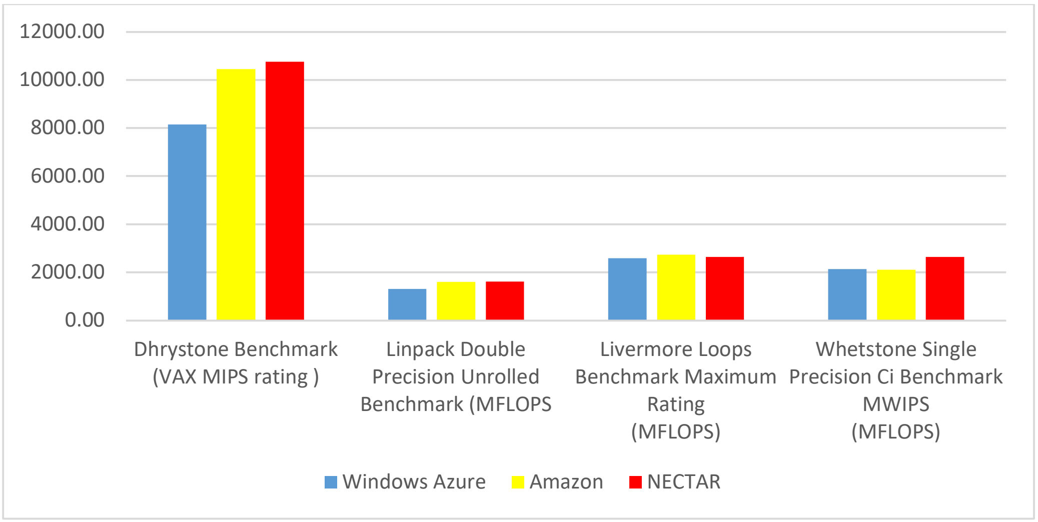


Figure 4.1. Classic Benchmark Results.

### 4.3.2 Disk, USB, and LAN Benchmarks

This test makes use of direct Input-Output (I/O) for the speed of Local Area Network (LAN) to avoid data from being cached in the main memory of the Operating System. Also involved in the benchmark tests are the read/write speed. In this test, a 64Kb file was written, read, and deleted 500 times and the result can be seen in Table 4.3.

Table 4.3. Disk, USB, and LAN Benchmark Test Results

| | **Windows Azure** | **Amazon** | **NECTAR** |
|---|---|---|---|
| **Write MB/sec** | 122.83 | 25.34 | 6.23 |
| **Read MB/sec** | 274.74 | 67.75 | 92.84 |

### 4.3.3 Disk, USB, and LAN Benchmarks Overall Comments

Disk, USB, and LAN performance are critical in determining the processing throughput of a high-performance computing cluster, as no matter how fast the CPU is, final processing times are constrained by I/O operations that are ultimately determined by the read and write speed of the Disc, USB, and Local Area Network (LAN) interfaces of the processing node. Results for the Disk, USB and LAN Benchmark Test results can be viewed in Table 4.3 and Figure 4.2.

In the Disk, USB, and LAN Benchmarks, for the write category, Windows Azure is the best followed by Amazon and NECTAR meanwhile for the read category, Windows Azure is still the best followed by NECTAR and Amazon. Overall, Windows Azure is the best followed by Amazon and NECTAR.

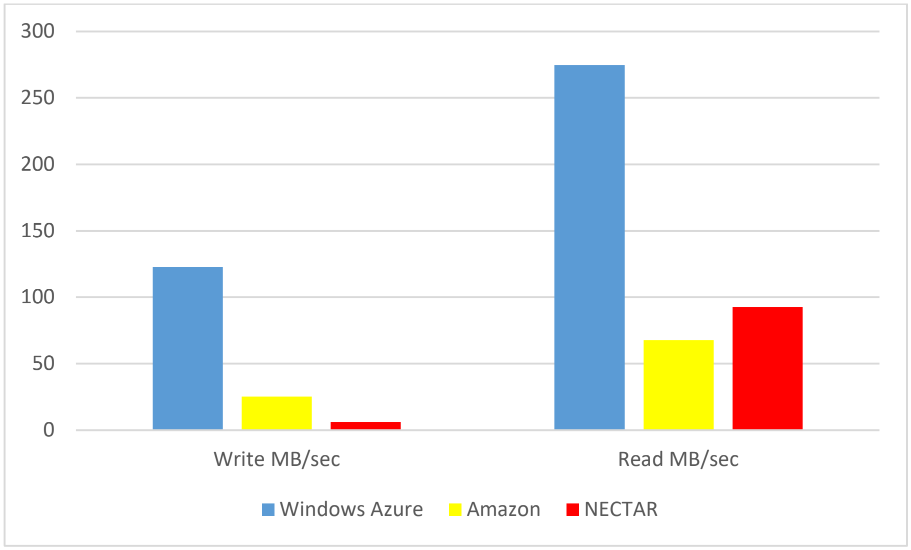


Figure 4.2. Disk, USB and LAN Benchmarks. (higher is better)

## 4.4 Multithreading Benchmarks

Multithreading influences high performance computing as it shows the efficiency at which a high-performance computer manages multiple concurrent processes.

*A. Simple Add Tests*

The tests involved in Simple Add Tests execute 32-bit and 64-bit integer instructions as well as 128-bit SSE floating point. The performance is very relative to the amount of CPU cores available in the system. Since the benchmark test is about multithreading, each thread is given an independent and different adding code to test for each thread. The values taken for this test is the average of two aggregates tested individually.

*B. Whetstone Benchmark*

As opposed to the previous Whetstone benchmark, this test focuses on multithreading applications *OpenMP*. Again, the number of cores present is a determinant factor on the speed of the test run. The results taken as reference for the test are based on the time taken for the last thread to finish and measured in Millions of Whetstone Instructions per Second (MWIPS).

*C. MP SSE (Multi Process Streaming SIMD Extensions) MFLOPS Benchmark*

The purpose of this test is to check for the multiplication of floating-point calculations with data from higher level of caches or from RAM. These programs can be used as a burn-in/reliability test and similar functions can be run on a different segment of data. The last Million Floating Point Instructions per Second (MFLOPS) value from the test is taken as reference.

### *D. MP Memory Speed Tests*

This test makes use of single and double precision floating point numbers and integers to test for the speed of the memory. The average value of the read, write and delete were taken individually and then graphed to figure out the best out of the cloud systems.

### *E. MP Memory Bus Speed Tests*

The bus/memory speed is tested by reading all the data at the same time. The value taken for this test is the ReadAll value of the largest file. This accounts for a sizeable cache and RAM usage stressing the bus and allowing for an estimation of maximum bus/memory speed.

### *F. MP Memory Random Access Speed Benchmark*

This benchmark test is about read and read/write tests that cover cache and RAM data sizes. The largest file (96MB) is chosen since it uses all the resources and maximises the stress on the cores for the test giving a very relatable value. The average of the serial read, read/write, and random read, read/write as well as mutex read, read/write is taken to give a general idea of how it performs on various platforms.

Table 4.4. Multithreading Benchmarks Test Results

| | **Windows Azure** | **Amazon** | **NECTAR** |
|---|---|---|---|
| **Multithreading Simple Add Test (Million Instruction Per Second)** | 13874 | 6343 | 14552.5 |
| **Multithreading Double Precision Whetstones (MWIPS)** | 4257 | 1981 | 5012 |
| **MP SSE MFLOPS Benchmark (MFLOPS)** | 18418 | 10018 | 27839 |
| **MP Memory Speed Test (MB/Second)** | 5902.44 | 6183.44 | 6715.44 |
| **MP Memory Bus Speed (MB/Second)** | 5582 | 7574 | 4961 |
| **RandMemMP Speeds (Memory Random Access Speed Benchmark) (MB/Second)** | 2372.33 | 2608.33 | 2622.67 |

### 4.4.1 Multithreading Benchmark Overall Comments

Results are illustrated in Table 4.4 and Figure 4.3. For multithreading benchmarks, NECTAR is the best in 5 out of 6 categories while Amazon is the best in 1 out of 6 categories. Windows Azure fared badly in all categories. However, in the multithreading add test and multithreading double precision Whetstones, it was close to NECTAR and the overall best performer in multithreading is NECTAR followed by Amazon and Windows Azure.

