TAMPERE UNIVERSITY OF TECHNOLOGY

The Faculty of Computing and Electrical Engineering

Jori Yrjölä

Using Service Oriented Architecture Platform on Cloud Computing Infrastructure

Master's Thesis

Examiner and subject approved in the Teaching and Research Council meeting on 4 November 2009.

Examiner: Prof Tarja Systä (Tampere University of Technology)

MSc Matti Pohjalainen (Nokia Siemens Networks Oy)

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master’s Degree Programme in Information Technology
YRJÖLÄ, JORI: Using Service Oriented Architecture Platform on Cloud Computing Infrastructure
Master of Science Thesis, 44 pages, 2 Appendix pages
October 2011
Major subject: Software Engineering
Examiner: Prof Tarja Systä
Keywords: cloud computing, Service Oriented Architecture, platform

Cloud computing is a growing Information Technology (IT) trend. It is said that cloud computing can change the essence IT business. Cloud computing is, like many other techniques, a convergence of old ideologies and techniques into something new. It combines features from cluster and grid computing with the help of virtualization. Cloud computing offers its users a seemingly infinite pooled computing resource over the network. Users can start, stop, and scale its power at will.

The goal of this thesis is to give a high level picture of cloud computing and implement a working prototype of a Service Oriented Architecture (SOA) platform on Amazon's Elastic Compute Cloud (EC2). The theoretical part studies what is cloud computing. Its supporting techniques and defining features are looked into as well as what benefits and drawbacks there are. Then a SOA Platform is implemented in EC2 in two separate configurations. The former implementation is an automatically scaling SOA cluster configuration. The cluster consists of Oracle SOA stack and uses the cloud computing. The cluster is used as platform to enable building of other enterprise systems using SOA technologies. The latter is a SOA training environment, which is used for training people to use Oracle SOA stack. Cloud computing enables trainees to take part in training sessions even with slow computers, because the necessary resources are accumulated from the cloud.

Both implementations successfully demonstrate some benefits and drawbacks of cloud computing. For example, the lack of upfront costs made this kind of project possible. Meanwhile, need for absent special requirements prevented the use of Real Application Cluster (RAC) database. Cloud computing is still evolving to smooth out some of its rough edges.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
YRJÖLÄ, JORI: Palvelulähtöisen Arkkitehtuurin Sovellusalustan Käyttö Pilvilaskentainfrastruktuurissa
Diplomityö, 44 sivua, 2 liitesivua
Elokuu 2011
Pääaine: Ohjelmistotuotanto
Tarkastaja: Prof Tarja Systä
Avainsanat: pilvilaskenta, palvelulähtöinen arkkitehtuuri, sovellusalusta

Pilvilaskenta on kasvava IT-trendi, jonka sanotaan muuttavan koko IT yritystoiminnan ytimen. Pilvilaskenta on monien muiden tekniikoiden tavoin useiden tekniikoiden yhteenliittymä. Se yhdistää ominaisuuksia muun muassa klusteri- ja grid-tekniikoista virtualisoinnin avulla. Pilvilaskenta antaa verkon yli käyttäjilleen näennäisesti rajattomat laskenta resurssit. Käyttäjät voivat aloittaa, lopettaa ja skaalata tätä resurssia mielensä mukaan.

Tämän diplomityön tavoite on antaa korkean tason kuva pilvilaskennasta ja toteuttaa toimiva palvelulähtöistä arkkitehtuuria (SOA) tukeva alusta Amazonin pilvilaskentajärjestelmään (EC2). Teoriaosuus tutkii mitä on pilvilaskenta. Myös sen tukevia tekniikoita, määrittäviä ominaisuusksia, hyötyjä ja haittoja tutkitaan. SOA-alusta toteutetaan EC2-pilveen kahtena erillisenä konfiguraationa. Ensimmäisenä toteutetaan skaalautuva SOA-klusteri, joka koostuu Oracle SOA -pinon ohjelmistosta ja hyödyntää pilvilaskennan ominaisuuksia. Tämän ympäristön on tarkoitus toimia alustana, jonka päälle voidaan rakentaa muita yrityksen järjestelmiä SOA-teknologioita käyttäen. Toisessa toteutuksessa tehdään yhden tietokoneen koulutusympäristö. Tätä ympäristöä käytetään, kun opetetaan Oracle SOA -pinon käyttöä. Pilvilaskennan käyttö tässä tarkoituksessa mahdollistaa koulutukseen osallistumisen myös vaatimattomalla tietokoneella, koska tarvittavat resurssit saadaan pilvestä.

Molemmat toteutukset tuovat onnistuneesti esiin joitain pilvilaskennan hyötyjä ja haittoja. Esimerkiksi ennakkomaksujen puuttuminen mahdollisti tällaisen projektin. Toisaalta puuttivien erityisominaisuuksien tarve esti Real Application Cluster (RAC) -tietokannan käytön. Pilvilaskenta kehittyy vielä ja samalla sen karkeat reunat siliävät.

PREFACE

Nokia Siemens Networks presented me with an opportunity to write a master's thesis. There were several potential subjects of which Using SOA Platform on Cloud Computing Infrastructure was chosen. The subject was chosen because cloud computing is a hot topic and it could benefit an existing project using SOA platform.

I was pleased to embark on this journey to the clouds. It provided me an opportunity to learn something new in the sense that I knew nothing about it as well as in the sense that it was totally new to the public. As well it was fun to learn something totally new, it also provided a challenge while trying to determine a proper scope of things.

The software stack and cloud computing environment were predetermined for this project by Nokia Siemens Networks (NSN). The software stack consists of Oracle SOA Suite 11g and whatever is needed for it. The cloud computing environment was chosen to be Elastic Compute Cloud (EC2) provided by Amazon. Beyond that I had mostly free hands in planning how to advance the project.

This thesis was written as a masters thesis for Tampere University of Technology (TUT). Thesis work was examined by Professor Tarja Systä from TUT and MSc Matti Pohjalainen from NSN. Several other NSN employees contributed to the project by offering advice on thesis and on issues concerning the implementation while not forgetting to boost my morale when the work seemed overwhelming. I would like convey my gratitude to all of them, Prof Tarja Systä for guiding the thesis, and Nokia Siemens Network for paying the bills.

When beginning the project, I had no idea what cloud computing is. Now I'd like to think I have some understanding of what it is, and why it is useful. Hopefully this thesis helps to explain some of that.

Jori Yrjölä
13 June 2011

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS AND NOTATION

Abbreviation	*Explanation*
AMI	Amazon Machine Image, a virtual machine image used with Elastic Compute Cloud.
API	Application Programming Interface, an interface through which a library or a service is used.
AS	Application Server, a software server application that runs other applications.
AWS	Amazon Web Services, a set of commercial web services.
BPEL	Business Process Execution Language, a language used for interacting between web services.
DB	Database, a data storage for applications.
DHCP	Dynamic Host Configuration Protocol, a protocol that is used to assign IP addresses in a network.
EBS	Elastic Block Storage, a persistent storage device in EC2.
EC2	Elastic Compute Cloud, a cloud computing environment provided by Amazon Web Services.
ELB	Elastic Load Balancer, a load balancer for EC2
GPGPU	General Purpose Graphics Processing Unit, a graphicks processing unit that can be used for any computing. They are highly efficient in parallel floating point operations.
GUI	Graphical User Interface, a graphical system that takes input from user, controls the software, and gives graphical feedback to the user.
IaaS	Infrastructure as a Service, a cloud computing service that lends virtual infrastructure.
IO	Input / Output, Input and output data of networking or computer storage devices such as harddrive.
NFS	Network filesystem, a protocol that allows a network drive to be accessed the same way as a local drive.
PaaS	Platform as a Service, a cloud computing service that lends web application platform.
RAC	Real Application Cluster, a database cluster provided by Oracle
RCU	Repository Creation Utility, a tool to create Oracle Middleware database schemas.
RHEL	Red Hat Enterprise Linux, A commercial Linux distribution.
S3	Simple Storage Service, a data storage provided by Amazon Web Services.
SaaS	Software as a Service, a cloud computing service that lends software services.

Abbreviation	***Explanation***
SCP	Secure Copy, a secure communication tool for copying files to or from a remote computer over the network.
SOA	Service Oriented Architecture, a design architecture for connecting enterprise systems.
SOAP	Simple Object Access Protocol, a XML based communications protocol that is used for calling operations remotely.
SSH	Secure Shell, a secure communication tool for accessing a command line of a remote computer over the network.
VM	Virtual Machine, a virtual representation of a computer that is running as a guest on a host computer.
WKA	Well Know Address, a Coherence's configuration parameter that determines a server address that is always part of the cluster.
XML	Extensible Markup Language, a text based document format.

