POLITECNICO DI MILANO
Scuola di Ingegneria dell'Informazione

POLO TERRITORIALE DI COMO

Master of Science in
Computer Engineering

STATISTICAL ANALYSIS OF DACAPO BENCHMARK ON AMAZON EC2

Supervisor: Prof.Marco Gribaudo

Master Graduation Thesis by: Seren Kuru
Student Id. number 759894

Academic Year 2012/2013

POLITECNICO DI MILANO
Scuola di Ingegneria dell'Informazione

POLO TERRITORIALE DI COMO

Corso di Laurea Specialistica in
Ingegneria Informatica

L'ANALISI STATISTICA DEI DACAPO BENCHMARK SU AMAZON EC2

Relatore: Prof. Marco Gribaudo

Tesi di laurea di: Seren Kuru
matr.759894

Anno Academico 2012/2013

Table of Contents

List of Figures

List of Tables

ABSTRACT

Currently, the usage of cloud computing is increasing exponentially. Accessing data and various types of applications through cloud services on internet provides flexibility. With multi-core systems becoming widespread, virtual machines now can run also in multi core systems on cloud servers. A range of core selection for different types of applications exist in order to boost performance. Interaction between some applications and multi-core processors, on the contrary, may present unpredictable performances.

My final thesis work displays experimental results of flexible eight core CPU provided by Amazon EC2, which is an IaaS provider. The objective is to describe the relation between multi-core virtual machine and applications. For that purpose, I ran some DaCapo Benchmark tests with different workloads and numbers of threads. Graphs are sketched with data collected from test results in order to interpret the relationship more accurate.

SOMMARIO

Attualmente l'uso del cloud computing sta crescendo esponenzialmente. Accedere a dati e vari tipi di applicazioni attraverso servizi cloud su internet fornisce molta flessibilità. Con la vasta diffusione dei sistemi multi-core, le virtual machine sono eseguite su questi sistemi su cloud server. Esiste una gamma di selezione dei core per diversi tipi di applicazioni per aumentare le prestazioni. L'interazione tra alcune applicazioni e processori multi-core, al contrario, può presentare prestazioni imprevedibili.

Il mio lavoro finale di tesi espone risultati sperimentali di un processore flessibile 8-core fornito da Amazon EC2, che è un provider IaaS. L'obiettivo è di descrivere la relazione tra la virtual machine multi-core e le applicazioni. Per raggiungere questo scopo, ho eseguito alcuni test DaCapo Benchmark con diversi carichi di lavoro e numero di thread. I grafici sono disegnati con i dati raccolti da i risultati dei test, per interpretare la relazione in modo più accurato.

Chapter 1

Introduction

In the beginning, when computing systems were very big computers which were able to process a small amount of data by executing a single application, that was written specifically for that hardware. In this condition there was no need to share resources among multiple applications. As the computational power of computers and the amount of applications started to increase, it was necessary to develop systems capable of sharing the available resources. Moreover, the applications have become more complex and the opportunity to parallelize some portion of them has led to the creation of multi-task applications. In this way the number of processes running in a single computer has been increased.

To face the increasing request of computational power, the major processor vendors, until recently, have tried to find a solution by increasing the frequency and the circuit complexity. This choice has given rise to problems relative to excessive heating and power consumption. Therefore, the vendors have remedied by starting developing architectures composed of multiple cores. Nowadays, a commercial personal computer is capable of handling hundreds of processes that share up to eight cores.

On the other hand, it soon became obvious that computational power of a personal computer can not be used effectively. This is because of the complex interaction between several factors that affect the performance including: application characteristics, operating system policies, and architectural characteristics. These factors had been discussed in multiple papers and theses.

In the modern world, the core of the computing infrastructure is made up of vast server farms which are called clouds, with an abundance of storage and processing cycles. Centralization of computation in these clouds, allows for pervasive access to a highly-available virtual machines with range of cores selection.

In this thesis, I focus on virtualized multi-core systems through a cloud platform. The problem approach is, performance evaluation in virtualized environment with types of applications. I executed different DaCapo Benchmarks to find relation between application characteristics on performance and impact on eight cores system.

1.1 Thesis Organization

The thesis is organized as follows.

Chapter 2 presents the key concepts that are considered necessary as a background for the comprehension of all the aspects of the thesis. It is described cloud computing, Amazon EC2, virtual machines used in clouds and DaCapo Benchmark.

Chapter 3 defines and describes experimental setup, focuses on how system prepared before starting, what were the priorities and how system automated for tests.

Chapter 4 shows experimental results of analysis conducted over the Dacapo Benchmark tests. It can be seen cumulative distribution charts, 3D graphs and box plot.

Chapter 2

Background

Having given a short overview in the introduction, this chapter revisits the key technologies and services related to this work. Its main purpose is to introduce the theoretical framework of my research work.

2.1 Definition of Cloud Computing

Nowadays, internet and new technologies are needs for businesses and individuals. In developing world, information is available anywhere and anytime. Few years ago ,traditional and only computer setup was requiring you to be in the same location as your data storage device that the cloud takes away that step.

Every day, we see in tech magazines or in any IT blogs, articles about cloud computing. There is neither a formal definition of Cloud Computing in literature nor a strict differentiation to its related concepts. When you ask five different professionals what cloud computing is, you may get five different answers. Even though not everyone agrees on what it is, there

have been many definitions of Cloud Computing by different researchers. Barkley RAD defines Cloud Computing as:

“Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what we will call a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, we call it a Public Cloud; the service being sold is Utility Computing. We use the term Private Cloud to refer to internal datacenters of a business or other organization, not made available to the general public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not include Private Clouds. People can be users or providers of SaaS, or users or providers of Utility Computing.” [1]

Cloud Computing is a term used to describe both a platform and type of application. As a platform it supplies, configures and reconfigures servers, while the servers can be physical machines or virtual machines. On the other hand, Cloud Computing describes applications that are extended to be accessible through the internet and for this purpose large data centers and powerful servers are used to host the web applications and web services [2].

To understand cloud term, let's consider your webmail service. Your email client takes care of all of the necessary hardware and software to support your personal email account. Your email does not locate in your physical computer. You access it through internet connection anywhere. This is simple example of how cloud computing works. We can propagate examples. Creating virtual albums to upload photos, running applications and storing data in servers which are located in internet. Or basically, entering a web page, beginning to use services that reside on remote and sharing confidential information. As you can see, cloud computing has popular usage among users and it is an evolving paradigm.

The cloud is a metaphor for the Internet and is an abstraction for the complex infrastructure it conceals. As something users see like a cloud but cannot see what is inside. There are some important points in the definition to be discussed regarding Cloud Computing. Cloud Computing differs from traditional computing paradigms as it is scalable, can be encapsulated as an abstract entity which provides different level of services to the clients, driven by economies of scale and the services are dynamically configurable [4]

As a result, cloud makes it possible for us to access our information anywhere at any time, and removes need of owning hardware and software to run home/business applications. However, these services are offered pay by demand or free. Users may be able to use a “pay-as-you-go” billing methodology or alternatively one where quota-limited and/or time limited access is enforced by the cloud system. Thus “billing” (or more properly user accounting) is one way to differentiate between public and private clouds. [11]

Cloud Computing infrastructure allows enterprises to achieve more efficient use of their IT hardware and software investments. Even though it is free or has a cost, this is helpful for businesses that cannot afford same amount of hardware and storage space as big companies. In the enterprise, the “adoption of Cloud Computing is as much dependent on the maturity of organizational and cultural (including legislative) processes as the technology, per se” [3] Additionally, if companies find less need of anything such as storage, higher performance hardware, they can reduce subscription or increase. Cloud computing is the delivery of computing as a service rather than a product ,whereby shared resources like audio files, videos files, data access ,software applications and storage resources are provided to PC , smart phones and tablets over the web or internet without requiring cloud users to know the location and other details of the computing infrastructure. Form of cost –efficient and flexible usage of IT services.

As a result cloud computing is broken down into three segments: "application" "storage" and "connectivity." Each segment serves a different purpose and offers different products for businesses and individuals around the world. [12]

2.2 Architecture of Cloud Computing

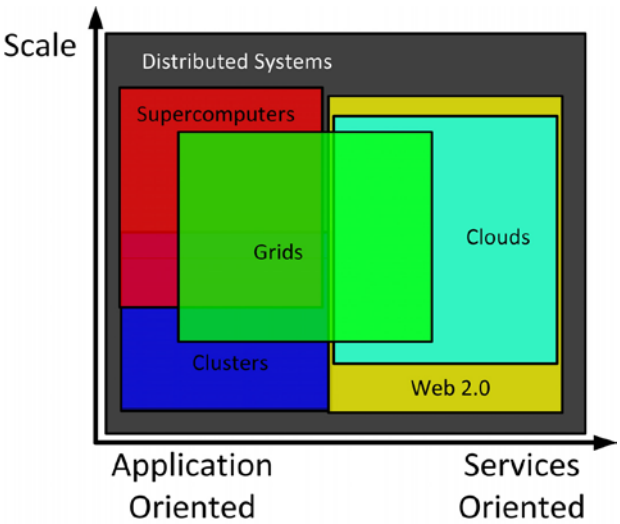


Figure 1 -Grids and Cloud Overview[6]

2.2.1 Distributed Computing

Distributed computing refers to the very idea of using distributed systems that are generally multiple computers connected to each other via computer networks to collaboratively process a common goal. Those computers communication can be homogeneous or heterogeneous, distributed globally or locally. According to the characteristics of localization or equality, distributed systems have different subsets, such as supercomputers, grids, clusters, web 2.0 and clouds. [12]

In order to facilitate a clear understanding of what exactly is Cloud computing, we compare Cloud computing with two other recent, widely-adopted or explored computing paradigms: Cluster Computing and Grid Computing.

a) Cluster Computing

"A cluster is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource." (Pfister and Buyya 2008) The resources in clusters are located in a single administrative domain and managed by a single entity. The schedulers in cluster systems focus on enhancing the overall system performance and utility as they are responsible for the whole system.

b) Grid Computing

"A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed `autonomous' resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements." (Buyya, 2002 Grid Planet Conference , San Jose, USA)

In Grid systems, resources are geographically distributed across multiple administrative domains with their own management policies and goals. , the schedulers in Grid systems called resource brokers, focusing on enhancing the performance of a specific application in such a way that its end-users' QoS requirements are met.

c) Clouds

"A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers."