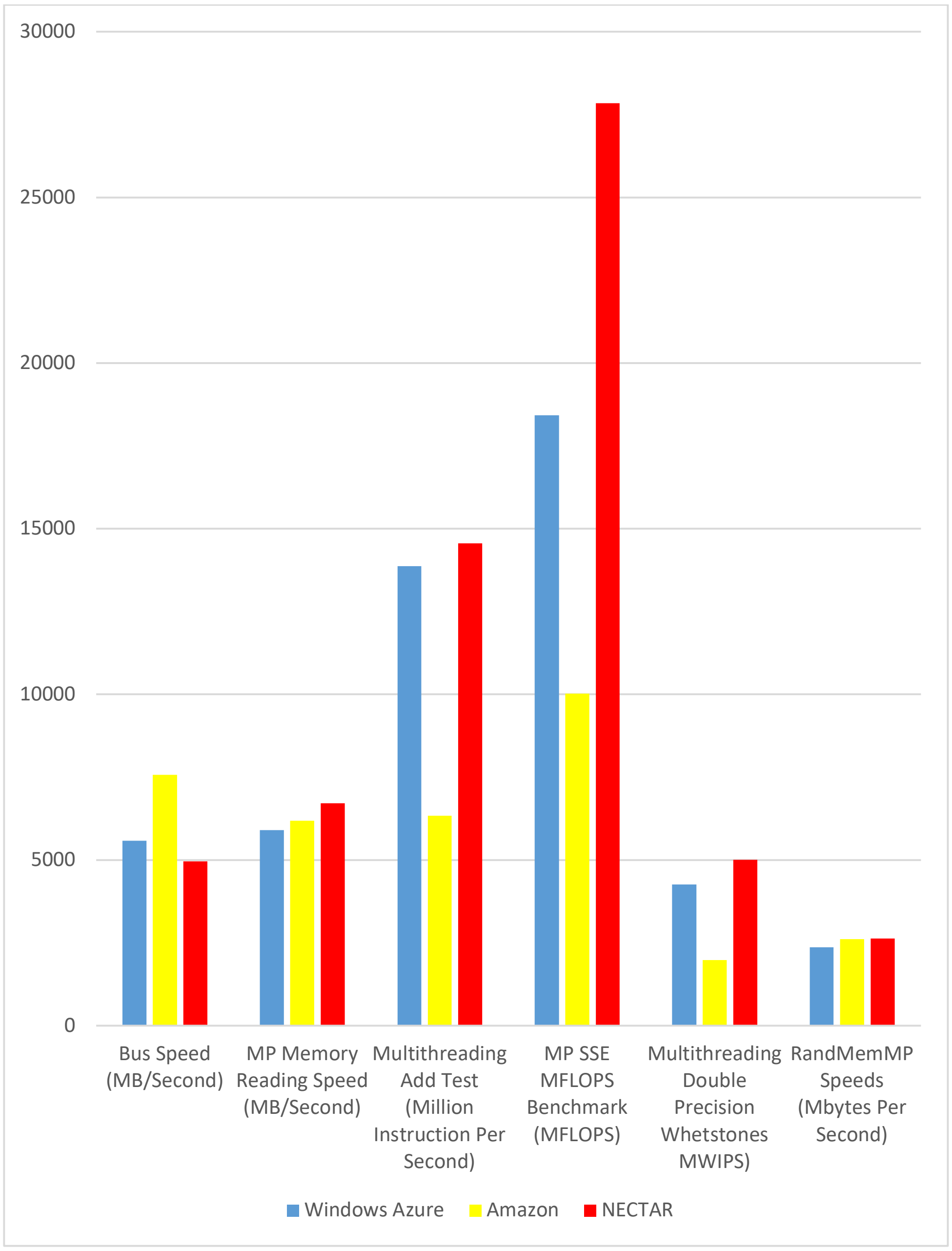


Figure 4.3. Multithreading Benchmarks

## 4.5 OpenMP Benchmarks for Parallel Processing Performance

*A. MemSpeed*

This test makes use of single and double precision floating point numbers and integers to test for the speed of the memory. The average value of the read, write, and delete were taken individually and then graphed to figure out the best out of the cloud systems.

*B. Original OpenMP Benchmark*

Taking the MFLOPS value, this test behaves in a similar way as Windows compilation, meaning the performance gains of the number of cores present is relative to the time taken for the test to complete as compared to a single core. The average value for data in and out is taken as the comparison value for the different platforms.

Table 4.5. OpenMP Benchmarks Test Results

| | **Windows Azure** | **Amazon** | **NECTAR** |
|---|---|---|---|
| **Memory Reading Speed Test (MB/Second)** | 4147.33 | 6457.78 | 4095 |
| **OpenMP MFLOPS Benchmark (MFLOPS)** | 4781 | 10035 | 13886 |

### 4.5.1 OpenMP Benchmarks Overall Comments

OpenMP benchmark results are summarized in Table 4.5 and Figure 4.4. For the OpenMP benchmarks, Amazon is the best in memory reading speed test followed by Windows Azure and NECTAR, while for OpenMP MFLOPS benchmark, NECTAR is the best followed by Amazon and Windows Azure. Overall, NECTAR and Amazon are the best followed by Windows Azure.

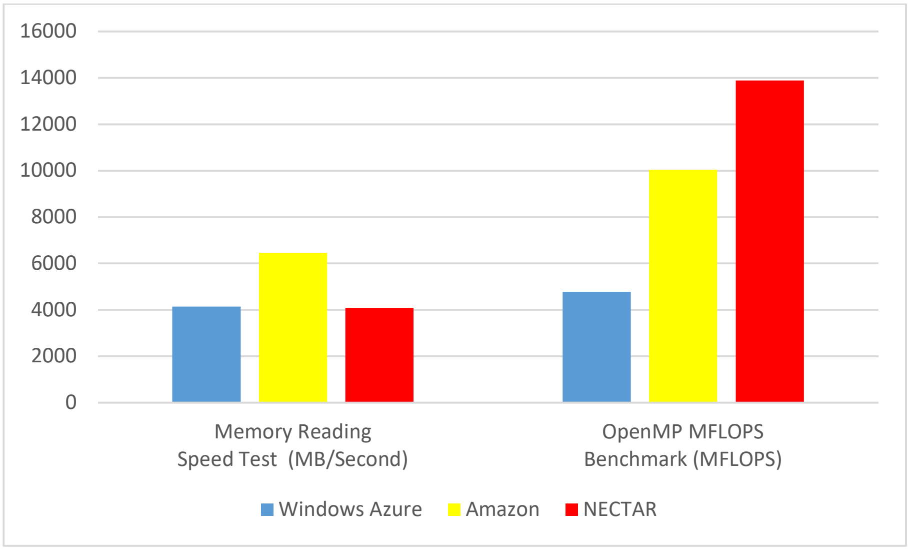


Figure 4.4. OpenMP Benchmarks

### 4.5.2 Memory Bus Speed Benchmark

*A. Bus Speed Test*

This test makes use of single and double precision floating point numbers and integers to test for the speed of the memory. The average value of the read, write and delete were taken individually and then graphed to figure out the best out of the cloud systems.

*B. Random/Serial Memory Test*

This test shows the behaviour of the memory with increasing file size in terms of data transfer. The values taken are similar to that of the MP Memory tests.

*C. SSE And SSE2 Memory Reading Speed Test*

This variation of the SSE (Streaming Single Instruction Multiple Data Extensions) benchmark measures Single Precision and Double Precision, floating point speeds, data streaming from caches and RAM. The alterations in this test avoid intermediate register to register operations to produce much faster speeds. Again, the largest value is taken as reference and compared across platforms.

Table 4.6. Memory BusSpeed Benchmark results (higher is better)

| | **Windows Azure** | **Amazon** | **NECTAR** |
|---|---|---|---|
| **Bus Speed Test (MB/Second)** | 5455 | 7461 | 2525 |
| **Random/Serial Memory Test (MB/Second)** | 1850.25 | 3113.625 | 1910.375 |
| **SSE & SSE2 Memory Reading Speed Test (MFLOPS)** | 4935.25 | 4267.25 | 5956.125 |

### 4.5.3 BusSpeed Benchmark Overall Comments

BusSpeed benchmark results can be examined in Table 4.6 and Figure 4.5. In BusSpeed benchmark, Amazon is the best in bus speed test followed by Windows Azure and NECTAR, while in random/serial memory test, Amazon is still the best followed by NECTAR and Windows Azure. For SSE & SSE2 memory reading speed test, NECTAR is the best followed by Windows Azure and Amazon. Overall, in this BusSpeed benchmark, Amazon is the best followed by NECTAR and Windows Azure.

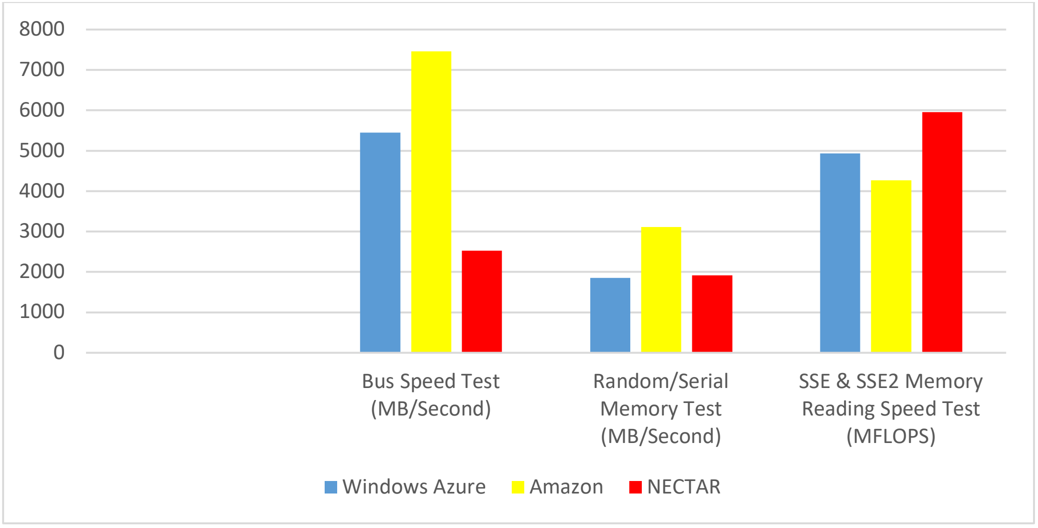


Figure 4.5. Memory BusSpeed Benchmark results

## 4.6 Benchmarking Overall Results and Conclusion

Table 4.7. Overall results (higher is better)

| | **Windows Azure** | **Amazon** | **NECTAR** |
|---|---|---|---|
| **Classic Benchmarks for CPU Performance** | Third | Second | First |
| **Disk, USB, and LAN Benchmarks** | First | Second | Third |
| **Multithreading Benchmarks** | Third | Second | First |
| **OpenMP Benchmark for Parallel Processing Performance** | Third | Second | First |
| **Memory BusSpeed Benchmark** | Third | First | Second |

### 4.6.1 Conclusion of Benchmarking Public Cloud Providers

The main aim of this exercise was to identify the best possible candidate for building an HPC+Cloud high performance cluster that can be deployed quickly and easily. Windows Azure came in last in almost all categories. However, given that HPC+Cloud requires high I/O throughput between on-premise HPC which is currently in the organization and the public cloud, the high I/O throughput is necessary to avoid data bottlenecks between the HPC and the Cloud.