Term	***Explanation***
Boto	A Python interface for Amazon Web Services API.
CentOS	A non-commercial Linux distribution which is a Red Hat clone.
Cloud	A single cloud computing environment.
Cloud Computing	An elastic, on-demand computing resource offered by a provider over a network.
Cluster	A cluster is a set of computers that are configured to work together for a single purpose.
Coherence	An in-memory distributed data storage.
Composite application	A deployment package for SOA server applications.
Dehydration	BPEL process is stored in a database when it is waiting for something.
Ephemeral	An ephemeral drive loses all changes done to it when it is powered down. Opposite to persistent.
Fedora	A non-commercial Linux distribution.
Firefox	An Internet browser.
High Availability	A fault tolerant hard- and software configuration.
Hypervisor	A system that monitors and runs guest operating systems in virtualization.
Image	A file that contains a filesystem and can include an operating system. An image can be used as a filesystem for a virtual machine.

Term	***Explanation***
Instance	EC2 uses a word instance for its virtual machines.
Jdeveloper	A multi-purpose integrated development environment.
Multicast	Technologies that allow a single data transmission to be received by multiple endpoints.
Operating System	The software that manages resources in a computer.
Persistent	A persistent drive is an opposite of ephemeral. It remembers all changes even if it is powered down.
Platform	A software system that provides a base functionality for software components that use it.
Scripts	A set of Python scripts that were created to operate the implemented system.
SOA Cluster	A configuration that serves as a platform for SOA systems. The cluster provides the distributed resources the platform needs in order to serve.
Web Service	A set of standards for creating services that are used remotely.

1 INTRODUCTION

IT infrastructure, including SOA systems, have traditionally been implemented on an actual hardware owned by the company. Virtualization alone has not shaken this tradition, because virtual servers still reside on servers back at the office. However, virtualization was a key ingredient in creating *cloud computing*, that is an elastic computing resource provided on-demand over the network. Cloud computing has set out to remove those dusty servers from the offices while lowering the expenses and increasing service quality.

Growing interest in cloud computing has raised interest in moving existing and future IT-systems to the *cloud*, which means a single cloud computing environment. Like many other companies, NSN has several projects with interest in reaching for the clouds. One of these projects involved a *SOA cluster* configuration, which is a multi node system that runs SOA technologies to run services. The goal of this project started to form around putting that SOA cluster into Elastic Compute Cloud (EC2) [1] by Amazon. While explaining how this was done it will be necessary to unveil some of the mysteries of cloud computing.

The goal of this thesis has two major parts. The first is to map the possibilities of cloud computing. This includes an explanation what cloud computing actually means. Also different cloud types need some clarification. After that there will be pondering on the possible benefits and drawbacks of cloud computing. The theoretical part will grasp the surface on most important cloud computing aspects, but it will focus on the issues surrounding the implementation. This knowledge is then utilized in the implementation part.

The goal of the implementation part is to create a working Service Oriented Architecture (SOA) [9] application cluster prototype and a training environment in Amazon's EC2. The cluster configuration uses SOA Suite 11g backed up by Oracle 11g database. This cluster is configured in a way that it benefits from the features of cloud computing. The cluster configuration is a platform that allows the development and use of other SOA based enterprise systems. The training environment offers trainer an easy way to provide a high performance environment to the trainees.

The ideas behind cloud computing are not new. Cloud computing is the closest thing to the idea of utility computing [21] so far. Utility computing is a thought that computing would be provided the same way as e.g. water or electricity that are available in every home pre-installed ready to use. Unlike computing nowadays, utility computing means outsourcing all the hardware and getting charged by the use. Some grid computing environments have reached towards this goal. Moving towards network-based systems and emergence of virtualization have affected the business model of computing so that it formed a new category called *cloud computing*. This happened as some large virtual clusters, and computational grids started to adopt a cloud computing features to better manage their economy.

The thesis is divided into 6 chapters with this introduction being the first chapter. The second chapter focuses on explaining cloud computing. It explains related techniques, cloud computing characteristics, different types of cloud computing, and other interesting aspects. The third chapter is about benefits and drawbacks of using cloud computing. The benefits and drawbacks are weighted in the light expenses, speed, and environmentalism. The fourth chapter explains the implementation of the SOA cluster. The implementation description includes details on the environment, the tools used, and the implementation process. The fifth chapter explains some cloud computing related abandoned approaches, and open issues. There are also recommendations for future development regarding the open issues. The sixth chapter describes a standalone SOA training environment. The purpose of the environment is to allow trainees to start immediately developing SOA without the need for a high performance system. The seventh chapter estimates the success of the implementation and some final thoughts. The success of the implementation is weighted in the light of gaining the benefits of cloud computing and running a working SOA cluster.

2 CLOUD COMPUTING DESCRIPTION

There are many attempts to define cloud computing. These definitions do not always match each other, and thus create confusion [13]. Some definitions appear to be too abstract to really define anything. Even cloud computing definition by National Institute of Standards and Technology (NIST) [18] comes with a disclaimer that the definition is still in process of taking its final form. NIST defines cloud computing by its key characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. In this thesis, these characteristics are used as a base for defining cloud computing. They offer a practical approach to define cloud computing in small steps.

The first section describes techniques that are closely related to cloud computing. The most relevant mutual and differentiating features are discussed here. These techniques are the cornerstones that lead the development towards cloud computing. The second section explains cloud computing characteristics. They define the common behaviour and features of cloud computing. The next section describes different types of cloud computing. These types operate on different abstraction levels. The last section explains other cloud computing aspects that are worth mentioning. These include cloud computing deployment models and service level agreements.

2.1 Related techniques

Related techniques are either an integral part of cloud computing or very similar for some characteristics. These are existing techniques that have been used before the emergence of cloud computing. Now cloud computing is using the best parts of these techniques to provide flexibility.

Grid, cluster, and cloud computing are each a type of distributed parallel computing. The first subsection is about grid computing. It has been referred to as an ancestor of cloud computing [6]. Next subsection defines what is a cluster. Clustering is probably the most common form of performance demanding computing. The third part explains virtualization. It is a key techniques that has allowed sharing computing power in an easy and safe way. The last part explains what is SOA. In short, it is a design architecture for connecting enterprise systems.

2.1.1 Grid computing

Grid is a distributed system that runs on multiple interconnected nodes. The idea in grid computing is to pool the resources of multiple participating factions to form a computational grid. The grid offers its users the necessary computing power that would otherwise be unavailable to them. Grid gives its users a generic programming interface that they use to do whatever they are doing with the grid. [21]

There are several different types of grids such as Data Grid and Computational Grid. These grids have different purposes and therefore different features. However, there are some common characteristics between different grids that are also common with cloud computing. One of the is the highly distributed nature. Both grids and clouds consist of up to thousands of nodes. Grids and clouds are both utilised by multiple users, but they differ in resource management. Part of that comes from the different ownership model used by grids and clouds. Grids are usually owned by multiple owners while clouds and clusters are privately owned. [20]

Grid is mostly utilised in computational or data intensive tasks, such as scientific calculations, that provide results for the user. After that the user may cease to utilise the grid. Consequently, the needs of one user in the grid may vary greatly. This is not an issue, because when one user stops using the grid, the capacity is released to other users. Heterogeneity of the grid means that there are many different types of computers which are in networks of different quality. This is why it is hard to run even loosely time critical applications, like web applications, in the grid. This also means that quality of each node cannot be assured. [21]

2.1.2 Computer clusters

A cluster is a set of multiple interconnected computers. Classifying a set of computers as a cluster requires software that makes those computers work together. For example, a cluster can run a distributed service where workload is divided among the computers in the cluster. Clustering is done with software and hardware configuration that supports it. [21]

The two most common reasons for clustering are performance and *high availability*, which means a fault tolerant hard- and software configuration. Performance clustering is a natural way to add performance if one node configuration is not enough. In this configuration, a system, like a load balancer, divides the workload among the clustered machines. High availability configuration adds reliability by avoiding a single point of failure. These configurations are also often used together in an active cluster configuration.

Grid nodes usually are globally distributed, whereas cluster nodes exist in one physical location and are connected with high speed network. Common reason for this is that clusters are privately owned and are built for some specific purpose with homogeneous hardware. This leads to the fact that a cluster can usually assure certain level of quality. [21] [20]

Compared to the grid and cloud environments, a cluster is an order of magnitude smaller unit [21]. Clusters tend to run the same system or service their entire lifespan, where the clouds and grids run a generic environment that runs many different applications.

