Benchmarking the cloud: A users perspective

Khushal Dave

Submitted as part of the requirements for the degree
of MSc in Cloud Computing
at the School of Computing,
National College of Ireland
Dublin, Ireland.

August 2014

Supervisor Mr. Vikas Sahni

Abstract

The increasing popularity of cloud computing and various options have manifolded over time. The foremost question in front of cloud users is how to precisely measure the performance capabilities of various options. The criteria for measuring performance could vary from measuring cost to floating point operations per second (FLOPS) or data manipulation (I/O).

Present benchmarking tools like Linpack,which measures performance in FLOPS can not evaluate the performance of multi-core instances. Being non-adaptive, these benchmark do not take system configuration and environment into consideration during testing. Currently, there are no properly designed benchmark suite which suitably adapts itself according to the node resources.

This research work focus on the designing of a scientific benchmark suite which will provide the maximum FLOPS performance of a CC over various CPU intensive tests.This will allow us to evaluate the performance on criterias like FLOPS and effect of multi-threading. This approach is completely automatized thus can be used on any compute instance starting from single node to multi-node instances.

The intelligent identification of suitable load data for individual tests which is automatically decided by the resource specification and the execution environment. Here we are using different data types to simulate CPU intensive tasks. This benchmark suite can be suitably used for any instance size without modification. The results obtained from these tests will allow the user to identify the suitable option amongst various CC.

The tool developed has been tested on Openstack and Azure instances and was able to provide the correct results to compare the performance. This can be easily used by any naive user to compare the performance of various CC's thus, allows easy comparison of performance.

Keywords: Linpack, Benchmark, Cloud, Performance, Microsoft Azure, Openstack,Adaptive-benchmarking.

Acknowledgment

Foremost, I would like to express my sincere gratitude to my supervisor, Vikas Sahni for the continuous support of my MSc study and research, for his patience , motivation, immense knowledge and support. His guidance helped me in completing this thesis in best possible way. I couldn't have imagined having a better mentor for this study.

Besides my supervisor, I would like to thank Pramod Pathak, Horacio González-Vélez, Robert Duncan. I am thankful to Keith Brittle, IT support and staff of NCI for their help in successful completion of this course. I feel humble due to the help provided by Alina Madalina Popescu for running the dissertation surgeries.

I would like to thank Appirio Ltd., for their support during the period of dissertation. I thank all who helped me directly or indirectly in any manner to finish my thesis work. I would like to thank my classmates, friends who have been very supportive in difficult time. Their motivation have helped me to concentrate on my work.

I express my sincere thank to my parents: Amrit Dave, Sadhana Dave, to Khushbu Vyas, Paritosh Vyas and Saksham for their continuous love, encouragement and support.This work is specially dedicated to my family because without them I couldn't have been here.

Contents

List of Figures

Listings

Chapter 1

Introduction

Linpack is the standard name when it comes to benchmarking. The basic parameter for measuring performance according to Dongarra et al. (2002) is the maximum number of floating point operations a processor can perform in one second. In this consumer centric era, the customer satisfaction is on the top priority. Consumer should be satisfied with the investment and the returns. The question seems to be valid because of the pay as you go nature of the cloud based services. To compare various available options one needs sophisticated comparison tools that gives correct results. This dissertation tries to provide an answer to the research question:

"Linpack for the cloud: Can we design a scientific benchmark Suite which when executed on distinct CC we can evaluate their performance based on different criteria (e.g. Cloud vendor/cost, Scientific calculation performance (flops) or with Data manipulation (I/O)?)?".

Majorly cloud services are available in three forms namely Infrastructure as a service (Iaas), Platform as a service (Paas), Software as a service (Saas) but, for the present research work we have only considered Iaas. Vedam & Vemulapati (2012) define benchmarking as executing a group of programs which stress a particular feature of the selected system using multiple workloads and examining the results to measure the performance.

There are so many benchmarks available, however there is a lack of in-depth research to design a benchmark which can analyze the performance of the system on various criteria altogether like scientific calculation performance (flops), Multi-threading. The adaptive benchmarking technique, in which benchmark adapts itself according to the environment, is also the today's need.

Conceptually, there is not much difference between benchmarking of a computing system and the Cloud system as indicated by Vedam & Vemulapati (2012). Although, there is a huge difference in case of execution of benchmarking approach and the calculation of the parameters. Generally most of the benchmark uses same techniques for measuring system parameters but Zhan et al. (2012), tries to draw attention to the fact that for every CC, the same benchmarking techniques cannot be applied. It should be used according to the instance characteristics.For example CC used for high volume throughput computing (HVC) like MapReduce will differ from the CC used for HPC. So they suggest that the benchmarks should be selected according to the CC and its usage.

As Gillam et al. (2013) stress that higher performance does not necessary mean that the system is better than other rather, it should be comparable throughout. So these points have been considered. Each parameter has a different effect for different category of application. We have identified the different parameters for performance evaluation and their effects on overall performance of an application in cloud environment.

This paper focus on the creating a benchmark which can compare the performance of CC using their FLOPS performance over various tests. It does not depend on their configuration. The adaptive nature of the benchmark allow it to identify the possible load data that can be used to test. The automatic and intelligent nature of this tool allows even a naive user to use this to effectively compare the performance of different available CC .

This document is structured as follows:

Chapter 2 discuss the background of the benchmarking and thus create the basic understanding of benchmarking and its concerns.

Chapter 3 presents the literature review around the research question and tries to identify the flaws in the present benchmarking system.

Chapter 4 discuss the design and architecture of the solution.

Chapter 5 elaborates the implementation detail and also highlights the key concerns.

Chapter 6 evaluates the benchmark results performed on various cloud platforms and also compares these with the readings from linpack.

Chapter 7 highlights the key findings and the future work.

Chapter 2

Background

This chapter covers the concept of Cloud computing, Services models, Performance, measuring criterias, Linpack and Benchmarking. The main aim of this chapter is to discuss the background of the research problem. This creates a basic outline of benchmarking and the shortcomings of present benchmarking system. The subsequent sections discusses the basics of these. Section 2.1 discuss Cloud computing, Service delivery and Deployment models. Performance and the various parameters on which it can be measured are discussed in section 2.2. Section 2.3 explains the concept of benchmarking and its requirement. The next section is dedicated to Linpack, which includes its history, method of operation and its criticism. The last section provide an introduction to the technologies used for this dissertation.

2.1 Cloud Computing

Cloud computing, which is usually referred as the cloud, delivers the computing resources on demand over the network Mell & Grance (2009). The resources can be anything from an application to a data center. These are provided over the internet as a metered service, so the user has to pay only for what they use.

2.1.1 Service model

The service delivery in Cloud Computing has been categorized into three different models based on the type of service offered.

Software-as-a-Service (Saas)

Mell & Grance (2009) explains SaaS as the cloud based application in which software application and its related data is kept in a central location. These application are usually accessed with help of a web browser.

Platform-as-a-Service (Paas)

PaaS provides the ability to develop application and deploy those on the cloud. This is performed by the tools and technologies provided by the service provider Mell & Grance (2009). This cloud based computing environment supports the full life cycle of delivering and building application. This platform include execution environment, web-servers, operating systems and Databases.

Infrastructure-as-a-Service (IaaS)

IaaS provides the computing resources which consists of networking, servers and storage Mell & Grance (2009). The compute infrastructure is normally provided in the form of virtual machines. The user can execute system software, applications over the machines. User does not have any control over the infrastructure but can manage the operating system and the applications.

2.1.2 Deployment model

Cloud services can be deployed in four ways which depends the structure of organization and the location of provision. The deployment models are namely private, public, community and hybrid model. As the majority of the deployment model used are public, so we have focused only on public clouds.

2.2 Performance

The performance of a computing system relates the amount of useful processing performed by it with the resources and time taken. The definition of performance may differ depending upon the context. For example: Fast processing of an Image, Short ping time in a network, High throughput computing (completing large number of jobs per unit time) etc.

2.2.1 Criterias of measuring performance

Depending upon the context of measurement the criteria for measuring performance of a computing system differs. Some of the criteria are as follows:

Cost of computation

The cost of computation relates to the total cost of resources consumed for computation.

Computational performance

It refers to the speed with which a task or the process is completed. It can be calculated in various forms including Operations per second (OPS), Floating point operations per second (FLOPS) etc.

Data Manipulation(I/O)

Data manipulation refers to the rate with which data can be processed. During this manipulation, the data needs to be transferred from one storage medium to another. This requires multiple I/O operations to be completed.