Therefore, Windows Azure, despite coming in the last place in all categories except Disk USB and LAN benchmarks, still is the prime candidate for deployment for the HPC+Cloud in Curtin Malaysia. Also noted was that many academic institutions that are already using Windows HPC will find easier integration with Windows Azure than any other public cloud.  This is particularly true as Curtin University Malaysia's current on-premise HPC is built on Windows HPC[9] and the choice of Windows Azure will facilitate rapid deployment of existing software and hardware resources to be quickly combined with the Cloud using the HPC+Cloud software framework.

## 4.7 Benchmarking HPC On-Premise vs HPC+Cloud vs HPC on Cloud

### 4.7.1 The main purpose of Benchmarking

In this section, benchmarking was conducted to gauge the performance of the HPC+Cloud cluster against HPC on-premise and HPC on Cloud cluster. To understand the purpose of the benchmarking exercise, the objective of building the HPC+Cloud infrastructure must be known. The objective is to test the ability of an on-premise HPC cluster to use resources on the Cloud using the proposed HPC+Cloud framework. If the HPC+Cloud cluster configuration produces results when benchmarked and the results are comparable to existing HPC on-premise and HPC on Cloud cluster configurations, then it means that HPC+Cloud cluster configuration has the ability to sustain and complete High Performance Computing loads[55]. HPC+Cloud is an infant framework that is new, therefore it is important to prove that it is functional. These benchmarks will prove that the HPC+Cloud framework is functional and able to deliver results.

It is important to know that the purpose of these benchmarks is not to set up or expect the HPC+Cloud to be the best performing cluster configuration. By its nature, HPC+Cloud hardware infrastructure is heterogeneous due to the lack of uniformity between node configurations. Therefore, the HPC on Cloud is predicted to be the most likely best performing cluster configuration due to its hardware homogeneity, and the fact that the hardware software infrastructure is managed entirely by the public cloud provider Microsoft Azure. Hence, it benefits from hardware vendor optimizations that are exclusive to Microsoft.

### 4.7.2 Experimental Setup and Assumptions.

Before starting to benchmark the performance of an HPC+Cloud cluster, first some preliminary testing must be completed to ensure basic functionality of HPC+Cloud cluster is functional and therefore ready to have its performance benchmarked against HPC On Premise and HPC on Cloud. The test results are observed in table 4.8:

Table 4.8 HPC+Cloud framework functionality test results

| **Test Criteria** | **Test Result** |
|---|---|
| On Premise Cluster Links to Cloud Nodes | Link is successful. HPC Nodes from cloud to on-premise nodes can ping each other. |
| On-premise cluster can transfer jobs to on cloud nodes | Transfer is successful. It Is observed that time to transfer is job data dependent. Jobs that have higher input data take longer to transfer. |
| HPC jobs filtering for jobs with privacy flag | Filter functions appropriately. The privacy flag worked to make sure jobs that users have designated suitable only for processing on-premise do not migrate. |
| HPC+Cloud framework can deallocate cloud nodes upon successful job completion | Deallocation of cloud nodes is successful. |

In the previous section, the various public cloud solutions were benchmarked, and it was determined that Microsoft Azure will be used as the public cloud provider. Microsoft Azure provides a fixed computing instance specification for cloud computing instances (a cloud computing instance is a unit of computing resource provided by a cloud provider). Therefore, it is quite difficult to have the exact same CPU configuration between on-premise HPC cluster and Microsoft Azure HPC cluster instance. To ensure parity, all CPUs are made to be the same class, that is Xeon based system from the same generation with a turbo clock speed of 4.8 GHz and a maximum single-core base frequency of 3.8 GHz. As for the on Cloud HPC instance, Azure Instance Standard_F4s_v2 is used. However, in a Microsoft Azure HPC cluster, instances by nature are shared hardware because they belong to the public cloud provider shared pool of cloud infrastructure. It is the nature of cloud computing where all CPU, RAM, and storage resources are virtually pooled together. The basic cluster configuration for HPC+ Cloud are one head node and two compute nodes. One compute node is on-premise with two CPUs and another compute node is on the Azure Cloud with Standard_F4s_v2 Cloud instances that provides 4 Virtual CPU cores. There is a VPN connection between the HPC on-premise and HPC nodes on the cloud.

The on-premise cluster configuration consists of a single cluster with one head node and two compute nodes with a total of 8 Virtual CPU cores. The On Cloud HPC cluster configuration consists of a single cluster with 8 CPUs consisting of one head node and two compute nodes with four Virtual CPUs each. The On Cloud HPC cluster is a control benchmark to observe how the benchmark will run if purely executed on Cloud. Head nodes for all three setups are the central control nodes and not used in processing the problem. Benchmarking is carried out via Roy Longbottom's Windows benchmarking tools and Microsoft Lizard. All network, storage, and RAM configuration are controlled to be as close a like as possible using current Microsoft Azure hardware configuration information. Table 4.9 summarizes the experimental setup for benchmarking the various types of cluster configurations.

Table 4.9. Benchmarking Instance Specifications.

| | **HPC on-premise** | **HPC+Cloud (a combination of on-premise and Cloud nodes)** | **HPC on Cloud** |
|---|---|---|---|
| **Processor type** | 2x compute Nodes with Intel® Xeon® E-2244G | 1x Azure Instance Standard_F4s_v2 (Microsoft Azure rated 4 virtual CPU cores) | 2x Azure Instance Standard_F4s_v2 (Microsoft Azure rated 8 virtual CPU cores) |

| | | | |
|---|---|---|---|
| | | And<br>1x Premise Compute Node (Intel® Xeon® E-2244G) | |
| **CPU speeds** | Turbo clock speed of 4.8 GHz and a maximum single-core base frequency of 3.8 GHz | | |
| **Total Number of Virtual CPUs** | 8 | 8 | 8 |
| **RAM(GB)** | 8 | 8 | 8 |
| **Network Bandwidth (Mbps) per Node** | 1750 | 1750 | 1750 |
| **Hard drive Space (TB)** | 2 | 2 | 2 |

## 4.8 Classic Benchmark Test Categories

*A. Dhrystone Benchmark*

It is a benchmark used to assess integer processing performance. Table 4.10 shows a comparison between the cluster configurations. The ratings obtained are that of VAX MIPS where VAX stands for Virtual Address Extension and MIPS means Million Instruction per Second.

*B. Linpack Benchmark*

It is a benchmark used to assess the floating-point processing power of a system. The Millions Floating-point Operations per Second (MFLOPS) is the unit by which the benchmark test is measured.

*C. Livermore Loops*

Livermore loops is a benchmark for parallel processing. 24 kernels were run three times at varying Do-loop spans to create short, medium, and long vector performance measures and the results obtained are in MFLOPS. Maximum, Average, Geometric mean (Geomean), Harmonic mean (Harmean), and Minimum are the findings generated for overall ratings, with the Geomean being the official overall rating.

*D. Whetstone Benchmark*

The Whetstone benchmark, measured in Millions of Whetstone Instructions per Second (MWIPS) is used for evaluating the performance of scientific applications. It has several modules aimed to represent a variety of operations that are commonly used in scientific applications. A combination of Integer and floating-point math operations, array accesses, conditional branches, and procedure calls are all employed, together with C specific functions like sin, cos, sqrt, exp, and log.

Table 4.10. Classic Benchmark Test Results (higher is better).

| | **HPC on-premise** | **HPC+Cloud** | **HPC on Cloud** |
|---|---|---|---|
| **Dhrystone Benchmark (VAX MIPS)** | 59887.5 | 60478.6 | 61979.33 |
| **Linpack Benchmark (MFLOPS)** | 19657 | 21347 | 22856 |
| **Livermore Loops (MFLOPS)** | 15567.6 | 16402.8 | 17509.4 |
| **Whetstone Benchmark (MWIPS)** | 14789 | 15869 | 16566 |

### 4.8.1 Classic Benchmark Overall Comments

Results for the classic benchmark can be viewed in Table 4.10 and Figure 4.6. In the Dhrystone Benchmark performance, HPC on Cloud has the best performance with 61979.33 (VAX MIPS) and an average 2% performance boost compared to the other two categories. Second is HPC+Cloud with 60478.6 (VAX MIPS), followed by HPC on-premise 59887.5 (VAX MIPS).

In the Linpack Benchmark performance, HPC on Cloud has the best performance with 22856 (MFLOPS) and an average 8% performance boost compared to the other two categories. Second is HPC+Cloud with 21347 (MFLOPS), followed by HPC on-premise with 19657 (MFLOPS).

In the Livermore Loops Benchmark performance, HPC on Cloud has the best performance with 17509.4 (MFLOPS) and an average 6% performance boost compared to the other two categories. Second is HPC+Cloud with 16402.86 (MFLOPS), followed by HPC on-premise with 15567.6 (MFLOPS).

In the Whetstone Benchmark performance, HPC on Cloud has the best performance with 16566 (MWIPS) and an average 6.5% performance boost compared to the other two categories. Second is HPC+Cloud with 15869 (MWIPS), followed by HPC on-premise with 14789 (MWIPS).

The performances of HPC on Cloud, HPC+Cloud and HPC on-premise are very close to each other given that the CPU configurations are similar using the same CPU models with the same turbo and base clock frequencies. However, HPC on Cloud configuration emerges ahead in each benchmark, with an average performance boost of 6% against the two other categories. This is most likely because HPC on Cloud system hardware and firmware is highly optimised by the CPU vendor Intel for use on the Microsoft Azure cloud. On-premise clusters are not typically optimised as they are COTS (completely off the shelf) hardware, meanwhile HPC+Cloud has a mixture of optimised cloud infrastructure and on-premises COTS hardware. Overall HPC on Cloud cluster configuration performs the best in Classic Benchmark test category.