2.1.3 Virtualization

Virtualization allows a complete separation of an *operating system*, the software that manages resources in a computer, from hardware. This is possible by allowing a host operating system, the system that runs on the hardware, to create a virtual environment that can run any machine code that the hardware supports.

In practice, virtualization means a case where a host operating system runs other operating systems as guests. This allows an easy and secure way to split the resources of a computer among users or software. The guest system is called a virtual machine (VM).

There are different levels of virtualization, but in this case we are interested in paravirtualization, because it is used by EC2. In paravirtualization, there is a simplified hardware abstraction layer inside the VM. Its OS contains a kernel that is enhanced with features that remove some inconveniences and add performance. The OS is modified without changing the application binary interfaces so that the applications themselves do not need modification to run properly. [17]

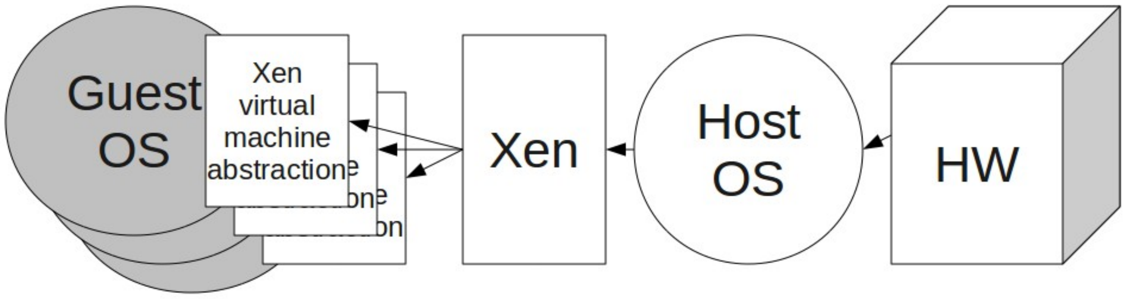

Figure 2.1: Xen virtualization

Xen virtualization is of particular interest, because it is used in EC2. Figure 2.1 shows a crude separation of components in Xen paravirtualization. Computer hardware runs an OS. This OS acts as host for virtualized guest operating systems. Host OS runs Xen *hypervisor*, which is the software that controls the guest OSes. Each guest OS has a VM abstraction that allows controlled access to hardware. The VM abstraction is a modification to guest OSes kernel. The modified kernel allows Xen to run Guest OSes with minimal performance penalty. [17]

In a cloud computing sense, this allows splitting big multi core machines with huge amounts of memory into smaller units in a controlled fashion. This means that a cloud computing provider can allow users to split an environment to a desired size unit. Virtualization takes care that these smaller environments do not take more resources than they are supposed to. This allows for a cloud computing provider to offer a stable quality of service to the customers. [21] [10]

2.1.4 Service Oriented Architecture

In enterprise IT architecture there is a need for different systems to utilize each other. Services Oriented Architecture tries to answer this need. SOA is defined as an architectural design principle that promotes loosely coupled interactive software agents. [9]

In technology perspective, SOA is a standard way in creating connections for enterprise systems. These systems offer their capabilities as services. The interfaces of those services use a standard implementation independent description and tools. Then those services offer descriptions describing how their interfaces are to be used. For example, a database that holds customer information offers a customer data service. The interface for a service is then provided in a implementation independent manner for maximum compatibility. Once implemented, this service is available for any other services that require customer information. [26]

There are many sets of technologies that are used for creating SOA. The most interesting for us is the *web service* approach, which means a set of standards for creating services that are used remotely. The web services approach is supported e.g. by Oracle SOA Suite which is used in the implementation phase. In this approach the service interface is defined with WSDL (Web Service Description Language), which is an Extensible Markup (XML) based language for describing web service interfaces. Web services send and receive Simple Object Access Protocol (SOAP) messages. SOAP is an XML-based communications protocol. [26]

Cloud computing often utilizes SOA principles. Amazon Web Services (AWS) for example, offers its service interfaces as web services. In implementation part of this project AWS is called from a Python script by using *Boto*. Boto is a Python library for interacting with AWS. It is explained further in Subsection 4.2.3.

2.2 Cloud computing

Cloud computing offers instantaneous, location independent, high quality, computing power to users and charges according to consumption. This computing power comes in many different forms depending on abstraction level and provider. Abstraction levels are categorized in three levels. Abstraction levels are from lowest to highest: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Also different providers have slightly different view about each level. However, the common characteristics determine what is cloud computing.

Cloud computing is a combination of grid and cluster ideologies with a twist of virtualization. It is a utility like way to provision computing power among users. Where clusters and grids differ, cloud computing brings them together. Cloud computing is a generalization of both allowing a user to customize it according to his or her needs.

Pay-as-you-go charging model makes it possible to use cloud computing for short running, processes demanding high performance, where grid is used typically. However, cloud computing providers set some restrictions on how much capacity a user can consume at once. For example in EC2 a user can run up to 20 instances. After that a special permission is needed to add capacity.

Similarly to clusters, in a cloud computing environment a user can run on-line processes that run indefinitely. Those processes run for extended periods, but may use very little processing power. These processes set a high bar for the quality of service. Cloud computing providers offer high quality of service.

This section describes cloud computing features. The first subsection explains on-demand self-service, which in short means, that users can use a cloud any time and receive instant service. Next subsection explains broad network access. Cloud computing is logically a networked resource that needs to be accessed through a network. The third subsection describes resource pooling feature. This feature means that cloud computing shares resources between the users. The fourth subsection describes rapid elasticity. Rapid elasticity means that the amount of user reserved computing resources can be changed at a moments notice. The last subsection describes measured services. With this feature user's resource consumption is tracked and the user is charged accordingly.

2.2.1 On-demand self-service

Cloud computing offers instant access to its services. In other words cloud computing is ready on-demand. This allows a user to access the cloud at any time and receive almost immediate service. On-demand self-service is closely entangled with rapid elasticity. [18]

Unlike cloud computing, grid computing supports coordinated resource sharing [20]. Resource usage is planned in advance and resources are allocated to the users according to their needs. Cloud computing model does not involve planned resource consumption in this sense. Cloud computing assumes that resources are sufficient. Service provider adds hardware resources when average loads climb high enough. However, cloud computing may restrict users who are doing big reservations. EC2 for example needs special arrangements, if the user wants to reserve more than 20 instances [1].

2.2.2 Broad network access

Cloud computing requires broad network access. This makes it location independent by offering cloud computing accessible from the internet. Network access allows remote client access. Computing power is available to all kinds of clients through networking. These clients may include any thick or thin client from data centres to mobile phones.

Cloud's network needs to be of high quality. Especially, it needs to be fast both inside and out of the cloud. Low latency makes things usable in cloud. For example, web pages lose much of their usability when latency is too high. The other aspect of fast networking is high throughput. That means that latency should not go up when there is much traffic in the cloud. [18]

2.2.3 Resource pooling

Characteristic to cloud computing is that the resources are shared among multiple users. The resource allocation is done by self services accessible to the end users. Users can choose an amount of resources or a service level on which they operate. The resources can be for example VMs like in EC2. There user can choose the amount and type of VMs from pre-made configurations he or she wishes to use. Those resources become available at a moments notice.

Grid computing is very similar to cloud computing in this sense. A huge computing resource is built to work as a unified service, a pool. This pool is then accessed by multiple users. Users release their resources when they are finished. Cloud computing differs in allocating this resource. Cloud computing users can slice up the smallest piece of the cloud cake while grid is often used for very demanding purposes. [20] [11]

2.2.4 Rapid elasticity

Rapid elasticity is closely connected with on-demand self-service. The amount of resources a user consumes can be changed rapidly. Depending on the cloud architecture, the user can either reserve an amount of resources he or she can consume or just consume the resources.

Once the user is registered to a cloud computing service, there is no bureaucracy in using the service. Cloud services have an Application Programming Interfaces (API) that allows the user to manage the resources. Alternatively, the user can configure a scaling automation software to manage the resources. This opens a lot of possibilities in resource management. For example, services that have much variation in demand can increase performance when needed and decrease when demand drops. [1]

Part of the rapid elasticity is being easily scalable. Cloud computing allows easily scalable systems. For such a system, it is easy to add and remove computing power. This allows the system to scale according to the load. The scaling differs between cloud computing architectures. Some cloud computing environments like Google App Engine offer an abstract completely automatically scaling system [7]. Meanwhile EC2 offers a set of tools to create a visible system for scaling [1] [11]. EC2 does not do load balancing automatically either. The user must take this into account when designing a scalable system in EC2.