Latency

Latency refers to the time duration between occurrence of one event and the computing system to respond to that event. For example: Network Latency, which suggests the time taken by the message from one node to another in a network.

A virtual machine (VM) is a software based implementation of an abstract machine which executes program like a physical machine. Figure 2.1 displays that all the physical resources like CPU, network, memory and disk are virtualized. Multiple virtual machines can be deployed on a single physical machine thus share the physical resources between each other.

It's the responsibility of virtualization platform to effectively manage and distribute the resources amongst all the virtual machines. It is very difficult to provide complete isolation to a VM when you have multiple of them cooperating for resources. When we consider cloud based services it is very important to provide complete isolation and promised performance. The service level agreements (SLA) between service providers and consumers, makes this very important in terms of business continuity and customer retention. If you don't provide the said performance then you will have to pay for it one or the other way.

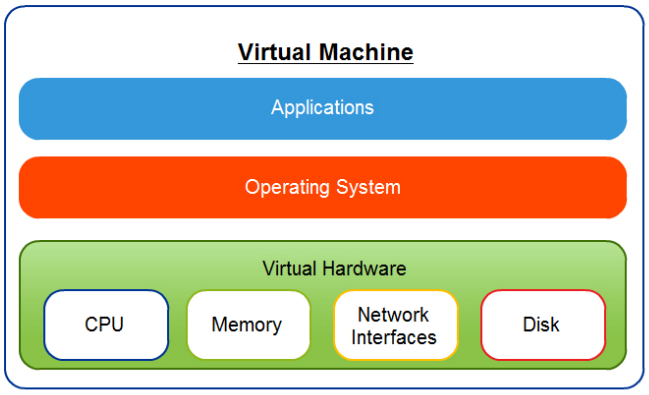

Figure 2.1: Virtual Machine

2.3 Benchmarking

It is the methodology which compare the performance of one subject to a standard performance values with the help of a specific indicator. In the case of a computing system, benchmarking is performed with the help of executing a program, group of programs or different tasks. This allows to analyze the relative performance of the subject under evaluation. This subject could be anything like processor (CPU), network and process scheduler. The purpose of benchmarking is due to various reasons which are as follows:

1. It is cumbersome to compare the performance of a computing system by just looking at the specifications. As a result, various benchmarking tests were developed.
2. Benchmark tries to stress system parameters to its limits so that highest achievable performance can be analyzed.
3. Performance of a computing system is a collective figure and does not depend on a single entity.

Binnig et al. (2009) has discussed the key points that are crucial for creating a benchmark for the cloud. They argue that the conventional benchmarks are not adequate to study the novel nature of cloud based services. They note that the biggest question for any new benchmark would be, to provide a metric to compare results for clouds from different vendors. In addition, the comparison becomes much more inevitable in the scenario where every vendor provide service in a different granularity. Cloud services in today's scenario vary from each other in various ways like cost, performance, consistency guarantees and others. A benchmark should help developers and system architects to select the best out of various available options.

The conventional benchmarks focus majorly on performance and cost. These parameters are still important for cloud based application. However, there are many more features specific to the cloud like scalability, fault-tolerance etc.

2.4 Performance:- the enabling factor

Presently, All the cloud offerings, offer same kind of services in the form of Compute Instances, Object storage, Compute clusters etc. Due to the virtualized nature of resources there is a variation in the performance . Customer require easy methods to check whether achieved performance copes with the expenditure. This question seems reasonable, when customer thinks of migrating to the cloud because selecting one option out of many needs proper methods to compare.With the help of this dissertation, we have tried to provide an overall solution that helps the user to identify the best offering. This offering will give them performance they require.

Present benchmarking methods that are available, mostly test a single resource for example Network performance like HttpPerf or computational performance like Linpack. Majority of user dont have a specific requirement as far as resources are concerned. However, they need a generic machine which can work as efficiently as it can with overall performance they require. For example If you have a CC which has low disk I/O , then whatever be the network bandwidth, single bottleneck does not allow the user to fully utilize that capacity.

2.5 Linpack

Linpack is a software benchmark which is used to measure the Floating point computing power of a computing system Dongarra et al. (2002). It calculates how fast a computing system can solve an n by n (dense) system of linear equations. We will have a look over the details in the following sections.

2.5.1 History

Linpack was developed in 1979 as a software based library written in FORTRAN by Jack Dongarra and others. It was used for performing mathematical linear algebra on computers. The creation of Linpack benchmark was merely an accident. It was originally designed to help the Linpack library users to find out the execution time required to solve a system of linear equations.

2.5.2 How it works

LINPACK uses BLAS (Basic Linear Algebra Subprograms) libraries for matrix and vector operations Dongarra et al. (2002). Use of BLAS helps in making efficient use of system hardware without even modifying the underlined algorithm. The Linpack benchmark measure the FLOPS power of a computing system. Linpack uses Lower Upper (LU) decomposition with partial pivoting. Solving the system of equations needs approximately O(n3) floating point operations. Thus time taken to solve the set of equation illustrates the FLOPS performed by the system.

The main objective of this benchmark is to estimate how quickly the system can solve real problems. This performance does not depicts the overall performance of a computing system but as the task is veritable the performance achieved is assumed to be high. The achieved performance closely relates to the peak performance specified for the system.

2.6 Technologies used

For the implementation we are using various technologies, which are as follows.

1. Openstack Cloud
2. Microsoft Azure cloud
3. Eigen Library
4. C++
5. Shell Script

2.6.1 OpenStack Cloud

OpenStack is an open-source software which can be used for building private and public clouds Openstack (2014). It handles big pools of compute,networking and storage resources for a data center. This is available in the Cloud Competency Center(CCC). The OpenStack being configurable by the administrator, allows us to create an equivalent specification of Azure instances. These instances have comparable system configuration.

2.6.2 Microsoft Azure

Microsoft Azure is a public cloud platform and infrastructure, managed by Microsoft for Iaas and Paas Microsoft (2014). Azure being a big player in cloud industry, is a suitable candidate for this type of research.

Openstack and Azure have been used for proof of concept(POC). This has helped us to test the tool over the equivalent CC's . The use of openstack, which is available as a private cloud on campus and Microsoft Azure which is a popular public cloud offering, both have allowed us to test the solution on wide variety of deployment models.

2.6.3 Eigen Library

We are using Eigen, which is a C++ template library for linear algebra: matrices, numerical solvers and related algorithms Guennebaud et al. (2010). It supports all matrix sizes. It is fast, reliable and has a detailed API which has eased the process of benchmark development. Most important point is that, it does not have any dependencies other than C++ standard library. We have used Eigen::Matrix class to create matrix for FLOPS performance evaluation.

2.6.4 C++

Developing system software could be done more easily with the help of object-oriented languages. As we are measuring performance of a system, we need a low level language having the power of objects. As per these, C++ was most suitable choice. With C++ we have used "g++" compiler which have allowed us to write more optimized code.

2.6.5 Shell Script

Shell Scripting is the scripting language in Linux platform. It provides great flexibility in terms of automation that can be achieved. We have used it for managing all the tasks together. As shells have specialized features for sharing data with other programs as well as for handling files. This has eased the process to implement the functionality.

Chapter 3

Literature Review

This chapter reviews the present literature regarding the benchmarking techniques and the key factors that needs to be considered while designing a benchmarking suite for a CC. In section 3.1, we have discussed how benchmarking helps us to peep inside the system and application performance which is the key enabling factor for benchmarking design. Dongarra et al. (2002) highlight that every application has a different resource requirement so needs to be taken into consideration while benchmarking design.

In section 3.2, we have discussed the basic benchmarking and different performance measuring criteria namely cost, computational performance and data manipulation. There are some gaps in the present research and we have tried to identify them in the subsequent sections.In section 3.3 we elaborate cost of computation and its effect. The section 3.4 is focused on the application performance in a multi-tenant environment. We have also discussed different types of application and some misconception that most benchmark designers have regarding performance evaluation, for example direct relation between application performances with the supplied resources.

In section 3.5 we have evaluated the present benchmarking system. We have also discussed the proper method of evaluation as suggested by Berndt & Watzl (2013) namely Calibration and Verification. Section 3.6 explains how this dissertation fills the gap and provide solution to some of the problems. On reviewing the present knowledge in this area, we would be able to identify the gaps and controversies to work upon. We have also thrown some light on the approach we have taken to achieve the aforementioned goal.