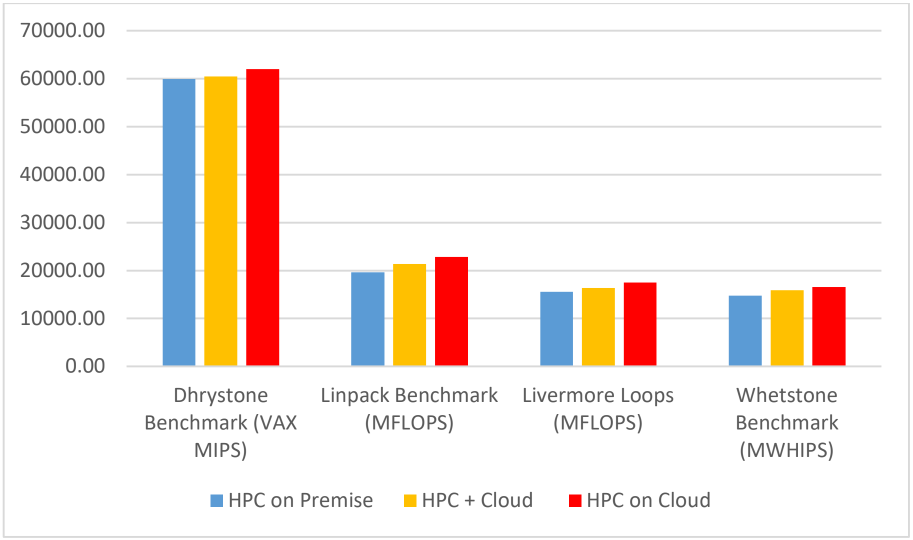


Figure 4.6. Classic Benchmark Results (higher is better).

## 4.9 I/O Operations Benchmark

This benchmark tests the Read and Write Throughput Input Output Operations on the compute nodes. To test the actual read write speed, the data is randomly varied to avoid it being cached in the main memory of the Operating System. Randomly varied versions of a 64Kb file were written, read, and deleted 500 times and the result can be seen in Table 4.11.

Table 4.11. I/O Operations Benchmark Test Results (higher is better).

| | **HPC on-premise** | **HPC+Cloud** | **HPC on Cloud** |
|---|---|---|---|
| **Write MB/sec** | 189.6 | 257.9 | 308.07 |
| **Read MB/sec** | 489.7 | 567.7 | 686.86 |

### 4.9.1 I/O Operations Benchmark Overall Comments

I/O Operations Benchmark determines how fast unprocessed data can be read from the network storage cluster and how fast processed data can be stored to the network storage cluster, therefore this benchmark also reflects on the network throughput performance of the cluster. I/O Operations are a potential bottleneck on a High-Performance Computing cluster. Results for the I/O Operations Benchmark can be viewed in Table 4.11 and Figure 4.7.

HPC on Cloud has the best performance in I/O operations benchmark with a significant average 36% performance boost compared to the other two categories. Second is HPC+Cloud with another significant average performance boost of 26% against third place HPC on-premise.

Public cloud systems deploy highly optimised and high throughput mass storage networks to manage their data storage for use by the HPC cluster and provide service level guarantees for I/O performance. Therefore, the dominance of the HPC on Cloud is clearly seen, it is the best in this benchmark, followed by HPC+Cloud, which partially benefits from the advantages of a mass cloud storage system. HPC on-premise does not have the benefits of storage vendor optimisations and access to high throughput mass storage networks provided by the cloud provider. Despite this, one advantage of HPC on-premise against the other configurations is that storage of data in an HPC on-premise is closer to the cluster, typically within the cluster LAN and not remotely stored, therefore the performance disparity between HPC on-premise and others is not as high as it should be.

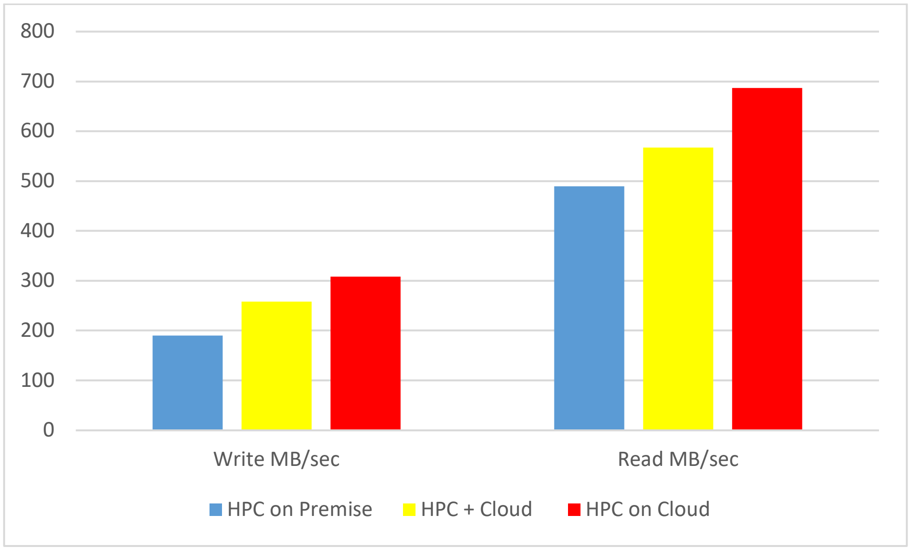


Figure 4.7. I/O Operations Benchmark. (higher is better)

## 4.10 Multithreading Benchmarks

Multithreading has an impact on high-performance computing because it demonstrates how effectively each independent node in the cluster can manage several multiple concurrent processing operations.

*A. Simple Add Tests*

Simple Add Tests run 32-bit and 64-bit integer instructions as well as 128-bit SSE floating point instructions. The performance is highly dependent on the number of CPU cores in the system. Because the benchmark test is focused on multithreading, each thread is given its own code to test. The data used in this test are the average of two aggregates that were tested separately.

*B. Whetstone Benchmark*

This test, unlike the preceding Whetstone benchmark, concentrates on multithreading applications. The number of cores present is a determining element in the test run's speed once again. The test results are based on the time it takes the last thread to finish, which is measured in Millions of Whetstone Instructions per Second (MWIPS).

### C. MFLOPS Program

The goal of this test is to see if floating point calculations are being multiplied with data from higher levels of caches or RAM. Similar procedures can be conducted on a different section of data as a burn-in/reliability test with these programmes. The test's most recent Million Floating Point Instructions per Second (MFLOPS) figure is used as a benchmark.

### D. Memory Speed Tests

To measure the memory's speed, this test uses single and double precision floating point numbers as well as integers. The average value of the read, write, and delete operations were taken separately and then graphed to determine the optimal figure.

### E. Memory Bus Speed Tests

Reading all data at the same time is used to measure the bus/memory speed. The ReadAll value of the largest file was used for this test. This accounts for a large cache and RAM consumption, which puts the bus under stress and allows for an estimate of the bus/memory speed.

### F. Memory Random Access Speed Benchmark

This benchmark test covers cache and RAM data sizes and includes read and read/write testing. The largest file (96MB) was chosen since it utilises all of the resources and puts the most stress on the cores during the test, resulting in a very relevant result. To get a rough picture of how it performs on different systems, the average of serial read, read/write, random read, read/write, and mutex read, read/write is taken.

Table 4.12. Multithreading Benchmarks Test Results (higher is better).

| | HPC on-premise | HPC+Cloud | HPC on Cloud |
|---|---|---|---|
| **Multithreading Add Test (Million Instruction Per Second)** | 57998.6 | 58955.7 | 60392.8 |
| **Multithreading Double Precision Whetstones (MWIPS)** | 19455.7 | 20766.6 | 21050.4 |
| **MFLOPS Program (MFLOPS)** | 98056.6 | 100578.5 | 116923.8 |
| **Memory Speed (MB/Second)** | 25776.5 | 26455.6 | 27869.1 |
| **Memory Bus Speed (MB/Second)** | 29456.6 | 30456.4 | 31432.1 |
| **Memory Random Access Speed (MB/Second)** | 8997.4 | 9345.5 | 10823.2 |

### 4.10.1 Multithreading Benchmark Overall Comments

Results are shown in Table 4.12 and Figure 4.8. For multithreading benchmarks, HPC on Cloud is clearly the best performing with an average performance boost of 9% over the other two categories. And HPC+Cloud coming in second place and HPC on-premise being third. Between second placed HPC+Cloud and third placed HPC on-premise, on average HPC+Cloud sees a 4% average performance boost over third placed HP on-premise. Multithreading Benchmarks results are close as all 3 HPC cluster configuration CPUs are of the same class that is Xeon based system from the same generation. HPC on Cloud has the upper hand as Public Cloud Providers typically optimise their CPU configuration for High Performance Computing processing.