2.2.5 Measured Service

Cloud computing charging model requires a measurement of services. The measured services is a key element that allows a user to be charged according to the consumption. Measured services are tightly attached to on-demand type of business. This way a user can consume at will and the cloud provider can charge for the resources that were used. The user can follow and predict the costs of his or her resource consumption.

The measurement is done somewhat differently in different cloud computing services. For example, EC2 measures amount of data that is stored, network usage, and *instance*, which is EC2 word for a virtual machine, reservation time [1]. Meanwhile, Google App Engine [7] measures CPU time instead of instance time. Network usage and data storage are used in similar fashion. The difference is that measuring CPU time does not count the time the system is idle. As long as the system does not do anything the only cost is the data storage.

2.3 Service models

Cloud computing has many different approaches. These approaches provide somewhat different ways to use and think about cloud computing. The common features that define cloud computing are still present. Mainly these differences can be categorized in different cloud types. These cloud types represent three separate abstraction levels where cloud computing services are offered. These abstraction levels are IaaS, PaaS, and SaaS.

First subsection explains IaaS. It is lowest level of cloud computing abstraction models. Second subsection is about PaaS. It offers a total abstraction from hardware with an application server stack. The highest level of cloud computing, SaaS, is described in the last subsection. It is a complete hardware and software stack implemented as a cloud service.

2.3.1 Infrastructure as a Service

IaaS is the lowest level of abstraction type of cloud computing models. It offers leasing of a virtual hardware infrastructure. That virtual hardware is almost immediately accessible and ready to use. Virtual hardware appears in form of VMs. VMs appear as computers to the user.

The VM can be used for almost any purpose with few restrictions depending on the cloud computing provider. For example, the VM has only the most basic set of virtual hardware at its disposal. If any specialized hardware is needed, it has to be external to the cloud.

The user is charged according to the resource consumption. Running a VM means that the user is charged according to the VM's maximum capacity, even if the VM is idle. [1]

2.3.2 Platform as a Service

PaaS is the second level of abstraction in cloud computing hierarchy. While in IaaS, a virtual hardware is offered, PaaS offers a software stack and a complete abstraction to hardware. The software issues such as parallelism and distribution are dealt by the software stack.

Platform is a level of abstraction that is often build on top of infrastructure to provide two things: abstraction and standardization. Abstraction means not having to care about things like paralleling. Meanwhile, standardization allows users, who are familiar with the standard, to develop software for it.

The user is charged based on how much resources the service consumes. The difference to IaaS is that only used CPU cycles are charged, at least in Google App Engine [7]. There is no CPU charge when the system is idle.

2.3.3 Software as a Service

SaaS is the highest level of cloud computing. SaaS offers a software product to the users. These products range form machine to machine services to desktop applications. Machine to machine systems can use SOA technologies to deliver services. While IaaS and PaaS are aimed for a software developer, SaaS is often aimed directly to the end user.

The idea of SaaS is to offer an instant solution to a specific problem. That solution is presented as a service that the end user can utilize according to cloud computing paradigm.

One type of SaaS business is to replace desktop applications [8], if not the whole desktop. Instead of running software on a desktop computer, the software is ran in the internet as a multi-tenant application. This means that traditional desktop program, like word processor, is ran on a cloud computing platform over the Internet. This offers several advantages over running such program on desktop. The need for software updates is completely removed from the end user perspective. Also differences between versions does not cause as much problems from the support perspective, because each user is using the same version of the program. However, it does not remove the need for a software developer to maintain compatibility between versions, because files made with an older version will most likely exist. Another alternative for the software developer would be to keep the files in a cloud data storage. This would would simplify the end user experience. For example, document sharing and versioning could be integrated to the software. Meanwhile, the developer could choose the best back-end system for the two tasks.

Pricing in SaaS is somewhat different from IaaS and PaaS. SaaS usually uses something else as a measurement than CPU and bandwidth usage. For example, Salesforce.com is a SaaS cloud provider. Its pricing is done via monthly subscription. [23]

2.4 Other aspects

There are some aspects in cloud computing that are not covered by either the technologies behind them or by cloud service model they represent. These aspects are affected mostly by the business side of cloud computing. Business determines what kind of cloud computing deployment model is good.

The first subsection differentiates public and private clouds. Public clouds are publicly available in the Internet while private clouds are not accessible except for certain participants. The second subsection covers cloud computing service level agreements. The agreements determine the contract between the client and the cloud computing provider.

2.4.1 Deployment models

There are four different deployment models defined for cloud computing. They offer different approaches to cloud computing. The two major models are public and private clouds. The other two models are hybrid cloud and community cloud.

The cloud computing environments that offer services to whoever is willing to pay for them are public clouds. Public cloud services and whatever the user puts in there are accessible through the Internet. However, the access is restricted by a firewall. A cloud is owned by the cloud computing provider and the users just use what they pay for. The instances in EC2, for example, get a public IP address that makes them accessible from the Internet. [18]

Private clouds are owned by corporations which uses them. They are internal to the corporation. They may be completely behind the firewall in a way that no access from the Internet is possible. However, there is some debate if private clouds are a part of cloud computing at all [13]. This argument is based on the fact that cloud computing business model assumes that the users are charged by the resources they use. However, since pay-as-you-go paradigm works as well in internal charging as an external, it is only a matter of defining if the word cloud is regarded as a public concept. [18]

Hybrid cloud is a combination of public and private clouds. It utilizes public cloud for non-critical information handling while the critical information is handled in private cloud. Hybrid cloud may also be used to extend the capacity of the private cloud. This scenario requires some form of compatibility between the environments. That compatibility is greatly dependant on the service model that the cloud represents. For example Eucalyptus, that could be used as a private cloud, has interface level compatibility with EC2. [18][6]

Community cloud is a pooled resource cloud that combines the resources of multiple community users. The idea is similar to the idea of a grid. The difference being the management of resources. Cloud model dictates that users can acquire resources on demand and are charged accordingly. Grid computing on the other hand relies on planned sharing of resources. [18]

2.4.2 Service Level Agreement

Service level agreement (SLA) is an important aspect to notice for anyone using cloud computing. The SLA contains the terms a user has to agree on when starting to use cloud computing. These terms include details on what the provider is offering the user and what are the responsibilities of both parties. As customers use any external service, they become more or less dependant on the provider. Often the most important aspect of SLA to the customer is the continuity of the service as customers need guaranteed services [21]. Cloud computing does not offer good guarantees.

For example EC2's SLA guarantees a 99.95% annual regional availability. This means that each region can be unavailable for roughly four and half hours a year. After that the user is entitled to service credit. Entitlement to service credit means that EC2 will return a portion of the clients expenses. Additionally the service credit must be separately requested. The service credit is in no way bound to the losses the customer may experience. [1]

3 BENEFITS AND DRAWBACKS OF CLOUD COMPUTING

The main driving force of cloud over traditional computing is that it claims to offer better quality with lower price. The focus of this chapter is to look into the theory of how cloud computing is able to offer benefits and what drawbacks there are. Most of the benefits of cloud computing come from different sources, but in the end they all are reflected as lower expenses. Resource pooling for instance, can be achieved without cloud computing, but with cloud computing it comes for free.

Benefits of cloud computing are explained in the first section. Cloud computing offers lower price, faster deployment, and environment friendliness. The other section describes what drawbacks cloud computing may have. Drawbacks are a result of the higher abstraction from hardware which limits some options while operating in the cloud.

3.1 Benefits

Cloud computing offers benefits for system developers over traditional environments. These benefits are lower expenses, faster time to market, and environmental friendliness. Cloud computing business model makes these benefits possible by pooling resources and offering them to the users via self-service.

The first subsection explains how cloud computing offers lower expenses than traditional server-based IT. The lowering of expenses come from the cloud design and business model that allows users to pay for only what they use and provider to operate at low cost. Next subsection handles the benefit of faster time to market. The ability to quickly procure resources means less time waiting and fast response times to market changes. The third subsection describes the environmental benefits of cloud computing. These benefits are a result of high utilization rate in a cloud. The fourth subsection describes the benefit of community support. The community forms naturally around cloud computing environment and supports it.