Clouds seem to arrangement of clusters and grids. Clouds are appear to be next generation data centers with nodes virtualized through hypervisor technologies such VMs which is accessible as a composable service via Web Service technologies.

Essential Characteristics of Cloud Computing

As described above, there are 5 essential characteristics of Cloud Computing which explains there relation and difference from the traditional computing. [8]

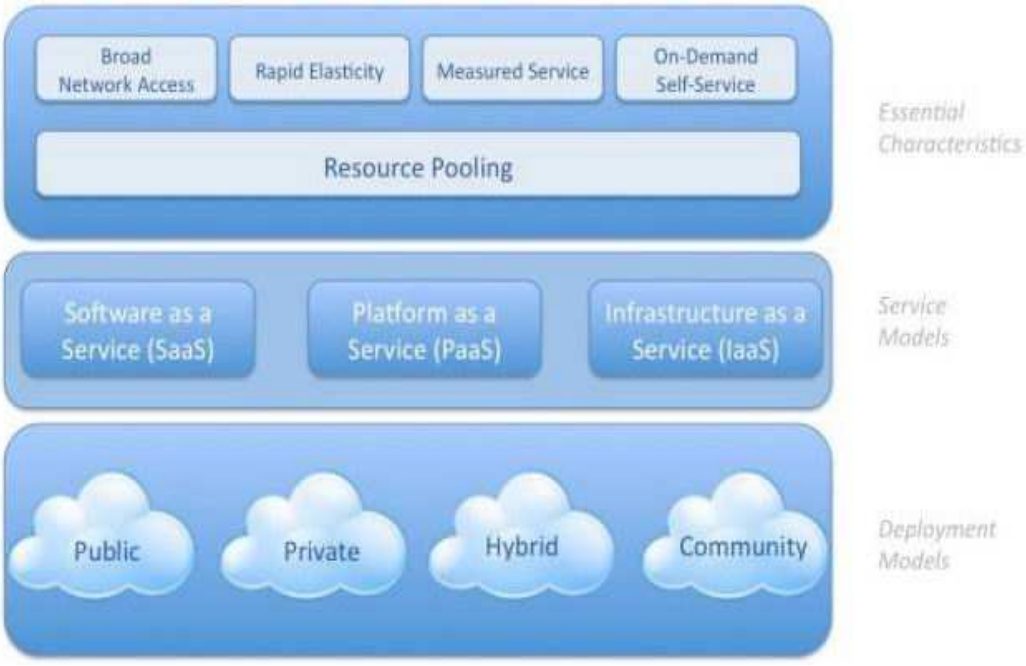


Figure 2 - Visual model of NIST Working Definition of Cloud Computing

On-demand-self-service

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service providerBroad Network

Access. It has capabilities over the network and accessed through standard mechanism.

Resource Pooling

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter)

Rapid Elasticity

Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured Service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Broad Network Access

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

2.2.2 Cloud Service Models

There are 3 Cloud Services Models and these 3 fundamental classifications are often referred to as "SPI model" i.e. software, platform or infrastructure as a service.[7]

a) Cloud Software as a Service

Software as a service, or SaaS, is probably the most common type of cloud service development. With SaaS, a single application is delivered to thousands of users from the vendor's servers. Customers don't pay for

owning the software; rather, they pay for using it. Users access an application via an API accessible over the web. Each organization served by the vendor is called a tenant, and this type of arrangement is called a multitenant architecture. The vendor's servers are virtually partitioned so that each organization works with a customized virtual application instance. For customers, SaaS requires no upfront investment in servers or software licensing. For the application developer, there is only one application to maintain for multiple clients. Many different types of companies are developing applications using the SaaS model.

As an example the so-called Google Apps offer software for business or private entities online that can do the fundamental business action that a usual on-premise office suite can provide. Google Apps involve document collaboration within text documents, presentation and spreadsheets as much as calendars and e-mail services.[17]

b) Cloud Platform as Service

In this variation of SaaS, the development environment is offered as a service. The developer uses the "building blocks" of the vendor's development environment to create his own custom application. It's kind of like creating an application using Legos; building the app is made easier by use of these predefined blocks of code, even if the resulting app is somewhat constrained by the types of code blocks available.

Force.com is an example of platform as a service offered from Salesforce providing a development platform that makes it very easy for developers to build multi-tenant applications. The applications run on the data centers of Salesforce, so there is no necessity to take care of maintenance, security and back-ups.[17]

c) Cloud Infrastructure as Service

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components. Specifically, it provides an environment in which external users can obtain exclusive access to raw metered hardware nodes in an on-demand fashion, similar to obtaining VMs from an IaaS provider. The system allows the software to be loaded and network connectivity to be under user control. [8]

Amazon Web Services is one example of that, where infrastructure is available on a pay-per-use self service basis and get servers, storage,

network configuration, set all that up and run it, while not having to worry about co-location, rental or datacenters.

2.2.3 Cloud Deployment Models

Private cloud

The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Community cloud

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Public cloud

The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Hybrid cloud

The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

2.3 Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud, also known as "EC2", is a commercial web service which allows paying customers to rent computers to run computer applications on. It presents virtual computing environment and allows scalable deployment of applications by providing a web services interface through which customers can request an arbitrary number of Virtual Machines, i.e. server instances, on which they can load any software of their choice.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Current users are able to create, launch, and terminate server instances on demand, hence the term "elastic". Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change.

An Amazon Machine Image (AMI) is an encrypted file stored in Amazon S3. It contains all the information necessary to boot instances of your software. Your AMIs are your unit of deployment. You might have just one AMI or you might compose your system out of several building block AMIs (e.g., webservers, appservers, and databases). Amazon EC2 provides a number of command line tools to make creating an AMI easy. Once you create a custom AMI, you will need to upload it to Amazon S3. Amazon EC2 uses Amazon S3 to provide reliable, scalable storage of your AMIs so that they can boot them when you ask them to do so. You can also choose from a library of globally available AMIs that provide useful instances. For example, if you just want a simple Linux server, you can choose one of the standard Linux distribution AMIs. Once you have set up your account and uploaded your AMIs, you are ready to boot your instance.

The running system based on an AMI is referred to as an instance. All instances based on the same AMI begin executing identically. Any information on them is lost when the instances are terminated or if they fail.

The Amazon implementation allows server instances to be created in zones that are insulated from correlated failures. EC2 is one of several Web Services provided by Amazon.com under the umbrella term Amazon Web Services (AWS).

EC2 uses Xen Virtualization. Each virtual machine, called an instance, is a virtual private server and can be one of several sizes. Instances are sized based on EC2 Compute Units which is the equivalent CPU capacity of physical hardware. In addition, servers in EC2—like any other server on the Internet—can access Amazon S3 for cloud-based persistent storage. EC2 servers in particular see both cost savings and greater efficiencies in accessing S3. To secure your network within the cloud, you can control virtual firewall rules that define how traffic can be filtered to your virtual nodes. You define routing rules by creating security groups and associating the rules with those groups.

Amazon's EC2 U.S. footprint spans three data centers on the East Coast of the U.S. and two in Western Europe. You can sign up separately for an Amazon European data center account, but you cannot mix and match U.S. and European environments. The servers in these environments run a highly customized version of the Open Source Xen hypervisor using paravirtualization. This Xen environment enables the dynamic provisioning and deprovisioning of servers, as well as the capabilities necessary to provide isolated computing environment for guest servers. When you want to start up a virtual server in the Amazon environment, you launch a new node based on a predefined Amazon machine image (AMI). The AMI includes your operating system and any other prebuilt software. Most people start with creating a standard AMI containing their applications, libraries, data and associated configuration settings. Or they use pre-configured, templated images to get up and running immediately. Then, they upload the AMI into Amazon S3. Amazon EC2 provides tools that make storing the AMI simple. Amazon S3 provides a safe, reliable and fast repository to store images. Amazon EC2 web service should be used to configure security and network access. Start, terminate, and monitor as many instances of your AMI as needed, using the web service APIs. Pay only for the resources that you actually consume, like instance-hours or data transfer. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use.

Many competitors to Amazon also provide persistent internal storage for nodes to make them operate more like a traditional data center.

2.3.1 Service Highlights

• Elastic

Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds or even thousands of server instances simultaneously. Of course, because this is all controlled with web service APIs, your application can automatically scale itself up and down depending on its needs.

• Completely Controlled

You have complete control of your instances. You have root access to each one, and you can interact with them as you would any machine. Instances can be rebooted remotely using web service APIs. You also have access to console output of your instances.

• Flexible

You have the choice of several instance types, allowing you to select a configuration of memory, CPU, and instance storage that is optimal for your application.

• Designed for use with other Amazon Web Services

Amazon EC2 works in conjunction with Amazon Simple Storage Service (Amazon S3), Amazon SimpleDB and Amazon Simple Queue Service (Amazon SQS) to provide a complete solution for computing, query processing and storage across a wide range of applications.

• Reliable

Amazon EC2 offers a highly reliable environment where replacement instances can be rapidly and reliably commissioned. The service runs within Amazon's proven network infrastructure and datacenters.

• Features for Building Failure Resilient Applications

Amazon EC2 provides powerful features to build failure resilient applications including:

o Multiple Locations

Amazon EC2 provides the ability to place instances in multiple locations. Amazon EC2 locations are composed of regions and Availability Zones. Regions are geographically dispersed and will be in separate geographic areas or countries. Currently, Amazon EC2 exposes only a single region. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same region. Regions consist of one or more Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from failure of a single location.

o Elastic IP Addresses

Elastic IP addresses are static IP addresses designed for dynamic cloud computing. An Elastic IP address is associated with your account not a particular instance, and you control that address until you choose to explicitly release it. Unlike traditional static IP addresses, however, Elastic IP addresses allow you to mask instance or Availability Zone failures by programmatically remapping your public IP addresses to any instance in your account. Rather than waiting on a data technician to reconfigure or replace your host, or waiting for DNS to propagate to all of your customers, Amazon EC2 enables you to engineer around problems with your instance or software by quickly remapping your Elastic IP address to a replacement instance.