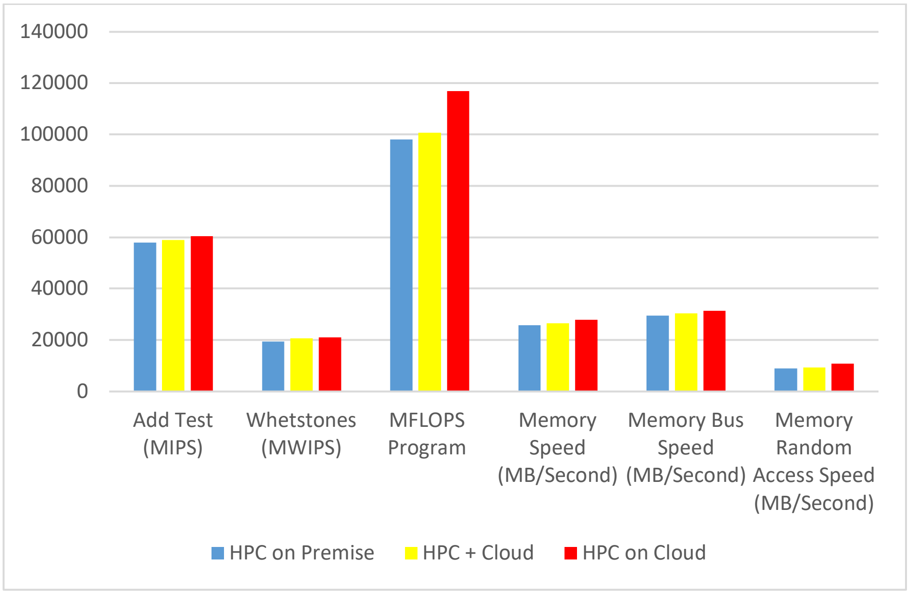


Figure 4.8. Multithreading Benchmarks (higher is better)

### 4.10.2 OpenMP Benchmarks for Parallel Processing Performance

OpenMP utilizes a shared memory paradigm when doing parallel processing, hence memory speed and CPU processing power is important to the performance of a cluster.

*A. MemSpeed- Memory Reading Speed Test*

To measure the memory's speed, this test uses single and double precision floating point numbers as well as integers. The average value of the read, write, and delete operations were taken separately and then graphed to determine the optimal figure.

*B. Original OpenMP Benchmark*

When more than one node is available on a cluster, OpenMP is a system-independent collection of methods and software that allows for automatic parallel processing of shared memory data. OpenMP is a vectorizing compiler that is similar to the High Performance Linpack Benchmark.

Table 4.13. OpenMP Benchmarks Test Results (higher is better).

| | **HPC on-premise** | **HPC+Cloud** | **HPC on Cloud** |
|---|---|---|---|
| **MemSpeed Memory Reading Speed Test (MB/Second)** | 23456.5 | 25566.5 | 26799.7 |
| **OpenMP MFLOPS Benchmark (MFLOPS)** | 55345.5 | 56345.5 | 57543.9 |

### 4.10.3 OpenMP Benchmarks Overall Comments

OpenMP benchmark results are summarized in Table 4.13 and Figure 4.9. For the OpenMP benchmarks Memory Speed Test and OpenMP MFLOPS test, HPC on Cloud comes out ahead with an average performance boost of 6% over the other two categories. The results are somewhat close as the average performance boost of second placed HPC+Cloud over third placed HPC on-premise is 5.5%.

Microsoft Azure hardware optimisation for High Performance Computing allows the HPC on Cloud to edge out HPC+Cloud and HPC on-premise cluster configurations. OpenMP parallel processing paradigm takes advantage of the optimization carried out in the virtual shared memory architecture at the hypervisor level in HPC on Cloud cluster configuration.

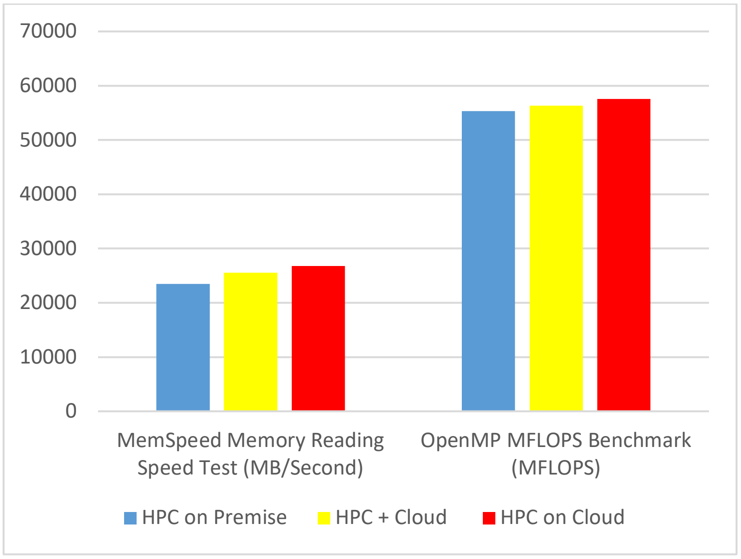


Figure 4.9. OpenMP Benchmarks (higher is better)

## 4.11 High Performance Linpack Benchmark via Windows Lizard

A version of Linpack that was optimised for multicore single node processing was used in the preceding Linpack benchmark in Section 4.3. However now the high-performance version of Linpack[56] is going to be used to evaluate the performance of high-performance clusters made up of many compute nodes with multicore CPUs. High performance Linpack was chosen since it is widely used and performance data for many public clusters are readily available. The Linpack Benchmark involves solving a dense system of linear equations. The performance of a cluster for solving a dense system of linear equations is reflected by the Linpack Benchmark score. Due to the regularity of the problem, the attained performance is relatively high, and the performance numbers provide a solid estimate of peak performance of a High-Performance cluster. Linpack Wizard or Lizard was specially developed, which is based on a canonical library wrapped in a convenient visual wizard (supplied with the HPC Tool Pack 2012). This wizard allows an express test with standard parameters automatically selected by the wizard.

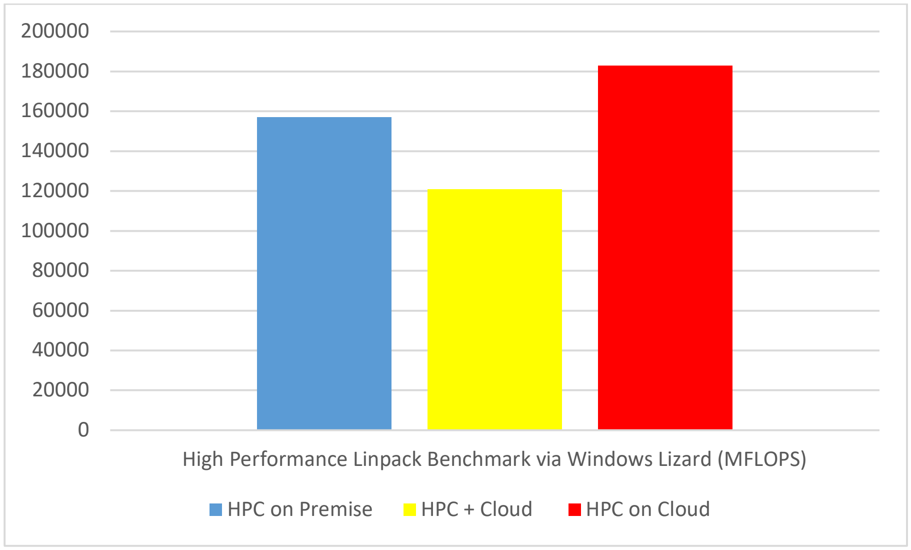


Figure 4.10. High Performance Linpack Benchmark (higher is better)

### 4.11.1 High Performance Linpack Benchmark Overall Comments

High Performance Linpack Benchmark results can be examined in Table 4.14 and Figure 4.10. HPC on Cloud comes out on top again with 182848 (MFLOPS) with a significant average 34% performance boost over the other two categories. But there is a reversal between HPC on-premise and HPC+Cloud. For the first time in these benchmarks, the HPC on-premise comes in second place with 157256 (MFLOPS). HPC+Cloud comes out in a distant third with 120776 (MFLOPS) showing a 23% performance drop against second placed HPC on-premise. Once again, Microsoft Azure Public cloud optimization in the internode communication helps the HPC on Cloud come out on top.

Linpack by its nature uses a non-shared memory model of parallel processing as it depends on the MPI (Message Passing Interface) parallel processing paradigm, where each node has its own memory independently and hence, communication between nodes is important. The main factor causing the HPC+Cloud to have a slower processing speed as compared to the other two cluster configurations is the time taken to transfer HPC jobs to cloud from the on-premise nodes. Internode communication happens across networks via Virtual Private Network (VPN) from the on-premise compute nodes to the on cloud compute nodes. Compare this to HPC on premise cluster and HPC on Cloud cluster, both have internode communication happening locally within the cluster.

Table 4.14. High Performance Linpack Benchmark (higher is better)

| | HPC on-premise | HPC+Cloud | HPC on Cloud |
|---|---|---|---|
| High Performance Linpack Benchmark via Windows Lizard (MFLOPS) | 157256 | 120776 | 182848 |

### 4.11.2 Summary of Benchmarking Overall Results

Table 4.15. Benchmarking Overall Results

| | **HPC on-premise** | **HPC+Cloud** | **HPC on Cloud** |
|---|---|---|---|
| **Classic Benchmarks for CPU Performance** | Third | Second | First |
| **Multithreading Benchmarks** | Third | Second | First |
| **OpenMP Benchmarks for Parallel Processing Performance** | Third | Second | First |
| **High Performance Linpack Benchmark** | Second | Third | First |

Table 4.15 shows that obviously HPC on Cloud emerges as the clear winner in every benchmark. HPC on Cloud cluster configuration performs the best in Classic Benchmark test category. Classic Benchmark emphasises CPU performance, especially at the compute node level. The results are close to each other as all three cluster configurations use similar CPU class. HPC on Cloud system hardware and firmware is highly optimised by the CPU vendor Intel for use on the Microsoft Azure cloud. On-premise clusters are not typically optimised as they are COTS (completely

off the shelf) hardware, meanwhile HPC+Cloud has a mixture of optimised cloud infrastructure and on-premises COTS hardware.

Multithreading benchmarks the cluster CPUs ability to handle processing threads. HPC on Cloud is clearly the best performing, with HPC+Cloud in second place and HPC on-premise being third. All three HPC cluster configuration CPUs are of the same class, that is Xeon based system from the same generation. Public Cloud Providers typically optimise their CPU configuration for High Performance Computing processing. Therefore, once again HPC on Cloud has the upper hand.