3.1.1 Expenses

Cloud computing offers benefits in lowering expenses. These expense cuts are a result of using the advanced features of cloud computing. They are mostly results of controlling the costs of running a complicated hardware resources. The elasticity of a cloud computing environment offers a flexibility in planning an infrastructure. Cloud computing offers a better way to use resources according to the actual system consumption. For example when planning for a web server, a good idea is to start with the cheapest computer and add resources when the need arises. When using a conventional approach, the server usage must be estimated and planned in advance. In most cases this leads to overestimating the capacity just to be safe. [13]

Cloud computing providers can minimize costs by concentrating on a huge cloud computing environment in a centralized, low-cost location. This cuts down maintenance personnel's travelling costs and optimizes their work time. A cloud computing provider can concentrate on hardware issues and on the cloud computing environment. Using homogeneous hardware and software configurations, the provider can run numerous computers with very little staff. [13]

Another side of expenses is upfront costs. Cloud computing offers an opportunity to use resources without any upfront costs. Rather than cutting the costs it reduces the risk. The cost of the hardware is spread over the lifetime of the project. There is also no need for dedicated hardware for short term projects, of which cost might otherwise be hard to rationalize. Projects that end before their time leave no lingering hardware behind. [13]

3.1.2 Faster Time-to-Market

Cloud computing claims that products using cloud computing allow faster time to market. This is a result of rapid elasticity. Rapid elasticity can be explained as two things: rapid scalability and rapid deployment.

In IaaS rapid scalability means that replicating and starting nodes is easy and fast. In other words, it is relatively easy to compensate for increased market demand for the system. EC2, for example, uses Amazon Machine Images (AMI) as a base for running instances. To add more computing power to an EC2 system, more instances are started. One *image*, which is the file that stores permanent parts of the instance, can be used to start multiple clone instances. On high load the user can start more instances, scale out, and add them to share the demand with a load balancer. However, this kind of clustering also requires support from the software stack running on the server. If such configuration is not possible, the user can often also start a more powerful instance. In most cases these procedures can also be automated.

Rapid deployment is the ability to instantly start using a production quality environment. This is made possible by the lack of hardware procurement and bureaucracy in using cloud computing. Users can start using the resources instantly after accepting the terms of agreement and adding a payment method. Lack of upfront costs also helps to cut through the bureaucracy inside enterprises and makes decisions faster. Planning is shortened by the fact that hardware planning is not as time consuming. There is less need to estimate how much electricity the hardware will consume, and what hardware to use. Planning of how many computers is required is unnecessary, except for projecting costs, because scaling up or out can be done when the need arises. Even more time is saved, because acquiring the hardware is bound to take time. [24]

3.1.3 Environmental friendliness

Environmentalism has been considered a great marketing asset ever since climate change became widely accepted. Cloud computing claims to be an environmental alternative to traditional computing models. It is offering the same amount of processing power with less impact to the environment.

Growing data centres consume more electricity. Big cloud computing providers like Google have taken interest in energy consumption [25]. In big data centres, even smallest optimizations in hardware or software configurations affect the total outcome. While processing shifts from smaller data centres to big cloud centres, energy efficiency optimizations become more cost-effective.

The resources of a traditional cluster are often planned with peak loads in mind. For example, running web applications sometimes means that workloads can vary greatly due to the vast amount of Internet users. To get the application running smoothly at all times, the cluster has to be scaled for a peak load. In regular use this means that the cluster is hardly ever running at more than 10% of its maximum capacity. This leads to a huge waste of resources. This loss is rarely avoided without compromising peak load performance. In cloud computing, the laws of probability will come to help. When the cloud is big enough, peaks in some users performance needs will not tip the boat. It is unlikely that most users experience peaks in their systems at the same time. This allows a big cloud computing operator to run constantly at high utilization rates. [12]

Cloud computing charging model also encourages to minimize wasted resources, because it directly affects the costs of usage. The model also helps to directly monitor how much resources are used. Cloud computing can also be compared to a server room scenario. Developers are not as much tempted to leave the instances running for the night, because in the cloud they do not have to go to the server room to start them again in the morning. [13]

3.1.4 Community support

EC2 has formed a tight community around it. This community is formed of EC2 users that are actively using the cloud environment and helping in solving problems in the community forums. Internet based services naturally form communities around them, especially with the help of the provider. EC2 helps the community to work together by providing the forums and a possibility to share custom AMIs. [2]

Sharing AMIs often gives EC2 users a head start when creating new systems. The community has many entrepreneurs and other community members that provide pre-installed and sometimes pre-configured systems to fulfil the needs of the community. [2]

3.2 Drawbacks

The benefits of cloud computing come from a specialized architecture and business model. This specialization naturally limits some options. These limitations affect both the implementation and the business.

The first subsection introduces possible hardware related drawbacks. These drawbacks are caused by the general purpose nature of cloud design. Next subsection catches a glimpse of performance related concerns in the cloud. In the cloud a user lacks control over its performance. The third subsection looks into public cloud security. Sharing resources with other users has caused security concerns. The last subsection looks into legislative side of cloud computing. Countries have different laws that restrict how and where can sensitive user data be stored.

3.2.1 Hardware

Cloud computing providers choose the best hardware for their purposes. This hardware offers generic processing capabilities and access to mass storage devices. For the user, this means that any specialized hardware cannot run in a cloud. The user must acquire that hardware outside the cloud. For example, modern GPGPUs (General Purpose Graphics Processing Unit) are not commonly available in cloud computing environments. The GPGPUs are particularly useful for parallel computing operations. This might change in the future due to increasing availability of GPGPUs [25].

Hardware in the cloud cannot be customized beyond a certain degree. Usually for IaaS clouds there are some pre made configurations that a user can choose from. The variances between the pre-made systems usually only affect the computing performance. In EC2 the only other factor is CPU architecture. The user can choose between 32-bit and 64-bit architectures. While this is rarely a choice, because the configurations using those architectures are at different performance levels. 32-bit systems occupy the cheap end with minimal performance while 64-bit systems occupy the higher performance configurations. [1]

While there is no direct access to any hardware, a user can reserve a certain amount of virtual hardware. IaaS allows user to run software directly on virtualized hardware. In PaaS, on the other hand, the user does not have access even to a virtual hardware. Special hardware needs still to be outside the cloud. While IaaS and PaaS target developers, SaaS is a complete product that targets end users. SaaS provider takes care of whatever special hardware or other requirements the system might have.

3.2.2 Performance

Similarly to grids, there might be performance issues in a cloud [11]. The performance of the cloud may vary. Although every system may suffer from performance issues, a cloud user does not have direct control over the issue. These performance issues may be caused by many reasons like network traffic, CPU load or Drive load.

While virtualization is an effective way to control the distribution of CPU power in IaaS Cloud, the cloud may still suffer from drive IO issues. Although clouds like EC2 have high IO performance, it does suffer under heavy load. [13]

Networking has some performance issues as well. Naturally cloud computing providers have to invest a great deal to their network. It is not sensible to share networking capability in the same sense as CPU is done in virtualization. This would lead to low throughput from single instance point of view. Another issues is that the users outside the cloud need high performance network. Assuming that a company transferred its internal data intensive systems to the cloud, they need a high performance connection to the user in order to work properly. While this is issue not directly related to cloud computing, it limits the deployment towards the cloud. [13]

3.2.3 Public cloud security

There is a lot of fear about public cloud security related issues [13]. In the cloud users share resources like memory, CPU and data storage. Virtualization takes care of CPU and memory privacy. Virtualization techniques, such as Xen, completely isolate the virtual environments from each other [17].

Data storage and transfer are more susceptible for an attack. Moreover, IaaS cloud offers free hands on what kind of software can be run on the cloud. In EC2, raw data access is protected by a Amazon proprietary disk virtualization layer. The layer wipes the disks before they are accessible by other users. To provide more security, AWS still recommends using an encrypted filesystem on virtual disks. [3]

3.2.4 Legislative

In any business, many applications use and store sensitive data about clients and users. The owners of these applications must obey the data privacy laws of the country they operate in. Some countries also have restrictions that prohibit storing customer data in some other countries. This somewhat breaks the aspect of cloud computing being a location independent computing model. EC2 has server regions in USA, Europe and Asia. This allows some flexibility while dealing with the legislative issues. [1][13][21]

On the other hand, cloud computing provider need a transfer of liability from them to the user. This means that the users doing illegal activities while using cloud should be held accountable. [13]

4 IMPLEMENTING A SOA CLUSTER IN THE CLOUD

Implementing a SOA cluster of existing Oracle components into a cloud was the second goal of the thesis. The purpose of the SOA cluster is to act as a SOA platform for SOA systems. These systems mostly consists of web services and BPEL processes. The purpose of these systems is entirely up to system developer. The platform acts as an enabler that makes a it easier to use SOA as a part of enterprise architecture.