3.1 Benchmark: An Insight for cloud

The pay as you go nature of Cloud Computing and the consumer centric era are the key enabling factor for the design of benchmarking system as discussed by Rak & Aversano (2012). The popularity of cloud computing and the number of service providers are increasing day by day. Due to attractive advantages and prominent offerings by the vendors its more difficult to identify one. The availability of various options, each having similar cost, similar performance and highly dynamic nature of virtual resources makes this comparison even harder to judge. Jung et al. (2013) highlight that due to black box nature of the underlying technologies and virtualized infrastructure, the main obstacle is to identify the best option out of many. A recent study done by Agarwal & Prasad (2012) which analyzes the performance of storage services over Windows Azure cloud platform supports this to a greater extent. They highlight that even though there are huge variety of cloud offerings, majority of them are best-effort based services without any guaranty for performance. Thus it is evident that the performance promised may not be provided by the service provider. From the users perspective, they should be satisfied by the investment done for a specific resource. Users sometimes find it cumbersome as there are no well defined ways to compare various cloud offerings.

As pointed out by Dongarra et al. (2002), performance of a computing system is a complex phenomenon which depends on various parameters. This includes the algorithm, the size of data to be processed, the application, the operating environment and the level of optimization which has been achieved before processing. Jung et al. (2013) highlight that the resource requirement of every application is different from one another and majorly depends on the type of tasks it performs. Diversity in the resource requirement of computational processes has been an enabling factor for the design of benchmarks. Various studies have been performed to compare the performance of various cloud platforms and cluster computing. Most of the previous studies have overlooked the fact that only Linpack values do not guarantee high performance of application. In fact the scope of benchmarks should broader in terms of parameters on which they judge performance.

Dongarra et al. (2002) try to argue that generic benchmarks cannot be used to evaluate the performance, so we need specific benchmarks that are specially designed to match their behavior. This argument is well supported by similar research by Tudoran et al. (2012), who have specified that the performance has a lot to do with the communication, Input/output within modules of the application and the computational pattern. From the above discussion it can be said that benchmarking application enables us to compare and identify the best option. It also notifies that the same benchmarking technique is not applicable for all CC. In the next section, we will have a look over the different parameters on which the performance of CC depends.

3.2 Criteria for measuring performance

The performance of an application does not depend upon a single system parameter. In the initial stages of benchmarking, Linpack was designed to measure the performance of a processor. The basic parameter of measuring performance according to Dongarra et al. (2002) is the maximum number of floating point operations a processor can perform in one second usually known as FLOPS. Apart from this , there are some more units to measure a system's performance. When the High Performance Computing (HPC) application came into picture, the definition of Linpack seemed no longer valid. The performance of every application depends on a variety of resources to perform its computation, for example computational power, storage, network and supporting software.

According to Vedam & Vemulapati (2012) Benchmarking criteria has been majorly divided into these four segments which are as follows.

1. CPU
2. Memory I/O
3. Disk I/O
4. Network I/O

Apart from this every user also has different requirements and their own definition of performance. So to design a scientific benchmarking suite we need to identify the parameters first.For some users performance might depend on a single parameter like throughput. Keeping this in mind, Jung et al. (2013) keep their research limited to a comparison of the number of responses per minute for a web-based application. They mention the term performance capability to refer the computational power or the responsiveness of a CC. Some of the other researchers, including Ghoshal et al. (2011), restrict their research to only evaluating the I/O performance of CC. While most studies in the benchmarking have only focused on a single parameter and thus fail to generalize the relation between system performance and resources. In contrast to this Zhan et al. (2012) highlight that, for generic benchmarking workload for the benchmark suite should be a mix of all range. For making it more generic, the benchmark suite should be configurable according to the category of application.

The basic approach to perform benchmarking in the virtualized environment has been explained by Berndt & Watzl (2013) while detailing the two criteria for benchmarking. These are the determination of total performance known as calibration and verification of the fact that the virtual system is actually performing well in terms of expenditure. However, Li & Liu (2013) mention that attraction of large companies towards cloud computing is acting as a catalyst to improve the performance and handling of computational resources. This is due to the instant growing demand in many of the performance critical fields like scientific computing and the users which needs highly adaptive and scalable services.

The level of communication between multiple components of an application has a larger impact on its performance. Every action of an application has an impact on overall performance of the application as well as on the other applications of the system. Tudoran et al. (2012) infer that a high amount of data transfers between Virtual machines (VM) in public clouds can impact the nearby application and thus it could be said that performance is affected in a multitenant cloud environment. The upcoming section will discuss the role of these parameters while evaluating their effect on CCs performance.

3.3 Cost: A major factor of decision

Living in a consumer centric market, quality of services(QOS) and cost are the major deciding factors for using any service. The issues associated with the calculation of cost of computation are complex. There has been a huge amount of research which tries to compare the cost of computation on various cloud platforms. According to Tak et al. (2013), the main point to consider is that each available option in cloud affects the total cost of ownership in a different way. Their research is about relating the significance and meaning of cost of each possible choice available in their context. According to Tak et al. (2013), to handle a specific workload the amount of resource required can be found out by benchmarking the resource requirement for a unit workload. This unit cost could then be used with the empirical formula to identify the cost of bigger workloads. The total cost of computation depends upon various factors including computation cost and other costs.

Both Gillam et al. (2013) and Ostermann et al. (2010) have highlighted that apart from the performance the resource acquisition and release time are also crucial while benchmarking the CC because this time is also charged. Their studies have found there is high variance between these parameters. for designing benchmarksTak et al. (2013) suggests that, apart from quantifiable costs the costs of software license and hidden costs should also be taken into consideration. It may vary as the open source softwares are free while each instance of proprietary software could be charged per core or per socket. This cost is unknown to most of the cloud users and comes into picture after signing the contract and during migration. This problem may keep cloud users feeling fear to use them in future.

The low cost of ownership of cloud computing is taken into consideration by Iosup et al. (2011) to suggest it as cost effective alternative for HPC. I think they have overlooked the fact that performance critical application like scientific application needs more guarantees in terms of performance, resource allocation and scalability. The sharing of resources and certain performance bottlenecks like multi tenancy, data localization needs to be taken into consideration. In addition to this Church & Goscinski (2011) specify that the cost stabilization or linear performance increase is not necessary in all applications. Their study infer that highly parallel application should be used with pay as you use techniques. However this interpretation contrasts with that of Tak et al. (2013) who argue that the advantages of pay as you go model is not appealing for properly designed applications as there resource requirement do not vary much.

Tak et al. (2013) highlight the crucial question about the economic viability of the migration process to cloud and dig deeper to explain how the application can benefit from advantages such as cost savings and hassle free administration. This dilemma of consumer before migrating to cloud has been kept as a central point in their research. From the above discussion it could be deducted that Computational performance is the second most important factor on which the performance of any computational system can be evaluated. In the next section, we have discussed the key points and misconceptions between researchers about computational performance.

3.4 Computational performance: What needs to be highlighted?

Computational performance can be perceived as the speed with which a process or task is completed. According to Akioka & Muraoka (2010), it can be measured by using some basic benchmarking tools such as Linpack. The Linpack internally solves system of linear equation using dense matrices as explained by Dongarra et al. (2002). For identifying the performance for HPC HPLinpack benchmarking tool is used which is a lot different from Linpack. It creates, solves and confirms the results of a random system of equations over distributed systems. According to Dongarra et al. (2002), the HPC requires a high level of message passing over distributed computers and HPLinpack tries to emulate the same environment quite well.

Church & Goscinski (2011) have found out that some applications may perform worse on smaller instances even if you increase the number of instances. This signifies that this area needs detailed research in terms of root causes and its elimination so that the performance can be improved. Emulating the behavior is totally different from running practical and real time applications. There are lots of overheads and background tasks in real applications that cannot be duplicated in a testing environment and can only be. Due to interdependency between the performance measuring criteria like computational performance and other factors there should be a substantial thought process before designing the benchmarking architecture. This might create problem in designing due to its highly complex nature and thus pose questions for the feasibility of proper benchmarking suites .

Ghoshal et al. (2011) have raised a question about the performance benchmarking tests that execute suites like Linpack, NAS and IOR for system as well as network performance. These tests conclude that these platforms are not suitable for communication intensive applications. Furthermore Church & Goscinski (2011) note that the sharing of resources within virtual machines causes an overhead in terms of performance loss and computational cost. In a similar study Berndt & Watzl (2013), also have the same view on this. They suggest that to find the exact performance of a virtual machine the benchmarking of clouds should be done within the VM as it would help us to analyze the performance precisely. In the following section we have looked at the performance dependency on a CF on the way I/O is being implemented and supported.