The next two categories, OpenMP Benchmarks and High Performance Linpack are specifically geared to benchmarking cluster performance. HPC on Cloud is the best in OpenMP Benchmarks, followed by HPC+Cloud and HPC on-premise. OpenMP utilizes a shared memory paradigm, where separate parallel processes in the cluster share a common virtual memory space. HPC on Cloud takes advantage of the optimization carried out in the virtual shared memory architecture at the hypervisor level in HPC on Cloud cluster configuration. What this means is that because RAM resources are pooled together in public cloud infrastructure, OpenMP processing takes advantage of it naturally as OpenMP is a shared memory paradigm parallel processing.

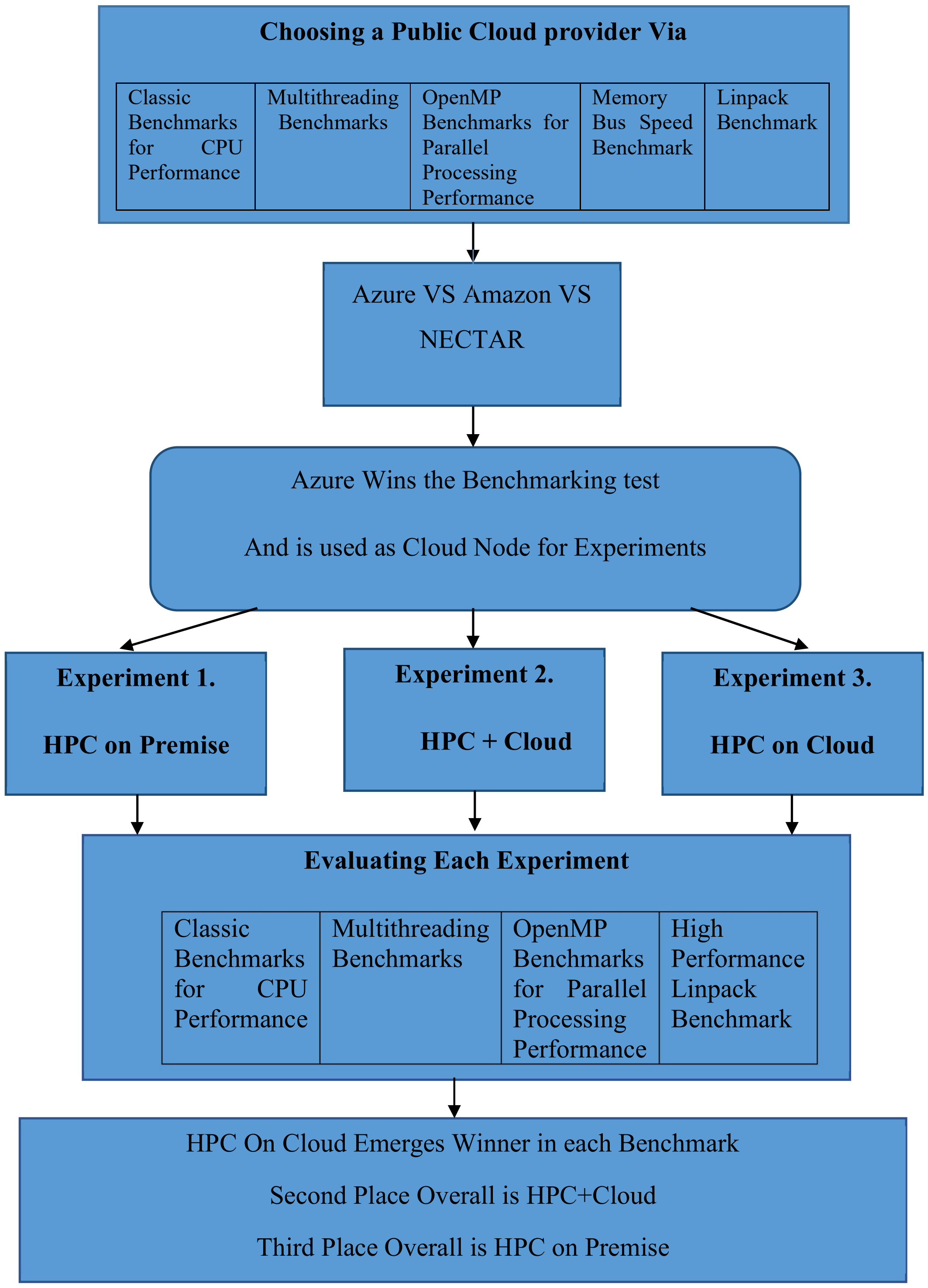
Meanwhile in High Performance Linpack benchmark, HPC on Cloud with 182848 (MFLOPS) takes the lead once more. However, there is a significant reversal. For the first time in these benchmarks, HPC on-premise places second with 157256 (MFLOPS), while HPC+Cloud places third with 120776 (MFLOPS) for the first time. High Performance Linpack is built on the MPI (Message Passing Interface) which uses a non-shared memory parallel processing paradigm, which necessitates communication between nodes. Because it takes time to transfer HPC jobs from on-premise computing nodes to cloud compute nodes, therefore HPC+Cloud has a slower processing performance than the other two cluster setups. A Virtual Private Network (VPN) connects on-premise compute nodes to cloud compute nodes, allowing internode communication across networks. Internode communication, on the other hand, takes place locally within the cluster in both HPC on-premise and HPC on Cloud clusters.

## 4.12 Significance of The Benchmarking Results of HPC+Cloud Cluster

The benchmarking exercise met its goal of observing how well an on-premise HPC cluster can utilise Cloud resources using HPC+Cloud framework. The HPC+Cloud cluster combination gives similar results to existing HPC on-premise and HPC on Cloud cluster setups when benchmarked. As a result, the benchmarks have demonstrated that the HPC+Cloud cluster architecture can support, sustain, and complete High Performance Computing workloads successfully. This validation is important as the HPC+Cloud framework is a new framework. The benchmarking exercise also exposed a weakness of HPC+Cloud cluster infrastructure as it requires internode communication across networks that potentially can be a bottleneck to the HPC+Cloud cluster processing performance. This bottleneck will become a significant issue to the performance of HPC+Cloud cluster if the number of nodes increase and potentially more internode communications occur between on-premise compute and on-premise cloud nodes.

## 4.13 Summary of the Chapter

In this chapter, three public cloud providers: Amazon, Microsoft Azure and NeCTAR were benchmarked. It was determined that Microsoft Azure was the best fit for HPC+Cloud software framework. The next step was that the HPC+Cloud cluster was implemented and benchmarked against an HPC on-premise cluster and an HPC on Cloud cluster. Figure 4.11 presents a summary of the achievements that were concluded in Chapter 4. In this chapter, by conducting these HPC benchmarks, the goal of demonstrating that the HPC+Cloud cluster can support, sustain, and complete High Performance Computing workloads successfully was achieved.



Choosing a Public Cloud provider Via
Classic Benchmarks for CPU Performance
Multithreading Benchmarks
OpenMP Benchmarks for Parallel Processing Performance
Memory Bus Speed Benchmark
Linpack Benchmark
Azure VS Amazon VS NECTAR
Azure Wins the Benchmarking test
And is used as Cloud Node for Experiments
Experiment 1.
HPC on Premise
Experiment 2.
HPC + Cloud
Experiment 3.
HPC on Cloud
Evaluating Each Experiment
Classic Benchmarks for CPU Performance
Multithreading Benchmarks
OpenMP Benchmarks for Parallel Processing Performance
High Performance Linpack Benchmark
HPC On Cloud Emerges Winner in each Benchmark
Second Place Overall is HPC+Cloud
Third Place Overall is HPC on Premise

Figure 4.11. Summary of Chapter Four

# 5 Conclusions and Future Work

## 5.1 Conclusions

The HPC Cloud software framework enables on-premise HPC systems to scale up without making costly hardware investments. Organizations with existing HPC clusters will be able to scale and upgrade their clusters using the HPC+Cloud framework at a lower cost than alternative HPC implementation architectures such as the HPC cluster implemented on grids, HPC cluster implemented on the Cloud and HPC cluster implemented on the Hybrid Cloud, all which were discussed in Chapter two of the thesis. When using external computing resources from the public cloud, privacy and legal concerns are an issue. However, the proposed HPC+Cloud framework handles this issue by filtering HPC jobs and ensuring privacy and legal compliance.

In this research, three public cloud providers: Amazon, Microsoft Azure and NeCTAR were benchmarked. It was determined that Azure was the best fit for the proposed HPC Cloud software framework. The HPC+Cloud cluster was implemented and benchmarked against HPC on-premise cluster and HPC on Cloud cluster. By conducting these HPC benchmarks, the goal of demonstrating that the proposed HPC+Cloud cluster is functional and can support, sustain, and complete High Performance Computing workloads successfully was achieved.

## 5.2 Summary of Objectives Accomplished

Alternate approaches to scale HPC implementation architectures such as the HPC cluster implemented on grids, HPC cluster implemented on the Cloud and HPC cluster implemented on the Hybrid Cloud were investigated and compared against the proposed HPC+Cloud framework. The advantages and disadvantages of each implementation architectures were discussed in Chapter 2.

In Chapter 3, the proposed HPC+Cloud framework was modelled using flowcharts and pseudocode and then, the framework was implemented. The framework implemented a software flag to manage privacy and legal concerns to ensure only HPC jobs permitted by the administrator were allowed to migrate externally to the cloud. Using the privacy flag variable, the HPC+Cloud framework filters HPC jobs and does not allow those jobs to migrate to the on cloud nodes hence ensuring privacy and legal compliance.

Next, the Microsoft Azure public cloud, Amazon and NeCTAR were benchmarked to determine the best public cloud provider to combine with the proposed HPC+Cloud software framework architecture. As a result, Microsoft Azure was selected as the Public Cloud provider in Chapter 4.