SOA cluster would be configured in such a way that it would benefit from cloud computing. EC2 comes half way towards a developer in this and offers some benefits for free. Others require effort from the developer and support from whatever software is installed on the cloud instances. Oracle SOA Suite supports clustering, but it does not support dynamic scalability. Dynamic scalability is a key ingredient in order to achieve Rapid Elasticity in the cloud.

The first section describes the general information regarding the work. It includes information about the goal, software stack, cloud infrastructure and finally other tools and resources used during implementation. The second section lists and explains the tools that were used during the implementation process. Their general purpose is explained with the reasons why they were used for this project. The third section is more about the concrete work. It describes work done to get it all together and working properly. However, it focuses on specific details involved in the cloud computing environment rather than trying to explain every step involved in the setup process.

4.1 Description

This section presents the description about the implementation work involved in this thesis. This information is relevant for understanding the steps that were necessary for achieving the goal. In other words, the answer to the question 'what?' is answered here.

The first subsection explains the focus of the implementation. It is about getting a working SOA cluster on EC2. The second subsection then defines what the SOA cluster is like. It explains the relations between the cluster components. The third subsection explains shortly what EC2 is like. Only the components relevant to this project are explained. Then the final subsection maps together the previous two subsections. It explains how the SOA cluster configuration is implemented in the cloud.

4.1.1 Focus

The focus of the implementation is getting the SOA cluster running on a cloud computing infrastructure. This is fulfilled when a proper cluster configuration is running in EC2 cloud as explained in Subsection 4.1.4. The configuration must be able to run a Business Process Execution Language (BPEL) test application provided by NSN. Another requirement is to use the advanced features of cloud computing to enable scalability to the cluster. The required software configuration was determined by NSN to use Oracle SOA Suite and Oracle 11g database. These software products have some of their own requirements which are further explained in Section 4.3.

SOA cluster requires certain features from its environment in order to work in a correct manner. From the cloud computing point of view, to fulfil its promise, it needs something from the software stack as well.

While some of the positive cloud computing features come naturally just by using it, others do not. For example the environmental friendliness and pay-as-you-go type of operating costs are naturally involved. Other features, like dynamic scalability, need some work before they can be acquired. Those are very much dependent on the software stack's support for such features. It is important to find out what cloud features can be utilized while using the Oracle SOA stack.

4.1.2 SOA cluster configuration

At high level the Oracle 11g SOA stack consists of an administration server, Oracle SOA server and an Oracle database. The administration server runs Administration Console and Enterprise Manager applications that are used for maintenance and management. Oracle SOA server runs BPEL processes and web services. The database manages process persistence for asynchronous BPEL processes.

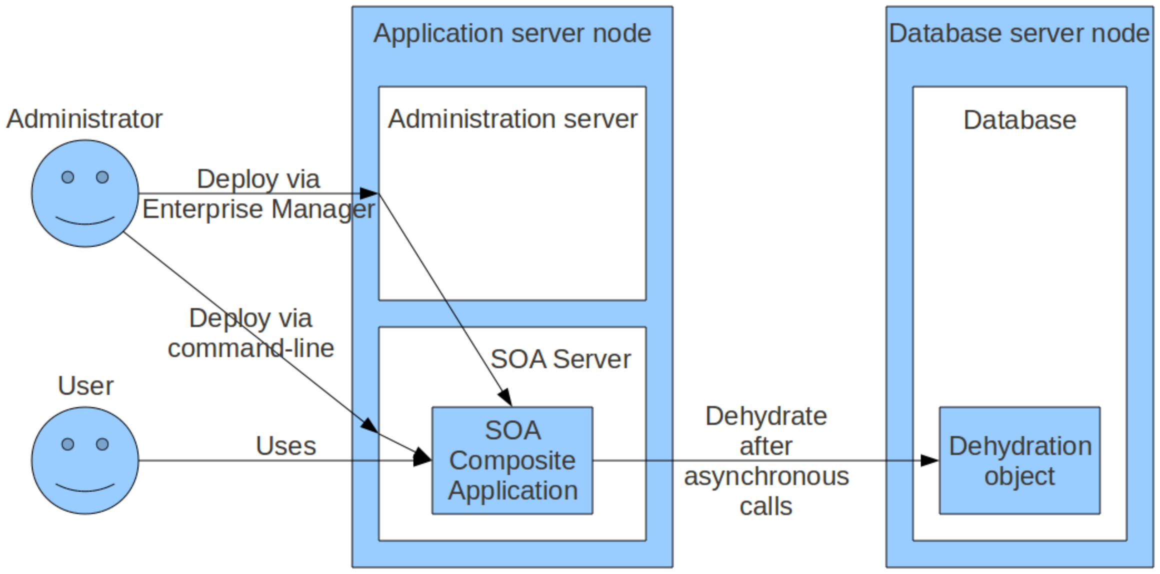

Figure 4.1: SOA configuration

Both the administration server and SOA server run on separate Weblogic application server instances. Weblogic application server is a JavaEE standard based application server. JavaEE, in turn, is an open standard for application servers that is designed for running web applications, in other words, applications that are designed to be used remotely. To deploy web applications or SOA packages, an administrator tool is required.

The administration server allows management options for SOA server. Most importantly the administration server provides a way to deploy, as shown in Figure 4.1, and undeploy *composite applications* to the SOA server. A composite application is a service container that may contain, for example web services and BPEL processes. These composite applications can be deployed with either command line or Enterprise Manager, which is a web based user interface.

SOA server is an application server that is designed to run composite applications. Figure 4.1 also shows that system administrator deploys the services to the SOA server for the users. The users can then utilize the composite application as they please.

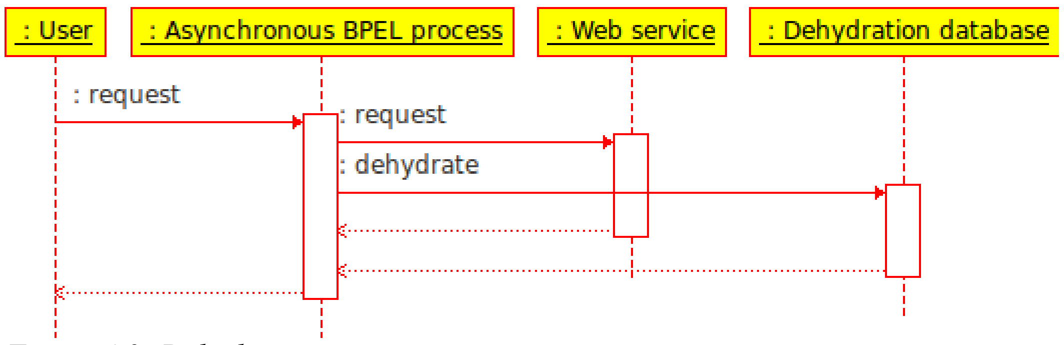

Figure 4.2: Dehydration process

In Figure 4.2 database works as a *dehydration* storage for BPEL processes. In other words, after making an asynchronous service call, a BPEL process is left waiting for a callback. While the process waits it is stored in the database. When the called process is done executing it will send a return message to a callback address. Once SOA server receives the return message, it revives the dehydrated process to receive the return message.

Unlike in Figure 4.1, the hardware configuration of the cluster consists of multiple application and database servers. The multitude of the servers adds reliability from high availability point of view and makes it possible to scale the size of the cluster. While using actual hardware this scaling requires manual work. In the EC2 these servers are represented by instances that run the given images.

4.1.3 Cloud computing infrastructure

EC2 is an IaaS cloud computing environment. It offers on-demand access to virtual servers that can be selected from pre-selected hardware configurations. Users can launch, use and shut down instances and are charged accordingly.