• Secure

Amazon EC2 provides web service interfaces to configure firewall settings that control network access to and between groups of instances.

• Inexpensive

Amazon EC2 passes on to you the financial benefits of Amazon's scale. You pay a very low rate for the compute capacity you actually consume. Compare this with the significant up-front expenditures traditionally required to purchase and maintain hardware, either in-house or hosted. This frees you from many of the complexities of capacity planning, transforms what are commonly large fixed costs into much smaller variable costs, and removes the need to over-buy "safety net" capacity to handle periodic traffic spikes.

2.3.2 Selecting Instance Types

Amazon EC2 instances are grouped into two families: Standard and High-CPU. Standard Instances have memory to CPU ratios suitable for most general purpose applications; HighCPU instances have proportionally more CPU resources than memory (RAM) and are well suited for compute-intensive applications. When choosing instance types, you should consider the characteristics of your application with regards to resource utilization and select the optimal instance family and size. One of the advantages of EC2 is that you pay by the instance hour, which makes it convenient and inexpensive to test the performance of your application on different instance families and types. One good way to determine the most appropriate instance family and instance type is to launch test instances and benchmark your application.

2.3.3 I/O Performance

Amazon EC2 provides virtualized server instances. While some resources like CPU, memory and instance storage are dedicated to a particular instance, other resources like the network and the disk subsystem are shared among instances. If each instance on a physical host tries to use as much of one of these shared resources as possible, each will receive an equal share of that resource. However, when a resource is under-utilized user will often be able to consume a higher share of that resource while it is available. The different instance types provide higher or lower minimum performance from the shared resources depending on their size. Each of the instance types has an I/O performance indicator (moderate or high). Instance types with high I/O performance have a larger allocation of shared resources. Allocating larger share of shared resources also reduces the variance of I/O performance. For many applications, moderate I/O performance is more than enough. However, for those applications requiring greater or more consistent I/O performance, users may want to consider instances with high I/O performance.

2.4 Virtual Machines

Most cloud service providers use machine virtualization techniques to provide flexible and cost-effective resource sharing among users. The Xen hypervisor , sometimes referred to generically as x86 a virtual machine monitor, is an open-source software program that coordinates the low-level interaction between virtual machines and physical hardware. A Xen-based virtual machine consists of the following components:

- At least one virtual disk that contains a bootable operating system. The virtual disk can be based on a file, partition, volume, or other type of block device.
- Virtual machine configuration information, which can be modified by exporting a text-based configuration file from xend or through Virtual Machine Manager.
- A number of network devices, connected to the virtual network provided by the controlling domain.[18]

Each virtual machine runs an instance of an operating system. A scheduler is running in the Xen hypervisor to schedule virtual machines on the processors. The original Xen implementation schedules virtual machines according to the Borrowed Virtual Time (BVT) algorithm.

Particularly for network virtualization, Xen only allows a special privileged virtual machine called driver domain, or domain 0 to directly control the network devices. All the other virtual machines (called guest domains in Xen) have to communicate through the driver domain to access the physical network devices. The way Xen realizes this is, the driver domain has a set of drivers to control the physical Network Interface Cards (NIC), and a set of back-end interfaces to communicate with guest domains. The back-end interfaces and physical drivers are connected by a software bridge inside the kernel of the driver domain. Each guest domain has a customized virtual interface driver to communicate with a back-end interface in the driver domain. All the packets sent from guest domains will be sent to the driver domain through the virtual interfaces and then sent into the network. All the packets destined to a guest domain will be received by the driver domain first, and then transferred to the guest domain.

Amazon Elastic Compute Cloud EC2 , provides a number of virtual machines based on a Xen hypervisor that manages physical servers and

provides a selection of virtual machine types: small, consisting of one virtual CPU core; large, containing two CPU cores: and extra large, with four cores.

There might be several Xen virtual machines running on one physical server. Each Xen virtual machine is called an instance in Amazon EC2. There are several types of instances. Each type of instance provides a predictable amount of computing capacity. The small instance is the primary instance type, which is configured with 1.7GB memory, 1 EC2 compute unit and 160GB instance storage. According to Amazon, ”one EC2 compute unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.” For applications requiring higher computing capacity, Amazon EC2 provides several high-capacity instances which are configured with 4 to 20 EC2 compute units. The input-output (I/O) capacities of these types of instances are not specified clearly.

Allocated EC2 instances can be placed at different physical locations. Amazon organizes the infrastructure into different regions and availability zones. There are two regions, useast-1 and eu-west-1, which are located in the US and in Europe respectively. Each region is completely independent and contains several availability zones that are used to improve the fault tolerance within the region. It is suspected that each availability zone is an isolated data center which is powered by its own powerline. Different availability zones in the same region are placed very close to each other. The region useast-1 has three availability zones, us-east-1a, us-east-1b and us-east-1c. The region eu-west-1 has two availability zones, eu-west-1a and eu-west-1b.

The use of virtual machine technology for cloud computing decouples the number of virtual and physical CPUs. There is a practical upper bound on the number of virtual machines and virtual processors that can be deployed on a server system. If a cloud computing service is oversubscribed, the system will become slow and unreliable, and new requests for service will be denied. Conversely, if the system is undersubscribed, the costs of maintaining unutilized capacity will be unrecoverable over time, and reduce profits.

2.5 Dacapo Benchmark

Benchmarks are typically used to evaluate new system features and optimizations after explored by researchers. If an idea doesn't produce a set of interesting benchmarks, community is unlikely to accept it.

In academia or industry, SPEC Java benchmarks are typically used for Java. When SPEC introduced these benchmarks, their evaluation rules and the community's evaluation metrics glossed over some of the key questions for Java benchmarking. For example, SPEC reporting of the "best" execution time is taken from multiple iterations of the benchmark within a single execution of the virtual machine, which will typically eliminate compile time. In addition to steady state application performance, a key question for Java virtual machines (JVMs) is the tradeoff between compile and application time, yet SPEC does not require this metric, and the community often does not report it. Also, SPEC does not require reports on multiple heap sizes and thus does not explore the spacetime tradeoff automatic memory management (garbage collection) must make.

Also to SPEC, prior Java benchmark suits include Java Grande, Jolden and Ashes. The Java Grande Benchmarks include programs with large demands for memory, bandwidth, or processing power. They focus on array intensive programs that solve scientific computing problems. The programs are sequential, parallel, and distributed. They also include microbenchmark tests for language and communication features, and some cross-language tests for comparing C and Java. DaCapo also focuses on large, realistic programs, but not on parallel or distributed programs. The DaCapo benchmarks are more general purpose, and include both client and server side applications. The Jolden benchmarks are single-threaded Java programs rewritten from parallel C programs that use dynamic pointer data structures. These programs are small kernels (less than 600 lines of code) intended to explore pointer analysis and parallelization, not complete systems. The Soot project distributes the Ashes benchmarks with their Java compiler infrastructure, and include the Jolden benchmarks, a few more realistic benchmarks such as their compiler, and some interactive benchmarks. The DaCapo benchmarks contain many more realistic programs, and are more ambitious in scope.

For Dacapo benchmarking efforts, start was given in mid 2003, when Dacapo research group decided that the existing Java benchmarks were limiting the progress. This benchmark suite is intended as a tool for Java benchmarking by the programming language, memory management and

computer architecture communities. It consists of a set of *open source*, *real world* applications with non-trivial memory loads. The initial release of the suite was the culmination of over five years work at eight institutions, as part of the Dacapo Benchmark Project, which was funded by a National Science Foundation ITR Grant.

It was followed with effort of identifying suitable benchmarks and develop a suite of analyses to characterize candidate benchmarks and evaluate them for inclusion. It started with the following criteria;

1. *Diverse real applications.*It was required applications that were widely used to provide a compelling focus for the community's innovation and optimizations, as compared to synthetic benchmarks.

2. *Ease of use. A*pplications to be relatively easy to use and measure.
3.*Responsive.* Adapt the suite as circumstances change.

4.*Open source*. Encourage community feedback and enable analysis of benchmark sources.

These criteria was implemented as follows.

1. Only *open source* benchmarks and libraries was chosen.

2. Diverse programs to *maximize coverage* of application domains and application behaviors was chosen.

3. On the client-side benchmarks that are *easy to measure* in a completely standard way, was focused with minimal dependences outside the scope of the host JVM.

4. GUI applications was excluded since they are difficult to benchmark systematically. In the case of *eclipse*, exercise a non-GUI subset.

5. Range of inputs was provided. With the default input sizes, the programs *are timely* enough that it takes hours or days to execute thousands of invocations of the suite, rather than weeks. With the exception of *eclipse*, which runs for around a minute, each benchmark executes for between 5 and 20 seconds on contemporary hardware and JVMs.

The main contributions of research are new, more realistic Java benchmarks, an evaluation methodology for developing benchmark suites, and performance evaluation methodologies. Needless to say, the DaCapo benchmarks are not definitive, and they may or may not be representative of workloads that vendors and clients care about most.

Around 20 students and faculty at six institutions had began an iterative process of identifying, preparing, and experimenting with candidate benchmarks. To realize the difficulty of identifying a good benchmark suite, it was made the DaCapo benchmark project open and transparent, inviting feedback from the community. As part of this process, it had been released three beta versions.

It was systematically analyzed by each candidate to identify ones with non-trivial behavior and to maximize the suite's coverage.It was included most of the benchmarks evaluated, excluding only a few that were too trivial or whose license agreements were too restrictive, and one that extensively used exceptions to avoid explicit control flow.

DaCapo 9.12 Benchmark Suite has a range of multithreaded benchmarks. The following table will show the nature of the threading in the some of existing benchmarks and their definition.