The HPC+Cloud cluster was built on Microsoft Azure public cloud. The HPC+Cloud cluster worked seamlessly for scheduling jobs on the integrated public cloud using the HPC+Cloud framework. To validate the implemented HPC+Cloud software framework, its functionality was further benchmarked in chapter 4.

The benchmarking exercise met its goal of observing how well an on-premise HPC cluster can utilize Cloud resources using the proposed HPC+Cloud framework. The HPC+Cloud cluster gave similar results to existing HPC on-premise and HPC on Cloud cluster setups when benchmarked. The benchmarks in section 4.7 have demonstrated that the proposed HPC+Cloud cluster architecture can support, sustain, and complete High Performance Computing workloads successfully. This validation is important as the proposed HPC+Cloud framework is a new framework.

The benchmarking exercise also exposed a weakness of HPC+Cloud cluster infrastructure as it requires internode communication across networks from the organization on-premise to the cloud. This potentially can be a bottleneck to the HPC+Cloud cluster processing performance. This bottleneck will become a significant issue to the performance of HPC+Cloud cluster if the number of nodes increases and potentially more internode communications occur between on-premise compute nodes and on cloud nodes.

## 5.3 Future Work

One suggested future direction for this research is to do further benchmarking on the internode communications that occur between on-premise compute nodes and on cloud nodes. The purposes are to investigate further on its effects on the efficiency of the proposed HPC+Cloud framework, and to investigate ways to overcome the communications bottlenecks between the on-premise HPC nodes and on cloud HPC nodes.

# References

[1] P. Mell and T. Grance, "The NIST-National Institute of Standars and Technology- Definition of Cloud Computing," *NIST Spec. Publ. 800-145*, p. 7, 2011.

[2] K. R. Sajay and S. S. Babu, "A study of cloud computing environments for High Performance applications," *Proc. 2016 Int. Conf. Data Min. Adv. Comput. SAPIENCE 2016*, pp. 353–359, 2016, doi: 10.1109/SAPIENCE.2016.7684127.

[3] A. Calatrava, G. Molto, E. Romero, M. Caballer, and C. De Alfonso, "Towards Migratable Elastic Virtual Clusters on Hybrid Clouds," *Proc. - 2015 IEEE 8th Int. Conf. Cloud Comput. CLOUD 2015*, pp. 1013–1016, 2015, doi: 10.1109/CLOUD.2015.139.

[4] S. R. Balakrishnan, S. Veeramani, J. A. Leong, I. Murray, and A. S. Sidhu, "High Performance Computing on the Cloud via HPC+Cloud software framework," *Proc. 5th Int. Conf. Eco-Friendly Comput. Commun. Syst. ICECCS 2016*, pp. 48–52, 2017, doi: 10.1109/Eco-friendly.2016.7893240.

[5] V. Simic, B. Stojanovic, and M. Ivanovic, "Optimizing the performance of optimization in the cloud environment–An intelligent auto-scaling approach," *Futur. Gener. Comput. Syst.*, vol. 101, pp. 909–920, 2019, doi: 10.1016/j.future.2019.07.042.

[6] M. Singh, "An Overview of Grid Computing," *Proc. - 2019 Int. Conf. Comput. Commun. Intell. Syst. ICCCIS 2019*, vol. 2019-Janua, pp. 194–198, 2019, doi: 10.1109/ICCCIS48478.2019.8974490.

[7] G. Zhang, Y. Yao, and C. Zheng, "HPC environment on Azure cloud for hydrological parameter estimation," *Proc. - 17th IEEE Int. Conf. Comput. Sci. Eng. CSE 2014, Jointly with 13th IEEE Int. Conf. Ubiquitous Comput. Commun. IUCC 2014, 13th Int. Symp. Pervasive Syst.* , pp. 299–304, 2015, doi: 10.1109/CSE.2014.83.

[8] M. Li, X. Yang, Z. Yu, and X. Li, "MyCloud: On-demand virtual cluster provisioning on HPC resources," *Proc. - 2013 IEEE Int. Conf. High Perform. Comput. Commun. HPCC 2013 2013 IEEE Int. Conf. Embed. Ubiquitous Comput. EUC 2013*, pp. 72–79, 2014, doi: 10.1109/HPCC.and.EUC.2013.20.

[9] J. HUTCHINSON, "Curtin trials DNA sequencing in Azure," *ITNEWS*, 2012.

https://www.itnews.com.au/news/curtin-trials-dna-sequencing-in-azure-271076.

[10] D. Chappell, “WINDOWS AZURE AND WINDOWS HPC Sponsored by Microsoft Corporation,” no. March, 2012.

[11] F. Ding, D. A. Mey, S. Wienke, R. Zhang, and L. Li, “A study on today’s cloud environments for HPC applications,” *Commun. Comput. Inf. Sci.*, vol. 453, pp. 114–127, 2014, doi: 10.1007/978-3-319-11561-0_8.

[12] M. Bahrami and M. Singhal, “A dynamic cloud computing platform for eHealth systems,” *2015 17th Int. Conf. E-Health Networking, Appl. Serv. Heal. 2015*, pp. 435–438, 2015, doi: 10.1109/HealthCom.2015.7454539.

[13] W. Nie, X. Xiao, Z. Wu, Y. Wu, F. Shen, and X. Luo, “The research of information security for the education cloud platform based on appscan technology,” *Proc. - 5th IEEE Int. Conf. Cyber Secur. Cloud Comput. 4th IEEE Int. Conf. Edge Comput. Scalable Cloud, CSCloud/EdgeCom 2018*, pp. 185–189, 2018, doi: 10.1109/CSCloud/EdgeCom.2018.00040.

[14] T. Passerini, J. Slawinski, U. Villa, and V. Sunderam, “Experiences with cost and utility trade-offs on IaaS clouds, grids, and on-premise resources,” *Proc. - 2014 IEEE Int. Conf. Cloud Eng. IC2E 2014*, pp. 391–396, 2014, doi: 10.1109/IC2E.2014.51.

[15] F. Isaila, J. Carretero, and R. Ross, “CLARISSE: A Middleware for Data-Staging Coordination and Control on Large-Scale HPC Platforms,” *Proc. - 2016 16th IEEE/ACM Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2016*, pp. 346–355, 2016, doi: 10.1109/CCGrid.2016.24.

[16] M. S. Kanna, “Loosely coupled MTC applications for multicloud deployment of computing clusters,” *Int. Conf. Electr. Electron. Optim. Tech. ICEEOT 2016*, pp. 4785–4790, 2016, doi: 10.1109/ICEEOT.2016.7755629.

[17] R. Bevilacqua, “A brief history of the evolution of HPC at a research institution in Argentina,” *CACIDI 2016 - Congr. Aergentino Ciencias la Inform. y Desarro. Investig.*, pp. 7–10, 2016, doi: 10.1109/CACIDI.2016.7785991.

[18] C. Langin, “We have an HPC system-now what?,” *ACM Int. Conf. Proceeding Ser.*, vol. Part F1287, 2017, doi: 10.1145/3093338.3093341.

[19] V. Karagiannis, “Compute node communication in the fog: Survey and research challenges,” *IoT-Fog 2019 - Proc. 2019 Work. Fog Comput. IoT*, pp. 36–40, 2019, doi: 10.1145/3313150.3313224.

[20] E. S. Jung, K. Maheshwari, and R. Kettimuthu, “Pipelining/Overlapping data transfer for distributed data-Intensive job execution,” *Proc. Int. Conf. Parallel Process.*, pp. 791–797, 2013, doi: 10.1109/ICPP.2013.93.

[21] R. F. E. Silva and P. M. Carpenter, “High Throughput and Low Latency on Hadoop Clusters Using Explicit Congestion Notification: The Untold Truth,” *Proc. - IEEE Int. Conf. Clust. Comput. ICCC*, vol. 2017-Septe, pp. 349–353, 2017, doi: 10.1109/CLUSTER.2017.19.

[22] M. Yang and H. Bie, “A low-overhead cluster management mechanism based on node information storage of max-heap tree,” *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019*, no. Itnec, pp. 1578–1582, 2019, doi: 10.1109/ITNEC.2019.8729272.

[23] T. T. Nguyen, H. Matsutani, and M. Koibuchi, “Low-reliable low-latency networks optimized for HPC parallel applications,” *NCA 2018 - 2018 IEEE 17th Int. Symp. Netw. Comput. Appl.*, 2018, doi: 10.1109/NCA.2018.8548063.

[24] M. Kang, D. I. Kang, J. P. Walters, and S. P. Crago, “A Comparison of System Performance on a Private OpenStack Cloud and Amazon EC2,” *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2017-June, pp. 310–317, 2017, doi: 10.1109/CLOUD.2017.47.

[25] S. Delfin, N. P. Sivasanker, A. Anand, and N. Raj, “Fog computing: A new era of cloud computing,” *Proc. 3rd Int. Conf. Comput. Methodol. Commun. ICCMC 2019*, no. Iccmc, pp. 1106–1111, 2019, doi: 10.1109/ICCMC.2019.8819633.

[26] H. Castro, M. Villamizar, O. Garces, J. Perez, R. Caliz, and P. F. P. Arteaga, “Facilitating the Execution of HPC Workloads in Colombia through the Integration of a Private IaaS and a Scientific PaaS/SaaS Marketplace,” *Proc. - 2016 16th IEEE/ACM Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2016*, no. December 2014, pp. 693–700, 2016, doi: 10.1109/CCGrid.2016.52.

[27] B. Power and J. Weinman, “Revenue growth is the primary benefit of the cloud,” *IEEE Cloud Comput.*, vol. 5, no. 4, pp. 89–94, 2018, doi: 10.1109/MCC.2018.043221018.

[28] K. Saritha, “Cloud in the De-Duplication Mechanism,” no. March, pp. 13–15, 2015.