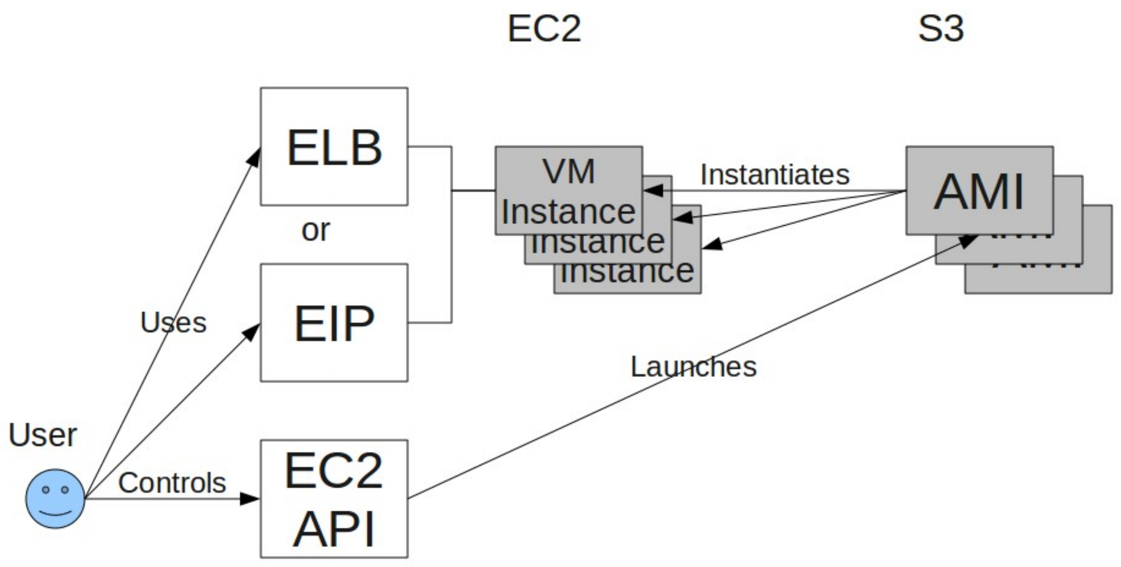

Figure 4.3: Parts of Amazon Web Services used in this project

As shown in Figure 4.3, Simple Storage Service (S3) is tightly connected to EC2. S3 holds the AMIs that EC2 uses for launching the instances. EC2 uses Xen virtualization on its server hardware. Xen hypervisor runs multiple instances hiding the underlying hardware from the user. The user uses EC2's web service interface to control the instances, for example, starting of stopping them. The web service uses Xen to control the instances. Web services also control Elastic Block Storage (EBS) drives. Each instance runs an operating system that runs the desired software.

Most instances are *ephemeral*, so the changes one makes to them during runtime will disappear when the instance is started again. Persistence can be supported in three ways. The first option is to use S3, a networked file storage that can be accessed across different EC2 sites. The second way is to use an EBS drive. It is a block device with customizable size. The EBS can be attached to an instance and mounted as a drive. The last way is to use an EBS bootable instance. This means that the instance is not ephemeral at all. However, a snapshot can be taken from EBS and used for restoring it to a previous state. This feature was added to EC2 at a late stage of the project and was not fully explored.

Amazon offers two 32-bit hardware configurations. They offer the lowest level of hardware as well as the cheapest price. Amazon also offers 64-bit hardware configurations, but they were not necessary for this project. 32-bit systems were selected to minimize the costs. A small instance is the cheapest. It offers processing power equivalent to a 1GHz x86 workstation processor and 1.7 GB of RAM. The other 32-bit system has two processors equivalent to 2.5GHz x86 workstation processor and the same amount of RAM. These two instance types were used during this project.

The software, like OS, can also be selected from AMIs offered by either Amazon or the EC2 community. Amazon offers a set of simple Linux and Windows configurations to start working on EC2. The community, on the other hand, offers a wide range of AMIs from generic to specific purposes. AMI can also be created by modifying existing AMI or built from scratch. Each approach was used in this project. To use a modified AMI, it is uploaded to S3 and declared as AMI using AWS Management Console or EC2 API tools.

4.1.4 Cluster configuration in EC2

The cluster is installed in EC2. The cloud environment offers the necessary tools for running the cluster. The configuration requires five instances while running with three AS nodes.

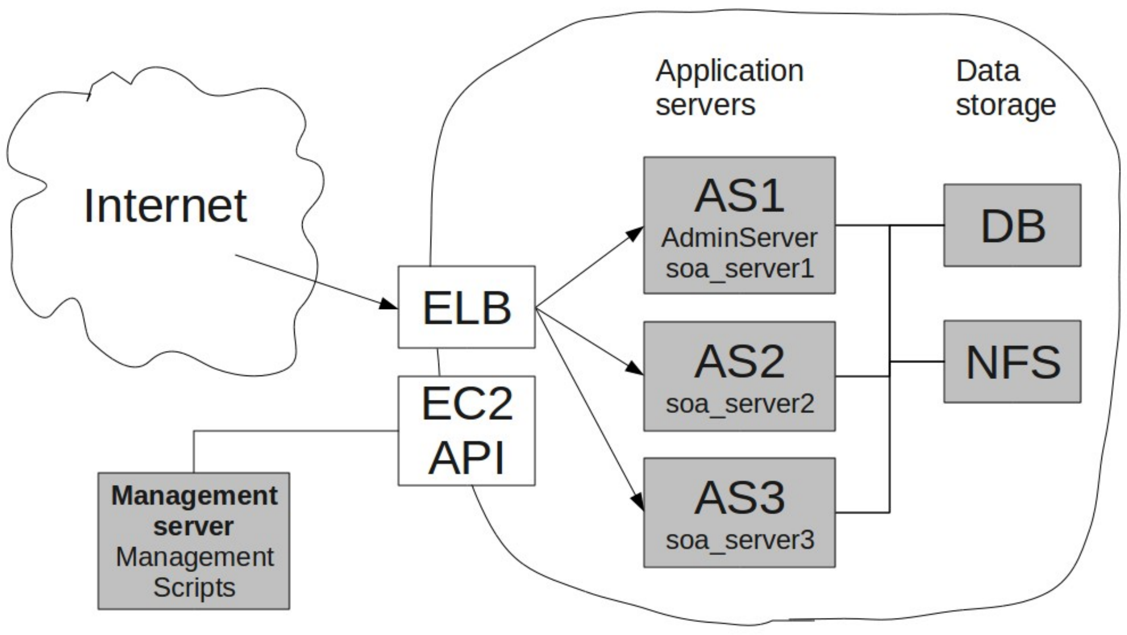

Figure 4.4: Cluster configuration in EC2

The configuration shown in Figure 4.4 consists of three application server nodes (AS1, AS2, AS3), a database node (DB), and a network filesystem node (NFS). The AS nodes share a NFS drive that contains binaries of the Weblogic software. They also share a DB for dehydration purposes. All these nodes are connected to each other via EC2 internal networking. Elastic Load Balancer (ELB) acts as an outside connection point for anyone accessing the applications running in the cluster. ELB distributes the load on AS nodes with a fault checking round robin algorithm.

Figure 4.4 also shows that the cluster is controlled by an external management server outside EC2. This management server is responsible for running management *scripts*, which are Python scripts created for this project to control the cloud environment. Management scripts control the cluster using EC2 API and by running direct commands on the AS nodes. Management scripts contain start up scripts and automated scaling control scripts.

4.2 Tools

Most tools used in this project are described in this section. These tools provide a way to operate the cloud computing environment over the network, operate AMIs and create automated scripts that start, stop and run instances. For each tool its purpose and how it was used in the project is described. Some tools are not described, because they were not essential for this project or were replaced by others.

The first subsection describes the AWS Management Console [4]. It is a Web graphical user interface (GUI) that is used in this project for controlling instances and EBS drives in EC2. Secondly, there are EC2 API and AMI tools [2]. These are command line tools for operating EC2 and manipulating AMIs. The third subsection explains Boto [5] and Python [19]. They are used for creating the management scripts. Next subsection describes the use of S3Fox [22]. It is an add-on for *Firefox*, an internet browser, that is used for loading AMIs and other files in and out of S3. The last subsection explains how Secure Shell (SSH) and Secure Copy (SCP) are used [14]. They are command line tools for secure remote operations and file transfer.

4.2.1 AWS Management Console

AWS Management Console is a web GUI application that controls EC2. It allows a user to manage most of the operations in the EC2 API. These operations include running and shutting down instances, creating and mounting EBS drives, and managing elastic IP addresses and load balancers. It is easy to use and GUI makes it a logical place to start learning EC2. [4]

In beginning of this project, the console was used for the instance management in EC2. This included mainly running and shutting down instances and using EBS drives. Later on the usage of the console decreased because of scripting. The scripts managed the start up and shut down process of the instances as well as managing the EBS drives. At that time the console was mainly used for observation.