Benchmarks	Description	Threading
Avrora	Simulates a number of programs run on a grid of AVR microcontrollers.	Driven by a single external thread, but it is internally multithreaded with each simulated element using a thread (i.e. each node in a grid of simulated nodes is threaded). Avrora demonstrates a high volume of fine granularity interactions between simulator threads.
Eclipse	Executes some of the (non-gui)jdt performance tests for the Eclipse IDE.	Driven by a single external thread it is internally multithreaded. However, some worker thread activity seems to be serialised, while others seem to engage in some fine granularity interactions. As such, eclipse exhibits periods of little concurrency and brief periods of moderate granularity concurrency.
Fop	Takes an XSL-FO file, parses it and formats it,generating pdf file.	
H2	Executes a JDBCbench -like in-memory benchmark, executing a number of transactions against a model of a banking application, replacing the hsqldb benchmark	Multithreaded, it is driven by one client thread per hardware thread and internally has a server thread for each client thread as well as other support threads. The number of client threads for the default benchmark size is set a one per hardware thread.
Jython	Interprets a the pybench Python benchmark.	Driven by a single thread. Internally it uses one thread per hardware thread, but the bulk of the Jython tests are single threaded.
Pmd	Analyzes a set of Java classes for a range of	Driven by a single client thread it is internally multithreaded using one worker thread per hardware

	source code problems.	thread.
Tomcat	Runs a set of queries against a Tomcat server retrieving and verifying the resulting web pages.	Multithreaded, driven by a client thread per hardware thread, each client thread performing a series of requests which are dealt with by a server thread for that client thread.

Table 1 - DaCapo Benchmarks [16]

Chapter 3

Experimental Setup

Experimental setup chapter indicates operations which were completed during preparation of system, execution of codes and collection of results. Virtual machine had been chosen and then arranged according to test requirements with latest release DaCapo benchmark. Later test codes had been executed and collection of results had been collected.

3.1 Preparing System

Prof. Piazzolla had created credential that grants me an access to Amazon Virtual machines. Since instances require hourly cost, it was utilized DEI's budget for my performance tests. After it had chosen region of server (US East N. Virginia), Amazon Linux AMI 2013.03.1 was selected for future instance types. Amazon Linux AMI which was 64 bit, includes Linux 3.4, AWS tools and Tomcat. Instance type for our tests were chosen among "on demand" and "general purposed" instances which require hourly costs to be used. For DaCapo tests, it was demanded to run tests on eight cores machine.

After M3 Double Extra Large (m3.2xlarge) had been chosen as eight cores machine (hourly cost/$1.00), instance required a new key pair. There is an option without a key pair. However user will not be able to connect to his/her instance unless s/he knows the password built in to that AMI. On the other hand, public/private key pairs allow users to securely connect to user's instance after it launches. For Linux server instances, a key pair allows user to SSH into user's instance. Then new key pair was downloaded. (.pem file) The rest had been followed as in the default configuration. When instance setup had been completed, it began to start.

To reach virtual machine, two more programs which are called puttygen.exe and putty.exe, were needed to be configured. PuTTY key generator (puttygen.exe), generates pairs of public and private keys to be used in PuTTY. Already downloaded .pem file was imported into PuTTYgen, then it was saved as private key to be used in future instance launches. On the other hand PuTTY is a free SSH client for Windows systems. After private key had been obtained, its file location was used to be indicated for file authentication in PuTTY's SSH Auth section. Instance always starts with a unique public DNS which is used for hostname section in PuTTY, to create connection between terminal and virtual machine. After it had been connected and verified the server's host key, it was asked to login with a username which was ec2-user. When client was connected to server, if there was required updates, they were shown and applied. Finally, system had been ready for installation of DaCapo Benchmark.

FileZilla client is a free and cross platform FTP software. In virtual machine, it had been used to create and arrange directories, files and folders. Two main folders, one for DaCapo benchmark jar file and the other had been for test results. Latest DaCapo benchmark is DaCapo-9.12-bach.jar file which was released in 2009 and used in tests. It had been installed via "*wget[URL]"* command into virtual machine. Other test bash files had been transferred via FileZilla since they had been small files. Then bash files had been converted to executable files with "*chmod A + X filename* " command.

3.2 Bash Scripts

In Linux, a .sh file is a shell script file that has been written using the BASH language. The shell's interface features a command line that you can enter text instructions into. Bash scripts had been used to achieve performance tests for eight cores virtual machine. After scripts had been uploaded and become executable, main.exe run with manually set number of threads active. In this headline, I would like to describe which features of bash scripts had been used for test results.

Tests had been done automatic. First, test environment had been prepared by setting active CPU cores. Then, Dacapo benchmarks need to start with some parameters. They are number of threads, N-workload and number of iteration. After execution of benchmark had been completed, bash script had collected the response time of each instances discarding the values that did not satisfy the experiment assumptions. In particular, another bash script had removed the initial and final transient of the

experiments when the number of concurrent instances was not constant. In the bash scripts, it had been used "awk utility" as a data extraction and reporting utility.

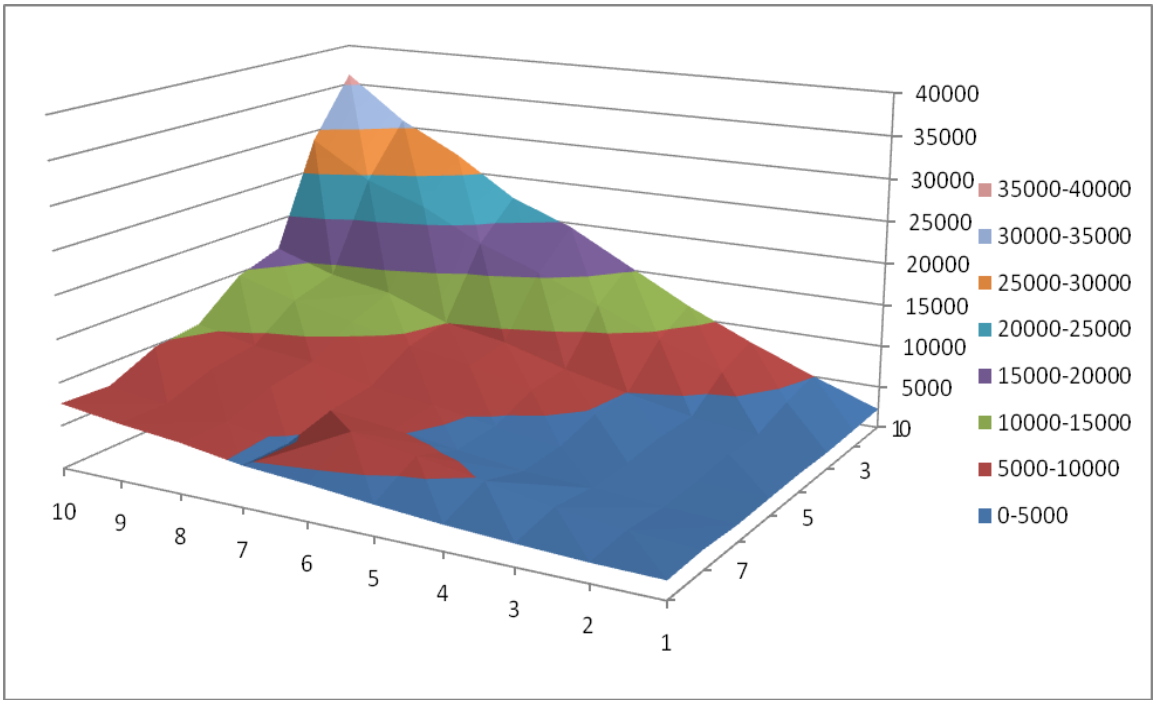
Chapter 4

Experimental Results

In this section, I describe the results obtained from the executions of the DaCapo Benchmarks in a controlled in environment of eight core Amazon EC2. Since experiment's results had been too much, I filtered them into result sets which had been done with thread one, to work on them. Four charts had been chosen among the others.

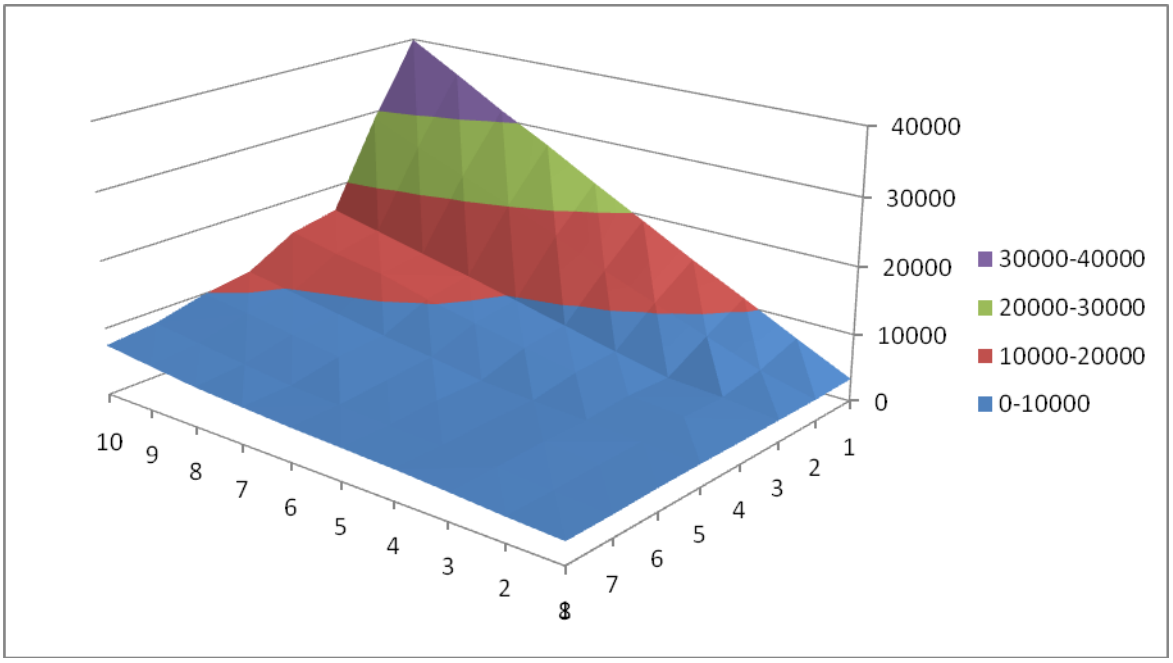
4.1 3-D Surface Charts

First group of experimental results, shows us levels of average execution time which had been produced by each activated core with different workloads . Before drawing surface charts, data had been grouped under core numbers with values of average time. X-axis has numbers from one to ten, they represent number of workloads. Y- axis has range of average time. Z- axis shows which cored had been active.