[29] Z. Li *et al.*, “NeCTAR Research Cloud System,” vol. 12, no. 2, pp. 1–12, 2016.

[30] S. Xu, S. M. Ghazimirsaeed, J. M. Hashmi, H. Subramoni, and D. K. Panda, “Mpi meets cloud: Case study with amazon ec2 and microsoft azure,” *Proc.*

*IPDRM 2020 4th Annu. Work. Emerg. Parallel Distrib. Runtime Syst. Middleware, Held conjunction with SC 2020 Int. Conf. High Perform. Comput. Networking, Storage Anal.*, pp. 41–48, 2020, doi: 10.1109/IPDRM51949.2020.00010.

[31] A. Prabhakaran and J. Lakshmi, "Cost-Benefit Analysis of Public Clouds for Offloading In-House HPC Jobs," *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2018-July, pp. 57–64, 2018, doi: 10.1109/CLOUD.2018.00015.

[32] J. Weinman, "Hybrid Cloud Economics," *IEEE Cloud Comput.*, vol. 3, no. 1, pp. 18–22, 2016, doi: 10.1109/MCC.2016.27.

[33] S. Deshmukh and S. Sumeet, "Big Data Analytics Using Public Cloud Infrastructure: Use Cases and Cost Economics," *Proc. - 2015 Int. Conf. Comput. Intell. Commun. Networks, CICN 2015*, pp. 782–784, 2016, doi: 10.1109/CICN.2015.159.

[34] I. Pelle, J. Czentye, J. Doka, and B. Sonkoly, "Towards latency sensitive cloud native applications: A performance study on AWS," *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2019-July, pp. 272–280, 2019, doi: 10.1109/CLOUD.2019.00054.

[35] S. Sok, C. Plewnia, S. Tanachutiwat, and H. Lichter, "Optimization of Compute Costs in Hybrid Clouds with Full Rescheduling," *Proc. - 2020 IEEE Int. Conf. Smart Cloud, SmartCloud 2020*, pp. 35–40, 2020, doi: 10.1109/SmartCloud49737.2020.00016.

[36] S. Wu, L. Liu, H. Jiang, H. Che, and B. Mao, "PandaSync: Network and workload aware hybrid cloud sync optimization," *Proc. - Int. Conf. Distrib. Comput. Syst.*, vol. 2019-July, pp. 282–292, 2019, doi: 10.1109/ICDCS.2019.00036.

[37] H. M. Tufo *et al.*, "Janus: Co-designing HPC systems and facilities," *State Pract. Reports, SC'11*, 2011, doi: 10.1145/2063348.2063370.

[38] K. Ahmed, J. Liu, and K. Yoshii, "Enabling Demand Response for HPC Systems through Power Capping and Node Scaling," *Proc. - 20th Int. Conf. High Perform. Comput. Commun. 16th Int. Conf. Smart City 4th Int. Conf. Data Sci. Syst. HPCC/SmartCity/DSS 2018*, pp. 789–796, 2019, doi: 10.1109/HPCC/SmartCity/DSS.2018.00133.

[39] A. Souza, M. Rezaei, E. Laure, and J. Tordsson, "Hybrid resource management for HPC and data intensive workloads," *Proc. - 19th IEEE/ACM Int. Symp.*

*Clust. Cloud Grid Comput. CCGrid 2019*, pp. 399–409, 2019, doi: 10.1109/CCGRID.2019.00054.

[40] R. Guharoy *et al.*, “A theoretical and detail approach on grid computing a review on grid computing applications,” *2017 8th Ind. Autom. Electromechanical Eng. Conf. IEMECON 2017*, pp. 142–146, 2017, doi: 10.1109/IEMECON.2017.8079578.

[41] H. Setia and A. Jain, “Literature survey on various scheduling approaches in grid computing environment,” *1st IEEE Int. Conf. Power Electron. Intell. Control Energy Syst. ICPEICES 2016*, 2017, doi: 10.1109/ICPEICES.2016.7853429.

[42] J. Emeras, S. Varrette, and P. Bouvry, “Amazon elastic compute cloud (EC2) vs. in-house HPC platform: A cost analysis,” *IEEE Int. Conf. Cloud Comput. CLOUD*, no. Cc, pp. 284–293, 2017, doi: 10.1109/CLOUD.2016.44.

[43] K. Goga, A. Parodi, P. Ruiu, and O. Terzo, “Performance analysis of WRF simulations in a public cloud and HPC environment,” *Adv. Intell. Syst. Comput.*, vol. 611, no. 2016, pp. 384–396, 2018, doi: 10.1007/978-3-319-61566-0_35.

[44] S. Kortas and M. A. Shaikh, “Towards an HPC Service Oriented Hybrid Cloud Architecture Designed for Interactive Workflows,” *Proc. Urgent. 2020 2020 Int. Work. Urgent Interact. HPC, Held conjunction with SC 2020 Int. Conf. High Perform. Comput. Networking, Storage Anal.*, pp. 36–46, 2020, doi: 10.1109/UrgentHPC51945.2020.00010.

[45] J. Dantas, R. Matos, J. Araujo, and P. Maciel, “Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud,” *Computing*, vol. 97, no. 11, pp. 1121–1140, 2015, doi: 10.1007/s00607-015-0447-8.

[46] S. Dash and S. K. Pani, “E-Governance Paradigm Using Cloud Infrastructure: Benefits and Challenges,” *Procedia Comput. Sci.*, vol. 85, no. Cms, pp. 843–855, 2016, doi: 10.1016/j.procs.2016.05.274.

[47] S. Harrell and A. Howard, “Hybrid HPC Cloud Strategies from the Student Cluster Competition,” *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2018-July, pp. 186–193, 2018, doi: 10.1109/CLOUD.2018.00031.

[48] M. U. Tahir, M. R. Naqvi, S. K. Shahzad, and M. W. Iqbal, “Resolving Data De-Duplication issues on Cloud,” *2020 Int. Conf. Eng. Emerg. Technol. ICEET 2020*, pp. 17–21, 2020, doi: 10.1109/ICEET48479.2020.9048214.

[49] K. Vinay and S. M. D. Kumar, “Virtual Machine based Hybrid Auto-Scaling

for Large Scale Scientific Workflows in Cloud Computing," *Proc. 3rd Int. Conf. I-SMAC IoT Soc. Mobile, Anal. Cloud, I-SMAC 2019*, pp. 526–530, 2019, doi: 10.1109/I-SMAC47947.2019.9032507.

[50] N. Mangla and M. Singh, "Effect of scheduling policies on resource allocation in market oriented grid," *Proc. Turing 100 - Int. Conf. Comput. Sci. ICCS 2012*, pp. 212–216, 2012, doi: 10.1109/ICCS.2012.30.

[51] J. Lee, J. Ko, and Y. J. Choi, "Dhrystone million instructions per second–based task offloading from smartwatch to smartphone," *Int. J. Distrib. Sens. Networks*, vol. 13, no. 11, 2017, doi: 10.1177/1550147717740073.

[52] G. Tan, C. Shui, Y. Wang, X. Yu, and Y. Yan, "Optimizing the LINPACK Algorithm for Large-Scale PCIe-Based CPU-GPU Heterogeneous Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 9, pp. 2367–2380, 2021, doi: 10.1109/TPDS.2021.3067731.

[53] J. Dongarra *et al.*, "Livermore Loops," in *Encyclopedia of Parallel Computing*, D. Padua, Ed. Boston, MA: Springer US, 2011, pp. 1041–1043.

[54] W. J. Price, "A benchmark tutorial," *IEEE Micro*, vol. 9, no. 5, pp. 28–43, 1989, doi: 10.1109/40.45825.

[55] B. Armstrong, H. Bae, R. Eigenmann, F. Saied, M. Sayeed, and Y. Zheng, "HPC benchmarking and performance evaluation with realistic applications," *2006 SPEC Benchmark Work.*, 2006.

[56] JEAN-F. TOMB *et al.*, "Comparative benchmarking of cloud computing vendors with High Performance Linpack," *Nature*, vol. 388. pp. 539–547, 1997.

# List of Symbols and Acronyms

| | |
|---|---|
| HPC | High Performance Computing |
| GNU | GNU's not Unix |
| SaaS | Software as a Service |
| IaaS | Infrastructure As a Service |
| PaaS | Platform As a Service |
| LAN | Local Area Network |
| RAM | Random Memory Access |
| CPU | Central Processing Unit |
| Resj | Resj are the Resources such as compute nodes and memory allocated to the job |
| Execj | Execj is the application to be execute |
| Envj | Envj  is the software stack (e.g., operating system and libraries) |
| Dj | Dj is the Data of the job |
| Sj | Sj is the suitability of the job as determined by the user, application, or the HPC+Cloud is stored into the Privacy_Flag variable |
| Job_Pidj | Job_Pidj is the process Identifier of the job |
| UT | HPC Utilization Threshold |
| Geomean | Geometric Mean |
| Harmean | Harmonic mean |
| SSE | Streaming Single Instruction Multiple Data Extensions |
| FPU | Floating Point Unit |
| ICECCS | International Conference on Eco-friendly Computing and Communication Systems |
| ICPADS | International Conference on Parallel and Distributed System |
| MFLOPS | Millions Floating-point Operations per Second |
| MWIPS | Millions of Whetstone Instructions per Second |
| NeCTAR | National eResearch Collaboration Tools and Resources Cloud |
| OpenMP | Open Multi-Processing |

| | |
|---|---|
| Linpack | Linear Equations Package |
| COTS | Completely Off the Shelf |
| VPN | Virtual Private Network |

## Authorship Acknowledgement

“Every reasonable effort has been made to acknowledge the owners of copyright material. I would be pleased to hear from any copyright owner who has been omitted or incorrectly acknowledged.”