3.5 Evaluation of present benchmarking system

To perform benchmarking properly Berndt & Watzl (2013) suggests that, one needs to consider isolation while evaluating the performance. A properly virtualized platform should show ability to give the same performance on different supporting hardware of same specification, provide linear scalability and should abstract the parallelism. They mention that it is important to verify the paid performance in virtual machines. There are various methods of benchmarking and interpreting the related findings. The benchmarking can help us in various ways as mentioned by Dongarra et al. (2002) that the performance can be highly enhanced by taking feedback from the underlying architecture. For example re-usability or localization of data in the memory hierarchy should be considered for faster processing. Creating the design environment which is in closed loop with the benchmarking ease the process of performance enhancement thus provides better result.

Luo et al. (2012) have discussed, why only floating point operations or I/O operations are not reasonable for benchmarking the whole cloud systems due to various reasons.

1. The floating point operations notify only the computing capability of the CPU. Many application consider data processing speed which depends not only on CPU while it depends on the performance of other operations as well. These operations include operations like memory access, network communication and disk access.

2. The compute intensive application does not represent large number of applications.

3. Science and engineering domain may have application in which CPU intensive tasks dominates but cloud systems have application which are CPU as well as I/O bound.

4. Various applications are largely computation based so using I/O rate as performance measuring criteria does not reflects the true capability.

The performance of a computing system is a complex issue and depends on various criteria of the system. The criteria includes various factors including the algorithm, problem-size, optimization capability of the compiler, Programming language used and the application to name a few. As Linpack only uses the dense linear systems, which does not represent operations performed in the scientific community. In addition, the performance of a computing system is a combined performance of all the system resource. A single one out of these can be a point of failure to achieve the peak performance. For example: Code which is not cache friendly can inhibit the system to reach its peak performance.

A similar study has been done by Ristov & Gusev (2013), which try to compare the performance of parallelization on windows azure. They have compared the performance of parallelization between Linux based OS with OpenMP and Windows based OS with its own C# based run-time environment for parallelization. The scope of their work is very limited to a single vendor and thus should not be generalized for other cloud service providers. However their work provides insight of how data placement in specific private caches could deliver better performance.

Ristov & Gusev (2013) have tested the performance using one to one mapping but it is not clear that whether thread affinity has been taken into consideration or not. As their study has revealed that the performance is higher if the data is placed in the private cache of a processor. We would like to study the performance while binding the threads to individual core. We would be studying the performance in a manner so that the results could be generalized.

Zhan et al. (2012) points out that the operations used in benchmarking code should also be selected according to the dominant operation types. For instance, HPC uses only Floating point operation. On the other side, High Volume Throughput Computing (HVC) application have very few floating point operation rather they have prominent I/O operation. Tudoran et al. (2012) try to highlight that most of the benchmarks find out discrepancies in the performance metrics. Whilst their support for in-depth analysis of the possible problems and the ways to eliminate is not promising. They also mention that some benchmarks use micro benchmarks like Network attached storage (NAS) and basic Linpack which do not analyze the performance on criteria like scalability and differences in the cloud platforms. So, it could be said that present benchmarking techniques still has some limitation and may not answer all the questions.

Some of the research like the study of Berndt & Watzl (2013) suggest that some applications known as embarrassingly parallel problems that work independently can scale easily because they require limited inter communication. This makes them appealing to be used on cloud for example database queries, video frame rendering and simulations. The feasibility of using these classes of problems to design applications and their performance has not been verified yet and should be taken into consideration for new benchmarking solutions .This can be taken as an opportunity to do more research to design better benchmarking platforms day by day.

A study in which Salah et al. (2011) have compared the performance of various Iaas offerings for CPU performance, Disk and I/O performance. There is point which needs to be noted that the virtual instances that have been used in their study are not of the same configuration. One instance has RAM of 7.8 GB while other has 1 GB. It is evident from their results that the first instance will definitely outperform the second one due to 8 times bigger RAM size. Ideally the comparison between two objects under consideration is viable only when both have same set of characteristics. However, they have acknowledged this limitation in their study. This notifies that the comparison should be done only comparable objects.

So, it has been inevitably clear that there exists a need for a new benchmarking methodology that takes the system configuration into consideration.

3.6 Contribution

So far, we have discussed the present benchmarking systems and associated problems. In a broader sense this question could be solved by designing a set of programs. These programs can be used to find the performance parameters and compare the associated resource consumption for same task on the multiple CC's. This have helped us to compare various options available to the user in form of various cloud offerings. The community is striving for a standard which can be used to measure the performance of a cloud platform based on various parameters. There is a substantial relationship between heterogeneous resource requirement and benchmarking design. It is evident from the above discussed concepts is that benchmarking systems needs to be broader in terms of the parameters on which they evaluate performance.

The critical analysis could only be performed when we have a closed loop between the design and testing. To provide an answer to this we have developed various benchmarking modules which can stress individual system parameter or resource. Combining all the module creates Scientific benchmarking suite as a whole. The modules could be programs, each stressing the system architecture over different boundary. Presently most of the research in community is confined to evaluation of performance over a single parameter. As we have identified the fact that every resource depends on one another so, the performance can not be generalized on the basis of one resource. Scientific applications being computation intensive are looking forward to be benefited from performance boost with cloud computing. This could be achieved only when we can perform an in depth analysis of the interaction among the segments of application and their effects on one another. This requires more sophisticated benchmarking suites which can evaluate the performance in a more critical manner. Once we reduce performance uncertainties and bottlenecks, the gratification of the users will definitely increase the adoption of cloud.

Chapter 4

Design

This chapter provides the design goal of developing a tool to identify the best CC from various available options. The analysis of the problem has been done in the literature review. This chapter will discuss the key design decisions, selection of technologies which have been used and their roles in the overall solution.

This tool is a leap ahead of the Linpack which only measures the performance of a computing system. Here we are not only considering FLOPS performance but we are also trying to study the effect of multi-threading on the system performance. The mathematical operation that are being performed are a proper mix of multiplication and addition. The in-detail performance evaluation will provide us the insight to effectively use Iaas.

The proposed solution study the effect of multi-threading on the overall performance of the system. In a conventional computing system, the OS manages threads and the processes. However in the virtualized environment of cloud computing, virtualization layer is responsible for mapping virtual resource to physical resources. It has to make sure that the resource are effectively shared between virtual instances. Any virtualization platform needs to be effective and efficient in the mapping of the resources. The summation of individual virtual performance should be nearly same as the performance capability of the physical hardware.

4.1 Proposed solution

The proposed solution focuses on measurement of performance from two different perspective.

4.1.1 Computational performance [FLOPS]

The solution identify the maximum number of FLOPS performed by the instance. It has been done by creating a CPU intensive task in which majority of the data which is being processed is of type double, float. The time taken for completing a fixed amount of floating point operation gives us the FLOPS reading for that Instance. In addition, we have also considered the Integer performance of the CC.

4.1.2 Multi-Threading Performance

Most of the application nowadays are multi-threaded. They use threads to segregate the work into chunks that can be done simultaneously. For example a web-server which have a daemon thread continuously listening for incoming requests from the clients. When the client sends a request to the server, the daemon thread forks a child thread which handles the client request and returns. Most of the code section is shared by threads which has been tried to replicate in the test. Multi-threaded performance has now became an important factor in deciding the suitability of any computing system.

The modules try to create an execution environment in which we have replicated the sharing model amongst threads.

4.2 Architecture of System

The architecture of the system has been depicted in figure 4.1. Here we have decomposed the process into separate modules that could be called individually and independently.

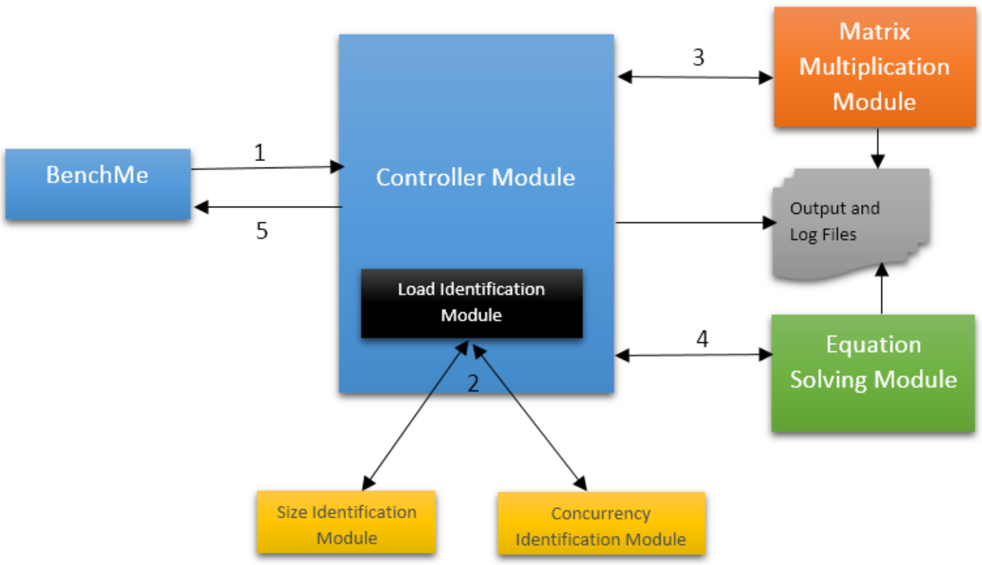
Matrix Multiplication Module
3
BenchMe
1
5
Controller Module
Output and Log Files
Load Identification Module
4
2
Equation Solving Module
Size Identification Module
Concurrency Identification Module

Figure 4.1: Architecture of Benchmarking tool

The proposed Benchmarking Tool consists of three modules namely Controller Module, Load Identification Module and Execution Module. Execution module includes Matrix multiplication module and Equation solver module. load Identification module is a combination of concurrency and size identification module with associated automation scripts.