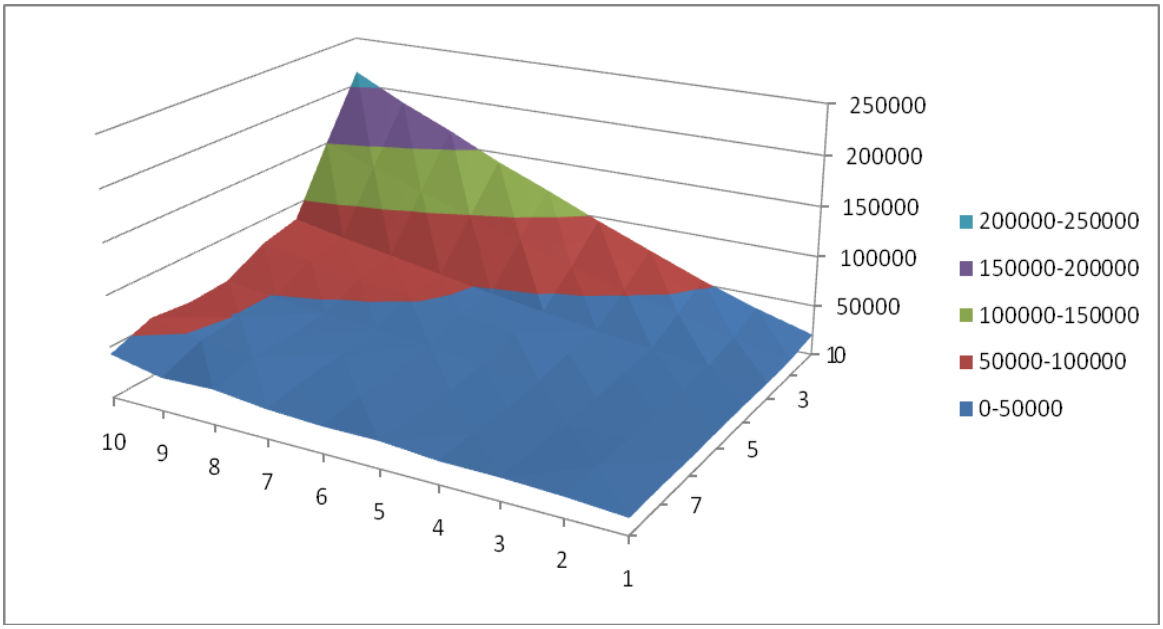
Graph 1 - 3D Jython Benchmark

As expected when one core had been active, it had consumed longest time, when compared with other cores. In core two and core three, we see exponential increase in average of time. However, between core six and core seven, there is a visible performance decrease. Core seven from workload one to six, falls behind core six. Core eight is the fastest. When we look at general picture, average time differences between cores and workloads are close each other which produces small scaled legend to show details in chart .



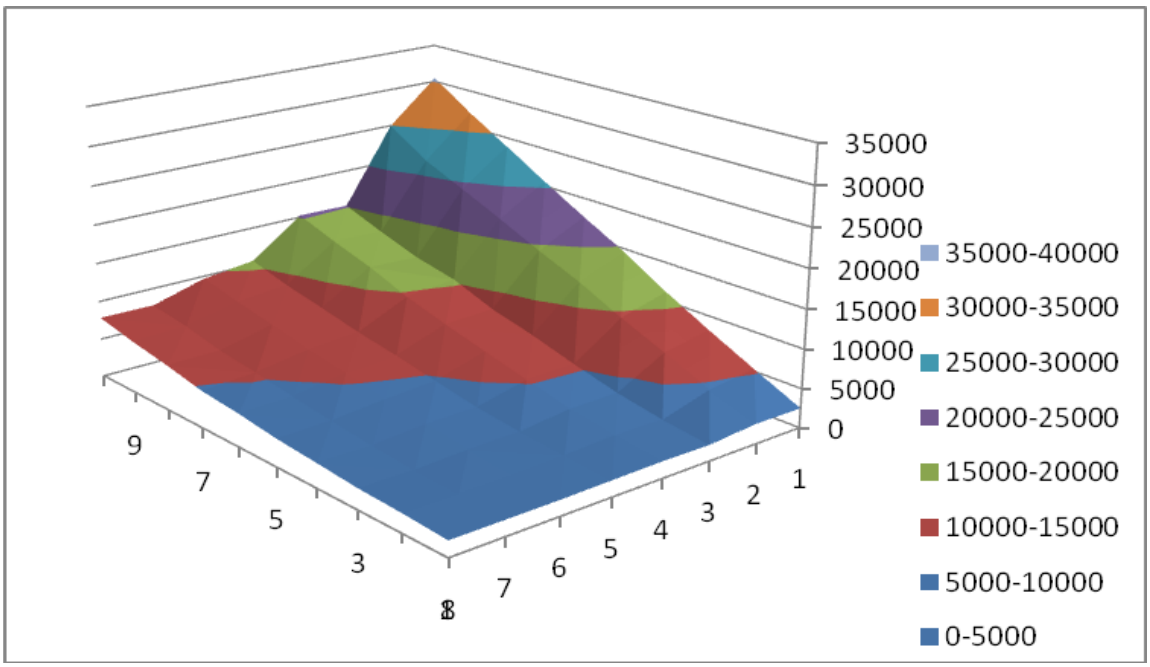
Graph 2 - 3D H2 Benchmark

When we compare with Jython benchmark, we can't see small ranges in the legend. In the beginning, it shows slow performance to H2 benchmark but in the end with eight core and workload ten, it is faster. Between core three and core four, between core five and core six, between core seven to core eight, till workload six, these couples' average times are really close to each other. The difference between them starts with workload six.



Graph 3 - 3D Eclipse Benchmark

Eclipse benchmark shows highest average of time which is the result of slow work. After core two, there is a dramatic increase in time. Core five stays slower until workloads nine and ten but catches core six and seven. On the other hand, core six and core seven returns response times almost equal. However, core eight still owns lowest average time.



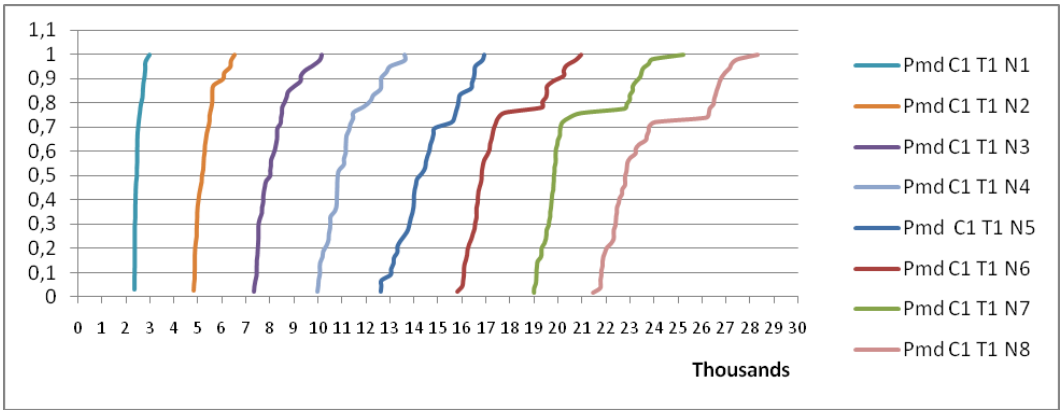
Graph 4 - 3D Pmd Benchmark

Pmd benchmark shows similar characteristics with Jython benchmark. After core two there is a crucial decrease in time almost half to half. Core eight and seven, gives results with nearly same average of time.

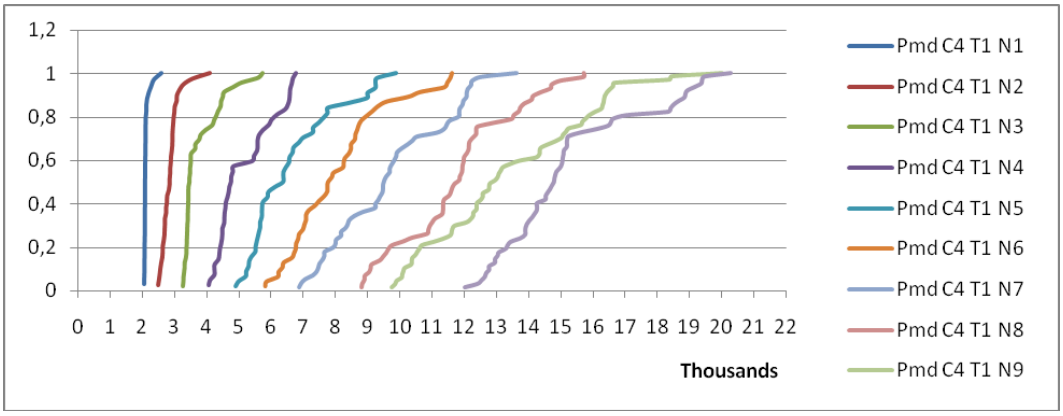
4.2 N-Workload Difference

Six example charts had been chosen for demonstration. These charts represent cumulative distribution function. First three of charts have 1 core, other three have 4 cores. Each curve represents the behavior of the selected benchmark run with a given cores configuration with an increasing number *N* of concurrent benchmark instances. X-axis gives response times of benchmark in a increasing order. Y-axis is the mean of number of times, to have correct progression of values over axis. As expected, when the workload N increases, response time is increasing too.

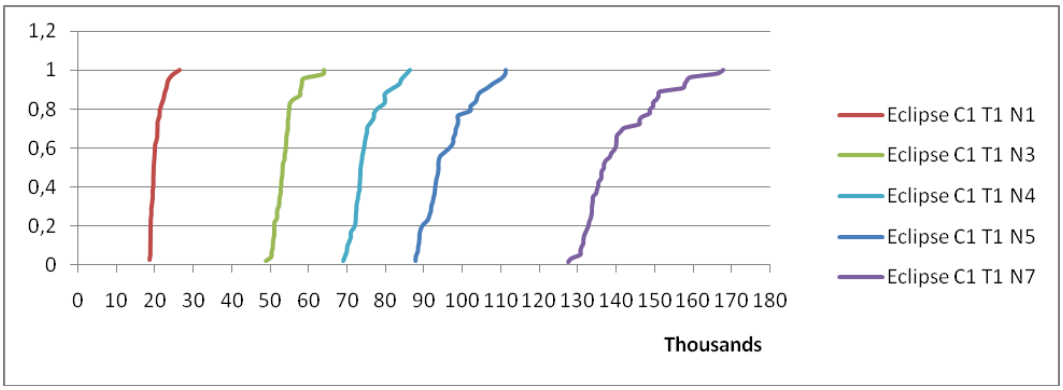
Pmd, Eclipse and H2 have a stable response time R in core 1. Also, for Eclipse and H2, when 4 cores are activated, they maintain stable response time R as long as N is below or equal to 4. On the other hand, Pmd does not behave this way. For pmd, response time starts to increase even when N is below the number of core. Probably this is due to the workload characteristics of the benchmark which includes both CPU and I/O operations.[20]



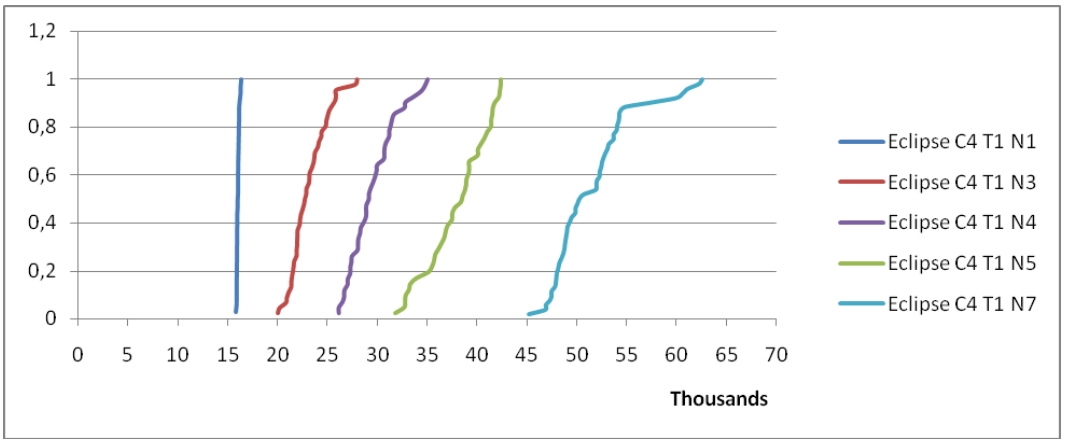
Graph 5 - 1 Core and Pmd



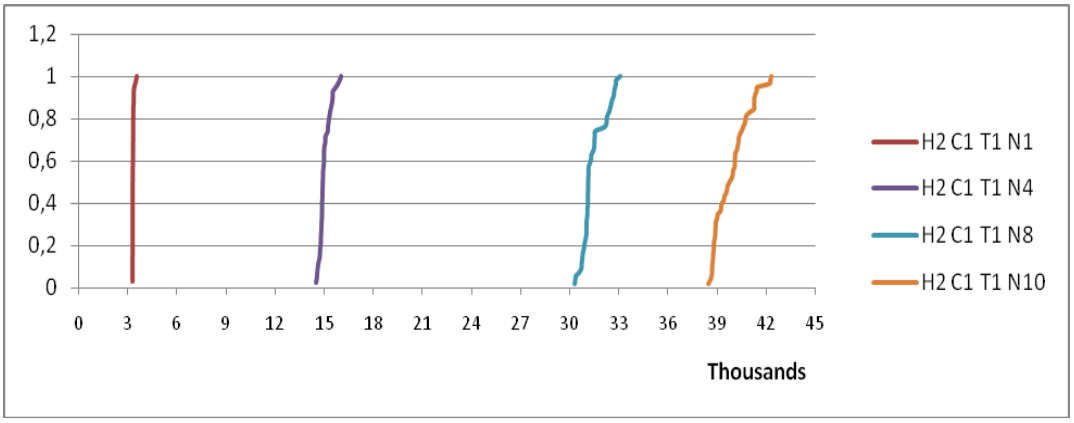
Graph 6 - 4 Cores and Pmd



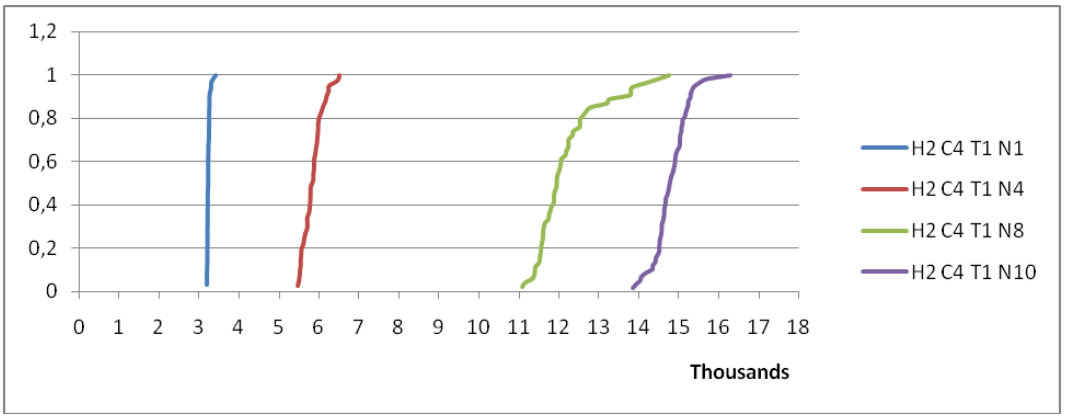
Graph 7 - 1 Core and Eclipse



Graph 8 - 4 Cores and Eclipse



Graph 9 - 1 Core and H2



Graph 10 - 4 Cores and H2

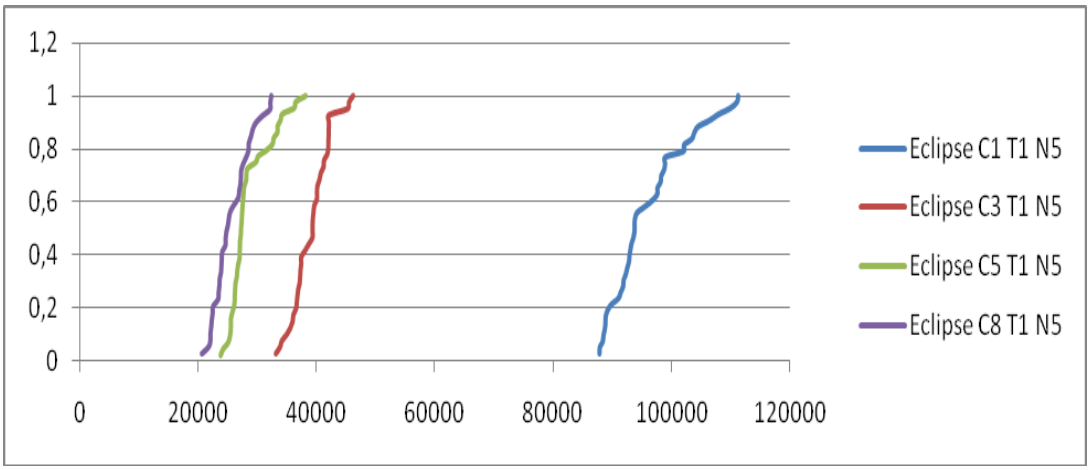
4.3 CPU Allocation

In this section, I try to evaluate benchmarks with different cores. I started by fixing number of basic iterations of each load. For N5 and N10,respectively iteration number was 13 and 10, regardless of the considered benchmarks. In this way all benchmarks had run with same amounts of workload but with different response time. This measured response time gives performance of CPU. Results are shown in figures 13,14,15.

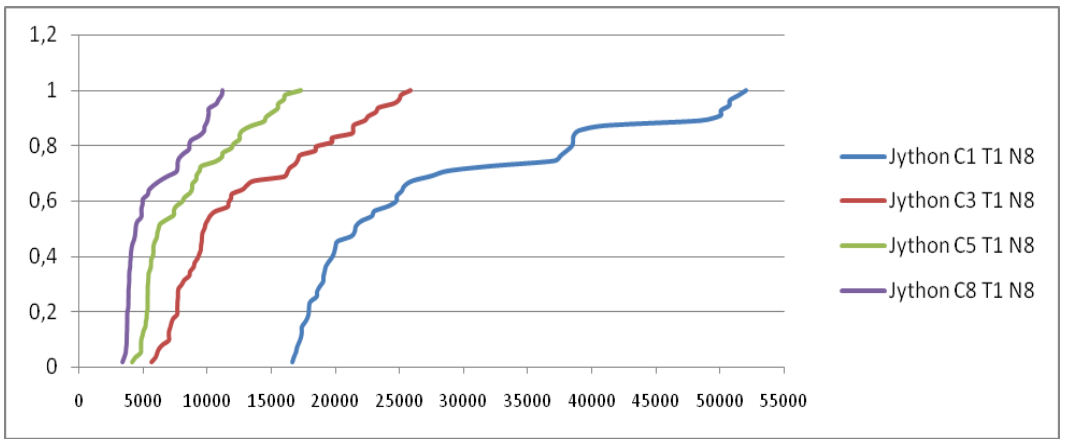
As expected, core 1 have the highest response time and core 8 has the lowest. In Eclipse, Jython and Pmd, response time difference between core 1 and core 3 is high. Also it is obvious that spread of lines narrow from core 1 to core 8 and results are more stable. In this case, I assume each core handles at most one of the N benchmark instances.