The main technologies uses in this solution are C++, Eigen Library and Shell Scripting. The brief role of these technologies have been included in section 2.6 .

The Controller Module controls all the benchmark execution and coordinates various tasks in a transparent manner. It coordinates with various modules to execute various benchmarking tests. It is responsible for coordinating with set-up the data required for the tests. After each execution of all the tests it collect the results and present it in a manner that can be used by a normal user for comparison.

The Load Identification Module identifies the input for various tests according to the system resources available like available memory, hardware concurrency, data type specifications. These module removes the problem of supplying the load data for benchmarking tests thus totally automate the work. Various other solution needs either users to provide the load inputs or have a standard data which is used on all execution environment. As identified in chapter 3, that the same benchmarking methods should not be used on all the computing systems. Thus the technique used here checks system's ability and thus intelligently identify the possible load data that should be used to test the system performance. The load identification module is invoked before execution of each test. It identifies the load-data based on the system resources and configuration.

Execution module constitutes of two sub-modules namely, Matrix multiplication module and Equation solver module. Matrix multiplication modules executes the matrix multiplication tasks rigorously with different data types and matrix sizes. It records the results in a file that is used to aggregate the results.

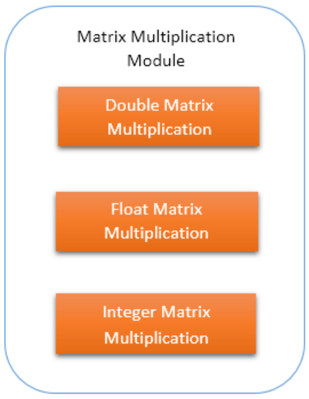
Matrix Multiplication Module
Double Matrix Multiplication
Float Matrix Multiplication
Integer Matrix Multiplication

Figure 4.2: Matrix Multiplication Module

Equation solving module consists of two tests. It tests the system performance while trying to solve linear systems. We have used LU decomposition with partial pivoting. The tests are performed and results are saved in the files that is used by Controller in the later phases.

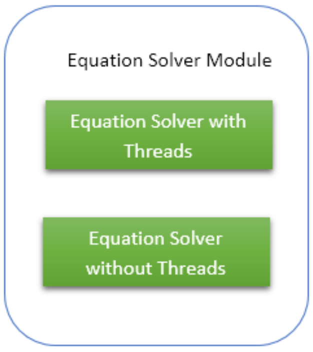

Figure 4.3: Equation Solver Module

In the last step the Controller combines all the results from various modules and provide easy to understand results to the user. In addition, it frees all the resource acquired for the tests and error handling.

Chapter 5

Implementation

In this chapter, we have explained the implementation and the set up of the environment for benchmarking a CC. The implementation of the controller module has been explained in section 5.2. Section 5.3 discuss the implementation detail of load identification and resource identification module. The implementation detail of matrix multiplication module and equation solver module have been discussed in 5.4.

5.1 Instance specification

We have used Opsnstack and Azure cloud for the implementation purpose. We have used different instances whose specification is as depicted in table 5.1. We have replicated the exact configuration of Azure instance in the Openstack private cloud. The clock rate of each Core or VCPU is 2.2 GHz. The A4 instance of Azure which emulates Intel sandy bridge has the clock rate of 2.5 GHz.

Sr. No	Cloud vendor	Instance type	CPU core	RAM	Cache
1.	Azure	A1	1 Core	1.75 GB	20 MB per Core
2.	Azure	A2	2 Core	3 GB	20 MB per Core
3.	Azure	A3	4 Core	7 GB	20 MB per Core
4.	Azure	A4	8 Core	14 GB	20 MB per Core
5.	Openstack	A1	1 VCPU	1.75 GB	512 KB per VCPU
6.	Openstack	A2	2 VCPU	3 GB	512 KB per VCPU
7.	Openstack	A3	4 VCPU	7 GB	512 KB per VCPU
8.	Openstack	A4	8 VCPU	14 GB	512 KB per VCPU
9.	Openstack	A4	8 VCPU	14 GB	4096 KB per VCPU

Table 5.1: Configuration of the instances used for evaluation

5.2 Controller Module

The task of coordination and control of the complete benchmark process is handled by this module. The development of this module has been done using shell script which have helped us to automate various tasks. It is adaptive in the matter of identifying load data. It automatically identifies the suitable load data with the help of Load identification module which invokes the Resource identification module. After the load data setup the benchmark process starts. In the last step the controller module prepares the result and informs the user about it.

5.3 Load Identification Module

The load identification module identifies the amount of free memory in the system using system commands.

```
free -b| awk '/-/ {print $4;}'>>freem.txt
read freem<freem.txt
rm freem.txt
```

Listing 5.1: Code for Identification of free memory in the system

Once the available free memory has been identified, this module calls size identification module which provides the amount of storage used to store a data type on that system. The size of the data type is identified as depicted in listing **??**.

```
float f;
int i;
double d;
int main()
{
std::cout << "int: " << sizeof(i) << ' '
<< "float: " << sizeof(f) << ' '
<< "double: " << sizeof(d) << ' ';
}
```

Listing 5.2: Code for Identification of data type size

The output of this code is moved to a file using a pipe. This file is then used to fill the values in the variables used in code 5.3.

```
./size | cat >> size.txt
read a b c d e f < size.txt
rm size.txt
```

Listing 5.3: Code for getting data type size

The values of the variable b, d, f corresponds to the number of bytes used to store an integer,float and double data type on that platform. The values of free memory and data type size are then used to identify the maximum possible load matrix size for that system. The calculated values are then used as load values by the execution module. We are keeping maximum used memory for the test run as 90% of the available free memory so that we keep ample space for the normal OS execution.

```
Memorybound=$((freem * 90/100));
  MatNum=$((hc + 2));
  echo $MatNum| cat >> logs.txt;
  Memorybound=$(awk "BEGIN {printf \"\%.2f\",${Memorybound}/${MatNum}}")
  Memorybound=$(awk "BEGIN {printf \"\%.2f\",${Memorybound}/${g}}");
  Mmax=$(echo "sqrt ( $Memorybound )" | bc -l) ;
  Mmax=${Mmax\%.*};
```

Listing 5.4: Identification of load matrix size

The total number of matrices are calculated by adding two to the hardware concurrency.We have two shared global matrix with one matrix associated with each thread. The division of available memory in bytes with number of matrices required and size of data type gives us the maximum number of bytes for a single matrix. Taking the square root of that values gives us the maximum order possible for a single matrix.

5.4 Execution Module

The execution module consists of two modules namely Matrix multiplication module and Equation solver module. Matrix multiplication module tests the performance of the system while using input matrices of different data type. Before any compilation, we need to provide the Eigen core library to the modules.The tool already has Eigen core library in the folder. So there is no need to explicitly download them. However, this could be done as follows:

1. Go to the Eigen library home page http://eigen.tuxfamily.org/index.php?title=Main_Page.
2. Download the Eigen 3.2.1 from the home page.
3. Extract the files and copy the Eigen folder in usr/include.

4. You can also keep the Eigen folder in the present working directly where you are executing the benchmark.

The code listing 5.5 is used to compile each module with specification of various flags.

```
sudo g++ -Ofast -march=native -std=c++0x MaxPerformanceGFLOPSforFloat.cpp -o MaxperF ↩
    -pthread
```

Listing 5.5: Compilation of source code

We have used various flags during compilation. Table 5.2 explain the flags used and the reason for their use.