I have to take into consideration threads of each application. Since Pmd and Eclipse are driven by a single external thread which is internally

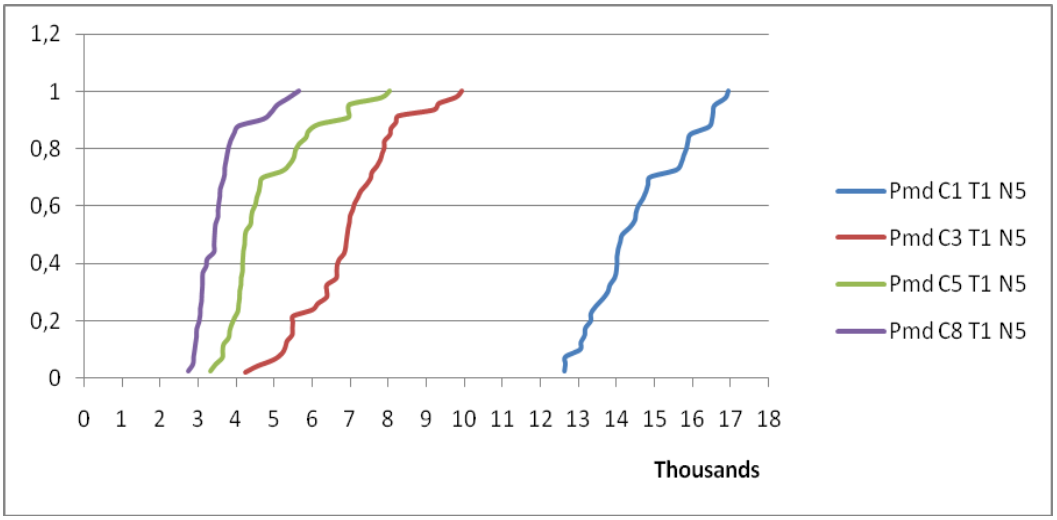
multithreaded, software parallelization is possible and charts show more stable response time for them. However, Jython is single threaded and response time increase even N is below the number of cores.



Graph 11 - Eclipse and multicore processor



Graph 12 - Jython and multicore processor



Graph 13 - Pmd and multicore processor

4.4 Box Plot with Mean

This chapter shows results using box and whisker plot. X-axis shows number of cores. Y-axis shows response time. Each core was tested with equal number of N loads which is from 1 to 10. These charts will give an overview of virtual machine performance for each benchmark.

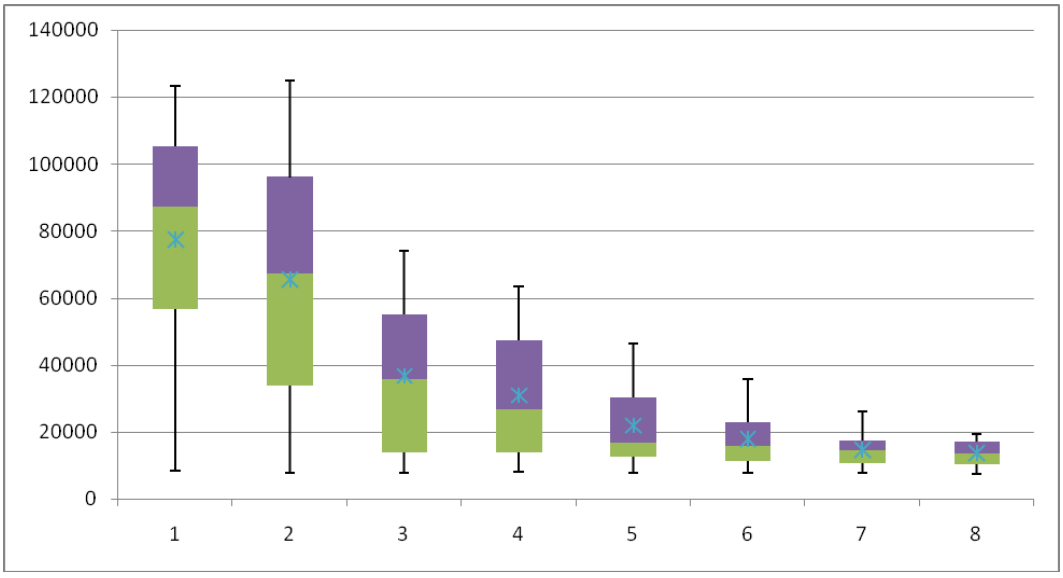


Figure 3 - Avrora Box Plot

Avrora has less improvement when passing from single core to double core. In core 2 whiskers have same length as the box, so data is compact but least stable. Since core 2 box is the largest, variance is more than others. Except core 7 and 8, other cores have long upper whiskers, their median slips down and mean value stands on upper quartile. I assume, these cores have high response time but they get more stable with ascending number of cores. On the other hand, core 7 is stable in response time but it still gives high response time. Core 8 behaves well balanced and stable.

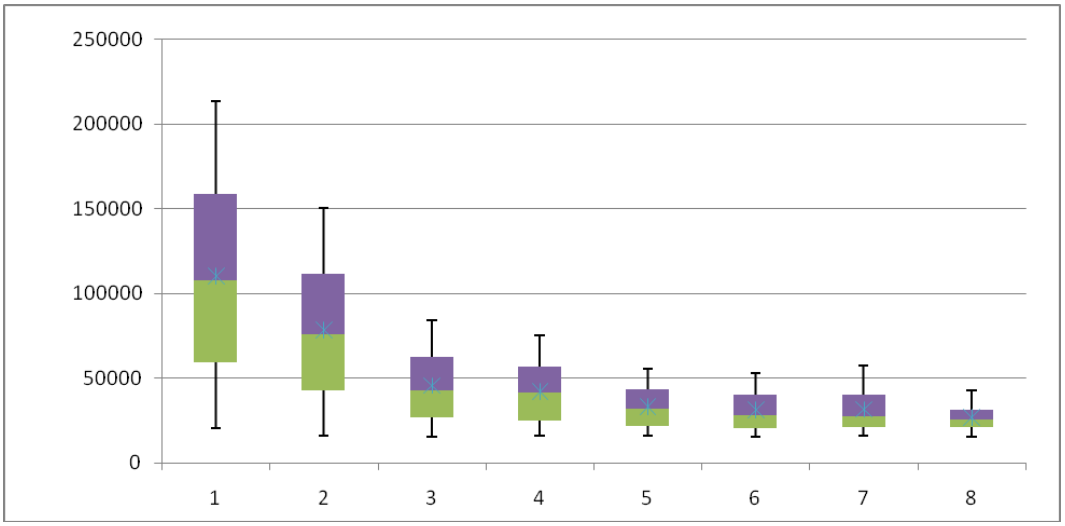


Figure 4 - Eclipse Box Plot

Eclipse was executed with high response time among the other benchmarks. Core 1 is wide with two long whiskers. I interpret core 1 as well-balanced with high variance. From core 1 to core 3, I see rational fall in response time and begin to become more stable. There is a slight performance difference between core 3 and core 4 which is faster. Core 5 has longer lower quartile than core 6 and shorter upper whisker than core 7. I suppose in general, core 5 gives low response time than core 6 and 7 and works faster for eclipse benchmark. Core 8 is narrow but data varies.

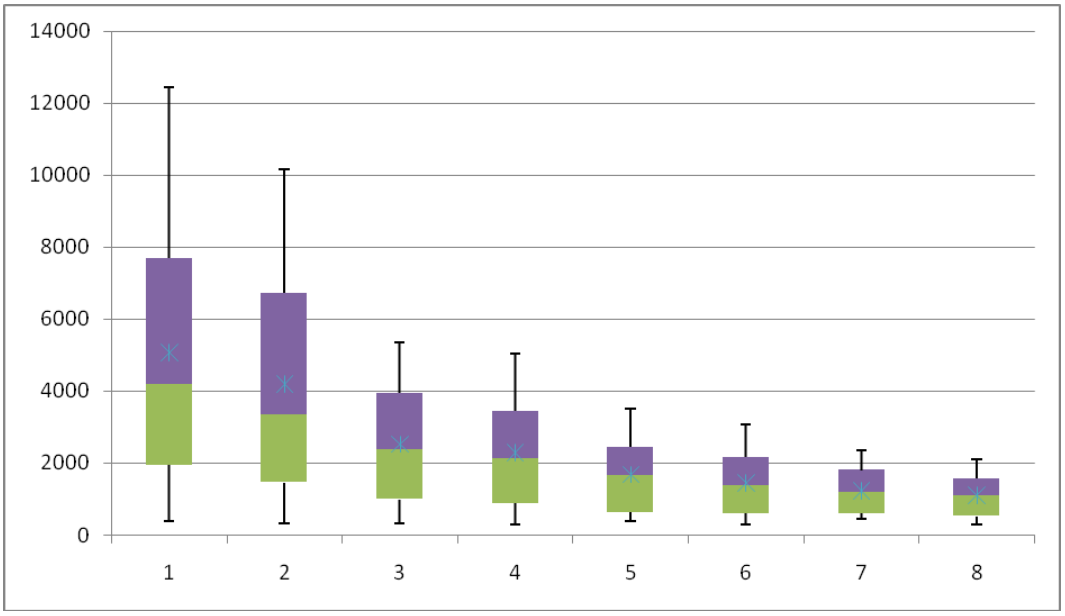


Figure 5 - Fop Box Plot

In the first two cores, upper quartiles involve mean of response time and they are skewed right. So I can assume performance of both cores is slow and response times are least stable. With core 3, an exponential decrease in response time begins till core 8 and boxes get narrow. Upper and lower whiskers get short and it becomes well balanced.

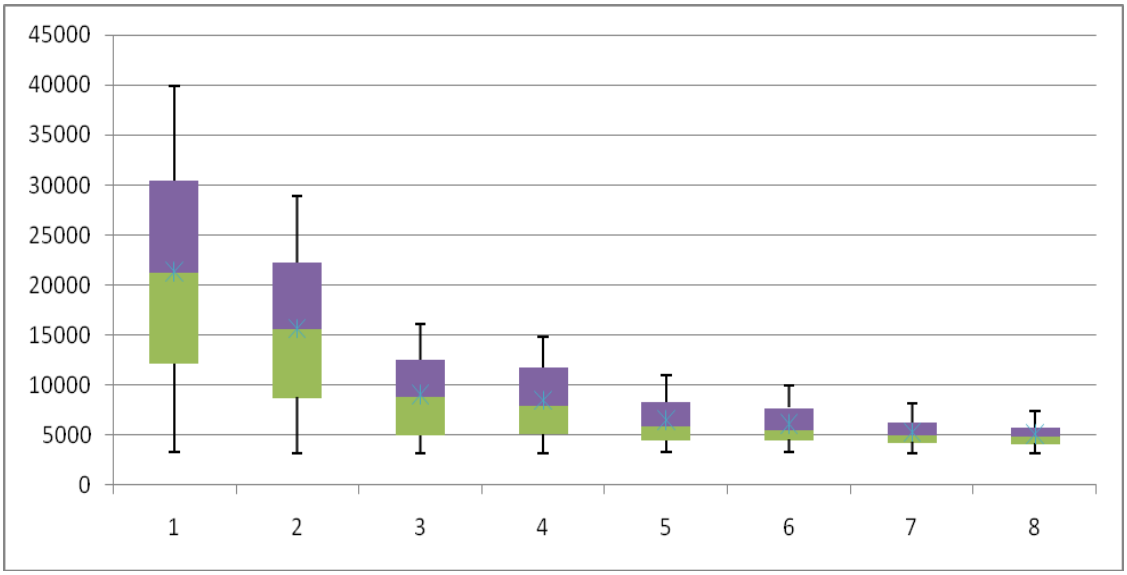


Figure 6 - H2 Box Plot

In H2, single core and double core plots symmetric boxes because they evenly split at median and mean is almost same as median. Whiskers are about the same length. These two cores have approximately same

variation in the data. The rest of cores seem to pair with each other in means of data spread. Core 7 and 8, are stable because of the less variation in data and spread than other cores.