Sr. No.	Name of flag	Reason of use
1.	-Ofast	To highly optimize the code for mathematical calculation
2.	-march=native	To enable all the instruction subsets supported by the underlined hardware
3.	-std=c++=x	To provide compiler and library support for the ISO C++ 2011 standard
4.	-pthread	To instruct linker to add support for multi-threading using POSIX library

Table 5.2: Various compilation flags used in the tool

Once the source code is compiled, the controller module starts rigorous execution of the matrix multiplication module.

```
NOW=$(date +"%F_%T");
for i in 1 2 3 4 5 6 7
do
Mstart=$MstartI;
ResultFile="Result-$NOW-$Mstart-$Mlast.csv"
while [ $Mstart -le $Mlast ]
do
./MaxperF ${Mstart} ${ResultFile}
Mstart=$(($Mstart + $MStepsize));
done
done
```

Listing 5.6: Code for executing the Matrix multiplication module

The C++ code creates global Eigen::Matrix objects. It identifies the hardware concurrency and then creates the threads accordingly.

```
MatrixXf X = MatrixXf::Random( N, N);
MatrixXf Y = MatrixXf::Random( N, N);
std::vector<std::thread> threads;
```

```
// Identifies the hardware concurrency
int numCPU = sysconf( _SC_NPROCESSORS_ONLN );
```

Listing 5.7: Code for creating matrix and identifying hardware concurrency

The module then performs the matrix multiplication task using threads. It keeps track of the exact CPU time taken for the matrix multiplication operation.

```
std::chrono::time_point<std::chrono::system_clock> tstart,tend;
tstart = std::chrono::system_clock::now();
R.noalias() = X * Y ;
tend = std::chrono::system_clock::now();
std::chrono::duration<double> elapsed = tend-tstart;
```

Listing 5.8: Code for Matrix multiplication

At the end a common variable provides the total effort by all the thread. The consistency of the shared variable is prevented by using locking the std::mutex object.

```
mtx.lock();
mflopsCount+=((2*(Num_Of_Op/1000000))/(long double)(elapsed.count()));
mtx.unlock();
```

Listing 5.9: Code for FLOPS count

Similar procedure is followed for the equation solver modules in which the code in listing 5.10 is used to set of equations.

```
    tstart = std::chrono::system_clock::now();
    {
        X.noalias() = A.lu().solve(B);
    }
    tend = std::chrono::system_clock::now();
```

Listing 5.10: Code for solving Equation

As the number of floating point operation in solving set of linear equations are different. A different formula as displayed in listing 5.11 is used to calculate the same.

```
ops = ( double ) ( 2 * ((double)N) * ((double)(N * N)) ) / 3.0 + 2.0 * (( double ) ( ↩
     N * N ));
```

Listing 5.11: calculation of number of operation needed in Equation solving module

Chapter 6

Evaluation

The main aim of this benchmarking tool is to evaluate the performance of any CC. It is a cloud-agnostic solution. In general, to evaluate the performance of a cloud configuration there are so many points that needs to be taken into consideration. The difference in performance can not be identified by just looking at Instance specifications.

Moreover, the user does not have to worry about the number of available cores, memory and platform specific attributes like bytes taken to store a data type. Intelligent auto identification of the complete range of possible load data solves this problem. It identifies the possible set of load data before each test. The solution is environment-agnostic in a sense that, it also takes into account the number of bytes taken to store numeric data type.

6.1 Readings from instance

We have executed the benchmarking on various instances sizes ranging from single CPU instance to Multi CPU instances. The tool intelligently automate the load identification for benchmark execution thus removes the need for a user to know the specifics about test data. Table 6.1 displays the maximum matrix size used for tests in the benchmarking. The minimum matrix size for all the test has been kept constant at 500 with step size of 500.

Instance type	Double Matrix Multiplication	Float Matrix Multiplication	Integer Matrix Multiplication	Equation Solve Threaded	Equation Solve Normal
A1	8000	11500	11500	8000	8000
A2	10000	14000	14000	10000	11500
A3	11500	16500	16500	11500	16500
A4	12500	18000	18000	12500	23000

Table 6.1: Matrix size used for Benchmarking

The tool tries to test the performance using matrix size shown in table 6.1. If in case the system is not able to allocate the required memory due to any problem, we have added the exception handling in each module which handles the exception and continues further without error.

We have taken the reading on various instance sizes on Microsoft Azure and Openstack cloud. Every instance is executing on Ubuntu 14.04 LTS operating system.There is a major difference in the maximum performance achieved from these instances as shown in figures below.

The figures below depicts the performance readings of the instances on Linpack 1000. The performance of Azure instances on the Linpack benchmark as shown in figure 6.1 is higher than the openstack instances. It is higher than the Openstack A4 instance which has much more CPU than Azure A4 instances. The main drawback of Linpack is that it does not adapt itself according to the instance specification. The performance of A4 instances is definitely higher than A1 instances which is not reflected in the readings. The main reason for this behavior is that the Linpack is not multi-threaded so being CPU intensive process, it is mapped to only one CPU thus, can not fully utilize the complete set of resources.

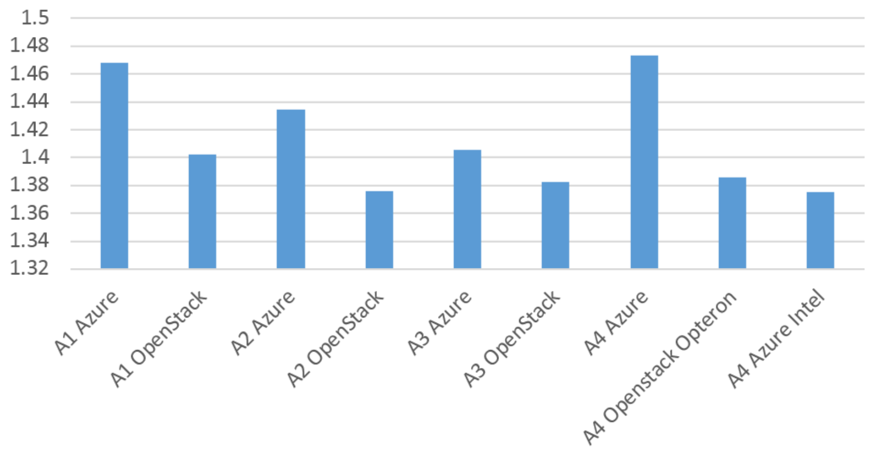

Figure 6.1: GFLOPS performance reading of various instances on the Linpack 1000 benchmark

On the other side, the performance readings of instances on different modules clearly shows a performance growth with the increase in the system resources. It is evident from the graphs that, the Azure instances outperforms over Openstack instances.

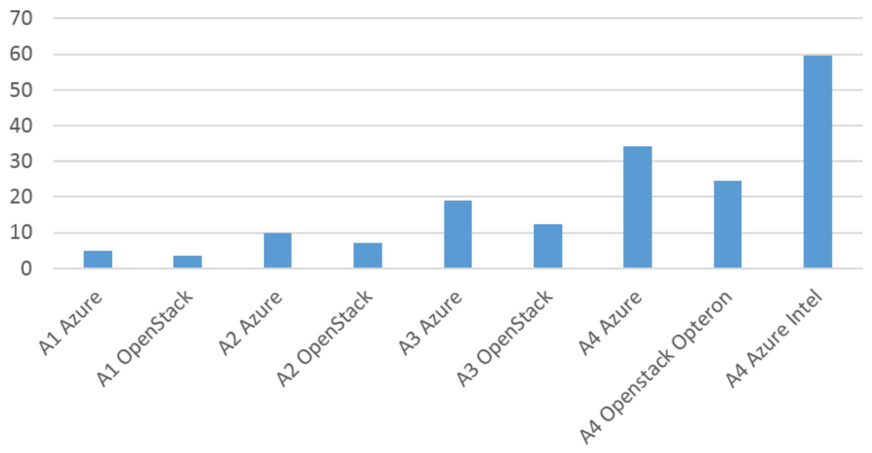

Figure 6.2: GFLOPS performance of instances on the Matrix multiplication module with double data type

To compare the like instances, we have tried to get as close to the Azure instances as we could. Withal, there is a limitation on the variety of configuration of instance that can be emulated in the Openstack private cloud. This limitation is ruled by the list of processor family that can be emulated by the hypervisor. In addition, the possible reason for the difference in the performance are as follows:

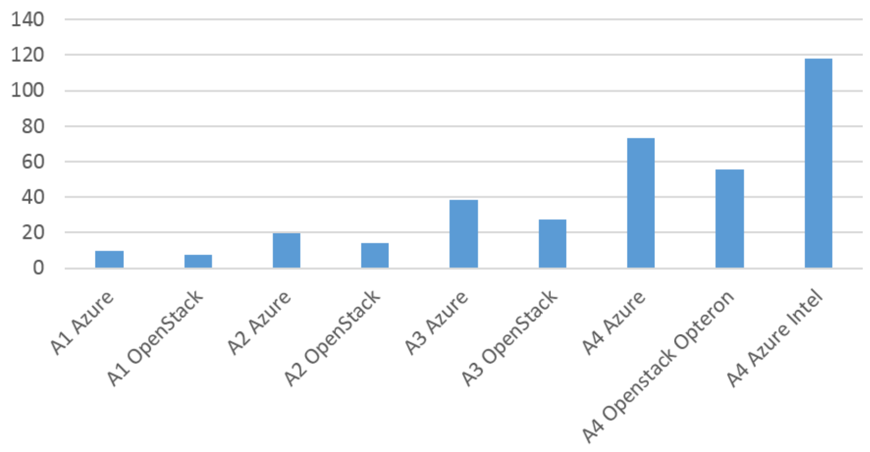

Figure 6.4: Integer OPS performance of instances on the Matrix multiplication module with Integer data type

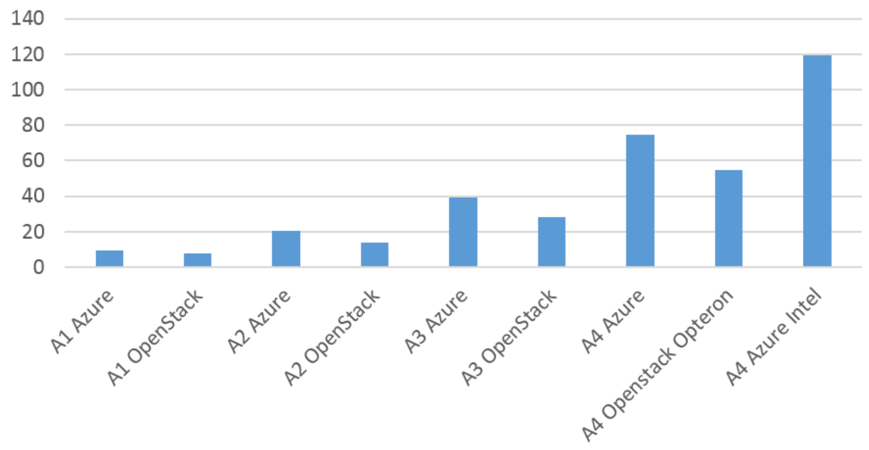

Figure 6.3: GFLOPS performance of instances on the Matrix multiplication module with float data type

The instances in Opnestack has small Cache which is the performance bottleneck. This small cache can not handle large data and thus performance degrades. Furthermore, Cache size of 512 KB not even comparable to 20 MB.

To study the effect of multi-tenancy on the instance performance, we tried to isolate one A4 instance in the openstack. we emulated Intel Sandy bridge with clock rate 2.5 GHz and cache size of 4MB, the performance received was around double for most of the performance tests. This should be noted that this performance boost is much more than Azure which has about 20 MB of cache.

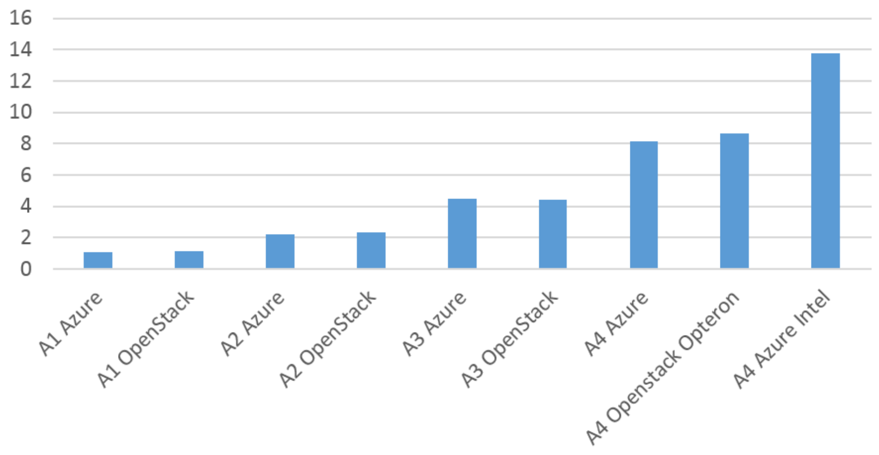

Figure 6.5: GFLOPS performance of instances on the Equation solver module with threads

The reason for the above performance boost is mere isolation. The isolated Intel Sandy bridge instance in Openstack is free from multi-tenancy. Thus the instance is not contending for resources and practically has complete ownership. Readings from the above tests assures that the multi-tenancy in the cloud platforms is the biggest barrier in the achieving the capable performance. In shared environment, the instance contends for the resources and waste a large amount of time which could be utilized for processing data. The effect of performance degradation could be reduced if the underlined fabric manage the resource in much more efficient manner.

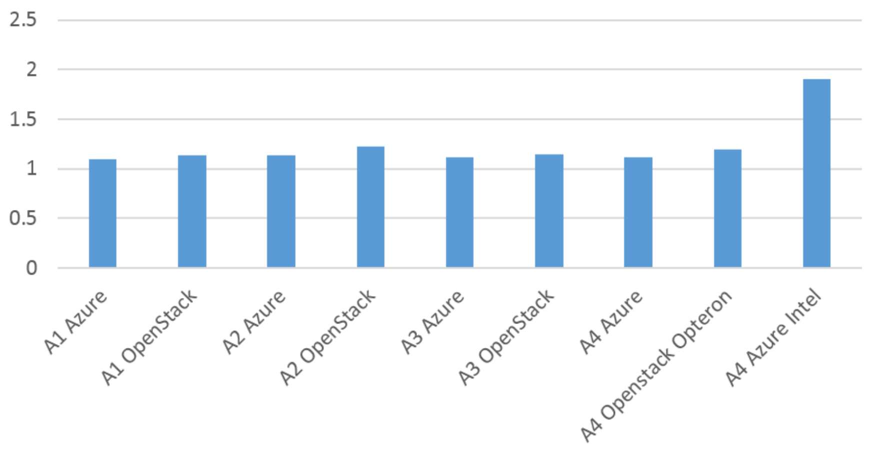

Figure 6.6: GFLOPS performance of instances on the Equation solver module without threads

To verify that the tool is giving the correct results we executed a basic program which has a loop and tries to perform double multiplication iteratively. We executed the program and calculated the time taken for the total execution on each instance. As this code runs on a single core thus it should replicate the behavior of Equation solver module without thread. Figure 6.7 clearly shows that the variation in performance are in resonance for both the tests. Thus, the tool accurately identifies the performance in an accurate manner.

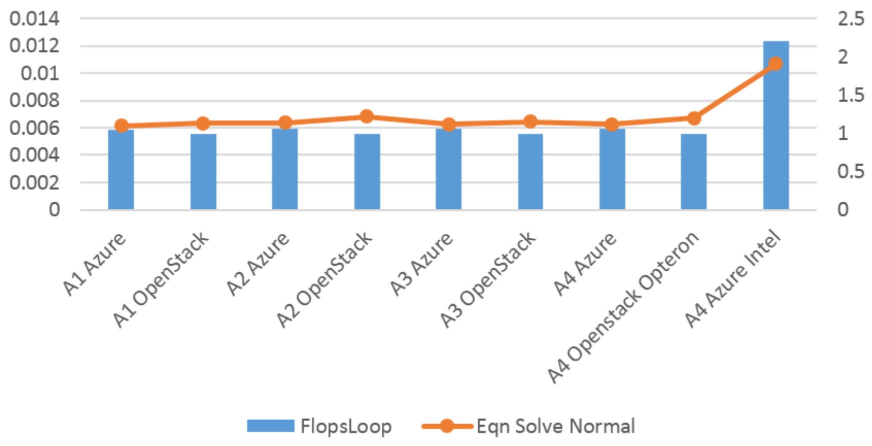

Figure 6.7: Verification of the accurateness of the tool

From the above readings it is evident that the standard Linpack benchmark is not suitable for the multi-core instances that are available now a days. Graphs clearly shows that its unadaptive nature is unable to push multi-core instance to its limits. Thus when a user executes the Linpack benchmark the results no longer relates to the maximum achievable performance. Furthermore, the code is not highly parallelized so as to find the best possible performance. We have created the benchmark which mimics the normal execution scenario which is multiple threads with shared code and data.

Chapter 7

Conclusion

This thesis proposed a scientific benchmark suite which can be used for evaluating the performance of different CC's based on various parameters including cloud vendor/cost, scientific performance(FLOPS), data manipulation(I/O) etc.

The evaluation chapter compared the performance of nearly equivalent instances on different cloud platforms. In this sense, various instances of different configuration were examined for their performance.

The proposed benchmarking tool has following advantages: it automatically adapts to the system environment thus intelligently determines the load matrix size for each test. This tool does not have any dependency except Eigen library. However these files are kept with other files so user does not have to worry about the dependencies. In addition, its adaptive nature gives it an edge over other benchmarking tools.

Though this tool has been tested on Azure and openstack cloud. Different cloud platforms which has Linux images can be bench-marked using this tool.

The initial development of this tool has focused on scientific performance (flops) on cc while taking into account the hardware concurrency. The future work will consider various other parameters including network performance, data manipulation(I/O) etc.

As we have seen from the performance graphs that the emulated cache plays a major role in performance of a CC. Thus the inclusion of cache specification in load identification and performance evaluation should also be considered. Further more creating various versions of the tool depending upon the underlined Operating system used can also be looked upon.

Bibliography

Agarwal, D. & Prasad, S. (2012), Azurebench: Benchmarking the storage services of the azure cloud platform, *in* 'Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International', pp. 1048–1057. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6270754 [Accessed 4th July 2014].

Akioka, S. & Muraoka, Y. (2010), Hpc benchmarks on amazon ec2, *in* 'Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on', IEEE, pp. 1029–1034. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5480963 [Accessed 7th December 2013].

Berndt, P. & Watzl, J. (2013), Unitizing performance of IaaS cloud deployments, *in* 'Services (SERVICES), 203 IEEE Ninth World Congress on', IEEE, pp. 356–362. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6655721 [Accessed 1th December 2013].

Binnig, C., Kossmann, D., Kraska, T. & Loesing, S. (2009), How is the weather tomorrow?: Towards a benchmark for the cloud, *in* 'Proceedings of the Second International Workshop on Testing Database Systems', DBTest '09, ACM, New York, NY, USA, pp. 9:1–9:6.
URL: *http://doi.acm.org/10.1145/1594156.1594168*

Church, P. & Goscinski, A. (2011), Iaas clouds vs. clusters for hpc: A performance study, *in* 'CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization', pp. 39–45. . [Online]. Available from http://www.thinkmind.org/index.php?view=article&articleid=cloud_computing_2011_2_30_20124 [Accessed 5th December 2013].

Dongarra, J. J., Luszczek, P. & Petitet, A. (2002), 'The linpack benchmark: Past, present, and future'. [Online]. Available from http://cluster.earlham.edu/project/curriculummodules/JOCSE_PetaKit/Resources/LINPACK.pdf [Accessed on 20th November 2013].

Ghoshal, D., Canon, R. S. & Ramakrishnan, L. (2011), I/o performance of virtualized cloud environments, *in* 'Proceedings of the second international workshop on Data intensive computing in the clouds', ACM, pp. 71–80. [Online]. Available from http://www.researchgate.net/publication/228448356_IO_performance_of_virtualized_cloud_environments/file/5046351af656a9a809.pdf[Accessed 15th November 2013].

Gillam, L., Li, B., John, O., Tomar, A. P. et al. (2013), 'Fair benchmarking for cloud computing systems', *Journal of Cloud Computing: Advances, Systems and Applications* **2**(1), 6. [Online]. Available from http://www.journalofcloudcomputing.com/content/2/1/6 [Accessed 15th February 2014].

Guennebaud, G., Jacob, B. & other (2010), 'Eigen v3', http://eigen.tuxfamily.org. [Online] Available from: http://eigen.tuxfamily.org/index.php?title=Main_Page [Accessed 25 Jun. 2014].

Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T. & Epema, D. H. (2011), 'Performance analysis of cloud computing services for many-tasks scientific computing', *Parallel and Distributed Systems, IEEE Transactions on* **22**(6), 931–945. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5719609 [Accessed 15th November 2013].

Jung, G., Sharma, N., Goetz, F. & Mukherjee, T. (2013), Cloud capability estimation and recommendation in black-box environments using benchmark-based approximation, *in* 'Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on', IEEE, Santa Clara, CA, pp. 293–300. [Online].Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6676707[Accessed 19th November 2013].

Li, B. & Liu, F. (2013), 'Cloud and data center performance [guest editorial]', *Network, IEEE* **27**(4). [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6676707 [Accessed 19th November 2013].

Luo, C., Zhan, J., Jia, Z., Wang, L., Lu, G., Zhang, L., Xu, C.-Z. & Sun, N. (2012), 'Cloudrank-d: benchmarking and ranking cloud computing systems for data processing applications', *Frontiers of Computer Science* **6**(4), 347–362. [Online]. Available from http://link.springer.com/article/10.1007/s11704-012-2118-7 [Accessed 22nd June 2014].

Mell, P. & Grance, T. (2009), 'The nist definition of cloud computing', *National Institute of Standards and Technology* **53**(6), 50. [Online] Available from: http://www.profsandhu.com/cs6393_s13/nist-SP800-145.pdf [Accessed 18 July. 2014].

Microsoft (2014), 'Azure: Microsoft's cloud platform — cloud hosting — cloud services', [Online]. *Microsoft Azure*. Available from: https://azure.microsoft.com/en-us/ Accessed 1-August-2014.

Openstack (2014), 'Home openstack open source cloud computing software', [Online]. *Openstack Cloud Software*. Available from: http://www.openstack.org Accessed 19-July-2014.

Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T. & Epema, D. (2010), A performance analysis of ec2 cloud computing services for scientific computing, *in* D. Avresky, M. Diaz, A. Bode, B. Ciciani & E. Dekel, eds, 'Cloud Computing', Springer, pp. 115–131. [Online]. Available from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5719609 [Accessed 21st February 2014].

Rak, M. & Aversano, G. (2012), Benchmarks in the cloud: The mosaic benchmarking framework, *in* 'Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on', IEEE, Timisoara, pp. 415–422. [Online]. Available from http://www.chinacloud.cn/upload/2009-06/09063000144624.pdf [Accessed 3rd December 2013].

Ristov, S. & Gusev, M. (2013), Performance vs cost for windows and linux platforms in windows azure cloud, *in* 'Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on', pp. 214–218. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6710581 [Accessed 14th July 2014].

Salah, K., Al-Saba, M., Akhdhor, M., Shaaban, O. & Buhari, M. I. (2011), Performance evaluation of popular cloud iaas providers, *in* 'Internet Technology and Secured Transactions (ICITST), 2011

International Conference for', pp. 345–349. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6148463 [Accessed 11th July 2014].

Tak, B. C., Urgaonkar, B. & Sivasubramaniam, A. (2013), 'Cloudy with a chance of cost savings', *IEEE Transaction on Parallel and Distributed System* **24**(6), 1223–1233. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6481060 [Accessed 1st December 2013].

Tudoran, R., Costan, A., Antoniu, G. & Bougé, L. (2012), A performance evaluation of azure and nimbus clouds for scientific applications, *in* 'Proceedings of the 2nd International Workshop on Cloud Computing Platforms', ACM, p. 4. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6336750 [Accessed 25th November 2013].

Vedam, V. & Vemulapati, J. (2012), Demystifying cloud benchmarking paradigm-an in depth view, *in* 'Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual', IEEE, pp. 416–421. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6340191 [Accessed 11th February 2014].

Zhan, J., Zhang, L., Sun, N., Wang, L., Jia, Z. & Luo, C. (2012), High volume throughput computing: Identifying and characterizing throughput oriented workloads in data centers, *in* 'Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International', IEEE, Washington, DC, USA, pp. 1712–1721. [Online]. Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6270846 [Accessed 7th February 2014].

Chapter 8

Appendix

8.1 Creation of execution environment

For the convenience purpose, these steps have been given below to help you installing the execution environment if needed.

8.1.1 Install g++ Compiler

We need to install "'g++"' compiler with the command as shown in listing 8.1. You might need administrative privileges for executing this command.

```
sudo apt-get install g++
```

Listing 8.1: To install g++

8.1.2 Eigen Library: setup

The tool already has Eigen core library in the folder. So there is no need to explicitly download them. However the below steps can be used to make it working.

1. Go to the Eigen library home page http://eigen.tuxfamily.org/index.php?title=Main_Page.
2. Download the Eigen 3.2.1 from the home page.
3. Extract the file and copy the Eigen folder in usr/include.
4. You can also keep the Eigen folder in the present working directly where you are executing the benchmark.