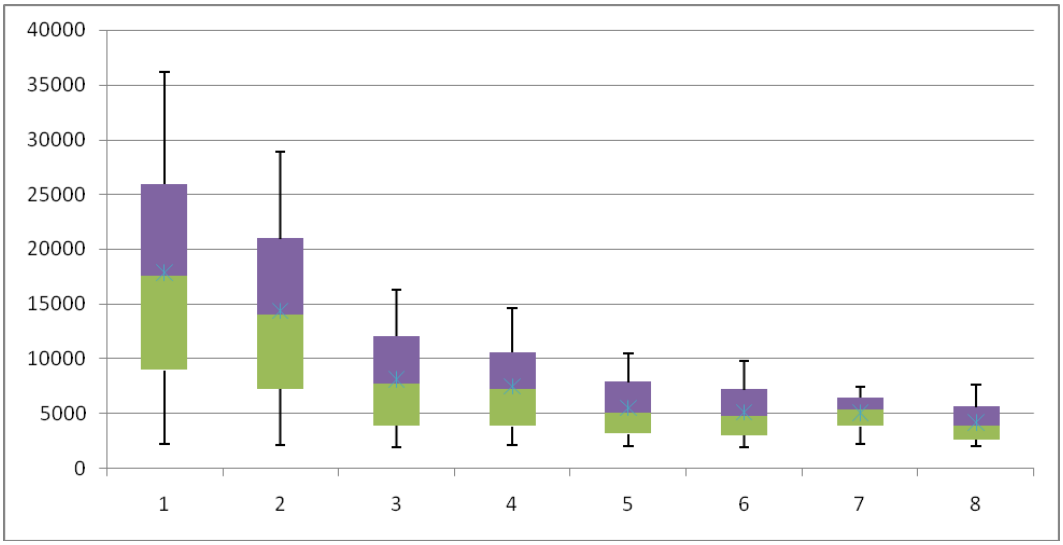


Figure 7 - Jython Box Plot

In general distribution of data for each core is right skewed with high response time, except core 7. On the contrary, core 7 is left skewed and most narrow data set in whole cores. In this core, data spread is wider in low quartile this means high amount in low response time.

From core 2 to core 3, there is a large gap between data sets because system start to give high performance. Core 3 and 4, seems similar in data variation. Also core 5 and 6, are showing similarity between them.

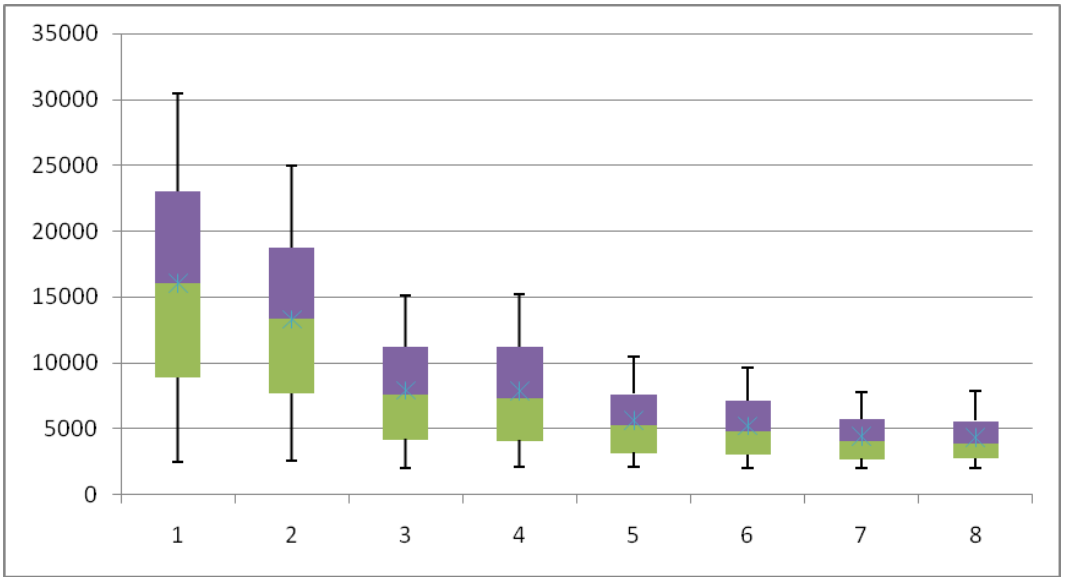


Figure 8 - Pmd Box Plot

Both single and double cores are symmetric and well balanced. But they have high variance in response time. Core 3 seems much faster than core 4. Core 5 and 6 have approximately same data variation and spread. Also we see same situation for core 7 and 8. As a result, I can say for pmd benchmark, instead of 8, we can run application with 7 core machine and this would be less costly.

Chapter 5

Conclusions and Future Works

Nowadays, we are accessing our applications, storage and services over internet through clouds. Since 2006, Amazon Web Services (AWS) provide cloud computing to external costumers. Virtual machines are one of the utility we use on AWS. In AWS, virtual machines have multi core. From single core to eight cores, we can select and make use for our applications. Unfortunately, there are few researches about performance of these VMs.

The main contribution of this paper has been evaluating performance of DaCapo benchmarks and has been measuring performance of multi-core virtual machine in Amazon EC2. Obtained results were used to demonstrate and evaluate a good estimation about performance. Since plain results were pointless, I created cumulative distribution charts, box and whisker plots and 3-D graphs to prove my points. Even though, i had done experiments with more than one threads, I showed results only with one thread. So I reduced result sets. But even with diminished set, I had a lot of data to have an idea about performance of an instance in Amazon.

For the future works, I consider, the impact of software multi threading on the behavior of the DaCapo benchmarks under different cores configurations. As a result, we can see how higher number of threads improve the response time performance of the benchmarks. And advantages can be discussed, if there are.

Although virtual machines on clouds are used for their efficiency, provisioning of a single core with many cores can be more costly than provisioning of many single-core machines. On this case, efficiency tests should be done with different benchmarks and also with different IaaS

providers. Typically, to meet customer needs the existing providers offer different numbers of virtual machines which varies according to their CPU types. So tests can also be done to see overall performance.

REFERENCES

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M. (2009). *Above the Clouds: A Berkeley View of Cloud Computing. Technical Report.* University of California at Berkeley.

[2] Boss, G., Malladi, P., Quan, D., Legregni, L., Hall, H. (2007), *Cloud Computing.* ww.ibm.com/developerworks/websphere/zones/hipods/.

[3] Fellowes, W. (2008). Partly Cloudy, Blue-Sky Thinking About Cloud Computing.

[4] Foster I, Kesselman C (1998) Computational Grids. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.4939

[5] Foster I, Kesselman, C, Tuecke S (2001) *The Anatomy of the Grid: Enabling Scalable Virtual Organization.* International Journal of High Performance Computing Applications

[6] Foster I, Zhao Y, Raicu I, Lu S (2008) *Cloud Computing and Grid Computing 360- Degree Compared.* In: Grid Computing Environments Workshop (GCE'08).

[7] Michael Miller (2009) Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online

[8] Mell P., Grance T,The Nist Definition of Cloud Computing http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[9] Huth A. , Cebula J. ,(2011) The Basics of Cloud Computing http://www.us-cert.gov/sites/default/files/publications/CloudComputingHuthCebula.pdf

[10] Bitzer F, (2008) Management Framework for Amazon EC2 http://cloud42.net/files/thesis.pdf

[11] Ahson S., Ilyas M. (2011) Cloud Computing and Software Services Theory and Techniques

[12] Gupta S., (2012) , Cloud Computing Technologies Overview and Comparison – "Microsoft Azure vs Amazon EC2"

[13] Buyya R., Yeo C., Venugopal S. ,Broberg J., Brandic Ivona (2008) Cloud Computing and Emerging IT Platforms: Vision,Hype and Reality for Delivering Computing as 5th Utility.

[14] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2, 2008.

[15] BlackBurn S.,Garner R.,Hoffmann C.,Khan A.,McKinley S.,Bentzur R.,Diwan A.,Feinberg D.,Frampton D., Guyer S., Hirzel M.,Hosking A., Jump M.,Lee H.,Moss J.,Phansalkar A.,Stefanovic D., VanDrunen T., Dincklage D.,Wiedermann B., (2009) The Dacapo Benchmarks: Java Benchmarking Development and Analysis

[16] The Dacapo Benchmark Suite web site. http://www.dacapobench.org/

[17] Giacomo D.,Brunzel T., (May,2010) Cloud Computing Evaluation, How It Differs to Traditional IT Outsourcing

[18] Chapter 1: Introduction to Xen Virtualization http://doc.opensuse.org/products/draft/SLES/SLES-xen_sd_draft/cha.xen.vhost.html

[19] Wang G., Ng E., The Impact of Virtualization on Network Performance of Amazon EC2 Data Center

[20] D.Cerotti, M.Gribaudo, P.Piazzolla, G.Serazzi, Flexible CPU provisioning in clouds: a new source of performance unpredictability

[21] D.Cerotti, M.Gribaudo, P.Piazzolla, G.Serazzi,(2013) , End-to-End performance of multi-core systems in cloud environments