4.2.2 EC2 API and AMI Tools

EC2 API and AMI Tools are command line tools that offer EC2 functionality. Input and output is done via typical command line interface. These tools are good for shell scripting and when commands are being run from a non-graphical environment. AMI Tools is usually run on a EC2 instance when creating new AMIs. [2]

API Tools were used only a little in this project. It was used in the beginning of the implementation to monitor EC2 and to register new AMIs. It was shortly replaced by Boto. AMI Tools on the other hand, were used untill the end of the project. It was used to package an instances and upload them to S3. [2]

4.2.3 Boto and Python

Boto is an integrated interface for AWS for Python programming language. Boto translates Python function calls into SOAP messages that are sent to EC2. Boto and Python offers the power of a programming language and the ability to run commands from command line. Compared to a regular command line tools, a programming language offers, for example, the use of objects as return values instead of having to rely on output parsing. [5] [19]

Boto and Python were used for command line purposes and scripting. Some of the scripts were just for helping the work on the command line while others were used as start up and management scripts for cluster automation. Scripts included starting individual server instances, mounting correct EBS drives, starting software on an instance, creating and managing configuration files, starting a cluster, and automated scaling.

4.2.4 S3Fox

S3Fox is an add-on for Mozilla Firefox browser. It is used for accessing S3 storage. It can be used to upload, download, and manage the contents of your S3 buckets. Managing in this sense includes re-factoring the directory layout and sharing content. [22]

In this project S3Fox was used for all the management in S3. The managing mostly involved removing old versions of AMIs. Another use was to share an AMI with other users.

4.2.5 SSH and SCP

SSH and SCP are widely used tools in secure communication between computers running Unix based OS. SSH is a tool to access a command line of another computer remotely. SCP on the other hand is a tool that allows copying files from one computer to another. [14]

These tools were used in their general purpose as well as in conjunction with scripts. The instances in EC2 were operated via SSH and binaries of the software were copied there via SCP. The scripts used SSH and SCP for much the same purpose as they were normally used. Scripts automated the launching of software remotely inside an EC2 instance. It was also necessary for scripts to automatically send files to EC2 instances via SCP.

4.3 Implementation

A scalable SOA Suite requires a clustered configuration. Cluster itself requires an administration server, a set of SOA servers, a shared file system for the SOA servers, and a database.

This section contains information of all the relevant work done in this project. Although most parts of the work was done in parallel, they are presented in separate logical parts. In other words, subsections and their content in this section does not follow an actual chronological order they were performed in.

The work is separated into six different subsections. Firstly a database is configured and start up scripts are created for it. The second subsection describes a OS selection and setup process for the application server. The third subsection is about setting up the application server. Next, the fourth subsection describes setting up a shared filesystem that is required for cluster configuration. The fifth subsection describes how the application server configuration was changed to enable a clustered configuration. The sixth subsection concentrates on EC2's load balancing and monitoring features. The seventh subsection describes a simple scaling automation. Many of these subsections refer to start up and management scripts. They are described in the eighth and ninth subsections, respectively.

4.3.1 Database server configuration

Setting up a database was simplified by the fact that Oracle had provided a pre made AMI that runs a Oracle Enterprise Edition 11g database on Oracle Enterprise Linux. However, the image required some configuration for it to be useful. The image had the software installed, but it did not contain the database.

Because AMI is ephemeral, data persistence had to be provided some way. EBS provided a way to persist the data. An EBS drive was mounted to the database instance and the database was installed onto it. SOA Suite also requires schemas that were installed to the database with Repository Creation Utility (RCU).

At this point the database was almost ready. EC2 environment circumstances were not dealt with yet. For example, mounting an EBS drive requires use of EC2 API. At this point Boto was taken to use to deal with the API. A script that ran on a non-cloud management server was used to start the database server, mount EBS drive and start the database instance.

There were also two issues about the dynamic IP addresses in EC2 that caused problems. The first issue is with application server finding the database. An elastic IP address was assigned to the database. Start up script automated the assignment whenever the database was started.

The other issue with addresses is that Oracle database has a listener process that assumes a static host name. The host name is set during the database setup and it does not change automatically when server host name changes. Start up script was configured so it will change this in the listener configuration file before starting the listener.

Once the database was working, final version of the database AMI was ready. Although, the start up script still needed modifications, the same AMI could be used. Start up script could make the necessary changes to the database configurations remotely each time the server starts.

4.3.2 Operating system for application server node

The first attempt was to install SOA Suite to *Fedora*, which is a Linux distribution derived from Red Hat Enterprise Linux (RHEL), further described in Subsection 5.1.2. Afterwards *CentOS* was given a chance. CentOS is an open-source clone of RHEL. SOA Suite is supported on RHEL, which was the desired target platform. However, RHEL was not available in EU region in EC2. CentOS is a clone of RHEL while Fedora is not entirely compatible. However, EC2 did not have supported AMI for CentOS. Although, EC2 community had some CentOS images, they did not appear to be clean

OS installations. They had some software installed on them.

A clean CentOS image was built from scratch. This was done inside EC2 using Fedora image, because it has Yum, Yellowdog Updater Modified. CentOS also uses Yum. This enabled the use of CentOS's Yum repository instead of Fedora's. Since Yum allows the installation of everything the operating system needs, it was possible to create an empty image and install CentOS into it via Yum. However, EC2 also requires their kernel modules to be present in each AMI. These modules are the modified Xen kernel modules mentioned in Subsection 2.1.3. They were downloaded from a EC2 site and installed into the image. An AMI was then created from this image with AMI tools. It became ready for usage after it was uploaded S3 and registered as an AMI in EC2.

4.3.3 Application server software

Setting up an application server started with starting up an empty CentOS image. At this stage a single node configuration of SOA Suite was going to be installed. SOA Suite was installed and configured as single server environment. This meant that Administration Console and SOA Suite was installed. Their connection to the database was through the elastic IP address.

The installation had two cloud computing related issues. The first of these was a prerequisite check that informed that the instance was running a kernel that was not supported. The warning was ignored and it did not cause further concerns.

The other problem emerged after the installation. The installation had configured the application server software with a host name that was in use at that time. Because EC2 uses Dynamic Host Configuration Protocol (DHCP) to give host names to the instances, it caused the host name to change when the instance was started again. This caused that the administration console and the SOA server were unable to communicate. The issue was fixed by pointing the references to localhost address and removing server certificate based authentication.

Similary to the database, a start up script was made for the application server. This script started the instance in EC2 with the help of Boto libraries. Then the scripts polls the EC2 service until the instance is running. Once it is, the start up script copies configuration scripts and files to the instance with the help of SCP. Then the start up script fires up the configuration script. The configuration script mounts the NFS drive and starts administration server and SOA server.

Although the single node configuration was not planned to be used for anything, it was thought that it could be useful for company internal courses using SOA Suite. Setting up servers into EC2 during a course would be easy and cheap. With a little effort, a course version of the single node server image was developed for testing. This testing raised an idea of a standalone single server training environment. That environment is further explained in Chapter 6.

4.3.4 Shared filesystem

A clustered Weblogic configuration requires a shared drive. NFS was chosen for this after the plan to use GFS, Global filesystem, on top of NBD, Network Block Device, had failed. The problem with GFS and NBD is described in Subsection 5.1.3.

NFS is a common protocol. This made it possible to use one of EC2's default Fedora based images could be used as a base NFS server. An instance was setup to automatically mount an EBS drive when started. Then it can share the NFS drive to other nodes. Those other nodes can automatically connect to this instance and mount the NFS drive over the network.

Although the image offered persistence, an additional EBS drive provided *persistent* storage, which means a storage that does not change back to the original state after being powered down, for the network share. This way the drive and the instance image could be restored separately in case of an error. This EBS drive was mounted to the server that shares the EBS drive over the network via NFS protocol. This server is accessed through Elastic IP address that is assigned to the server.

4.3.5 Clustering

Clustering the application servers started with starting up the single node AMI and removing all the Oracle software from it. Then the NFS drive was mounted to it. Instead of installing the software to the AMI's drive they were installed to the mounted NFS drive. This allows the configuration to have a shared filesystem that is required by the cluster configuration.

The cluster was configured with three application server instances. The first server would be running both Weblogic instances: Administration Console and SOA server. The rest of the application servers would only be running the latter. The configuration required host names to be given to each Weblogic instance. This is the address all Weblogic instances listen to and the Administration Console uses to connect to the SOA server instances.

To circle around EC2's dynamic host names, the host names were hard coded into `/etc/hosts` file with the corresponding servers' IP addresses. This solution had its own problems, because whenever a node was shut down and started again, hosts file needed adjustment. However, starting and shutting down the nodes was done with scripts with access to EC2, it was easy to generate new hosts file whenever an AS node was started. This file was then sent to each running node. It enabled nodes to resolve the host names given during start up. It also allowed to match the nodes IP address to the host name it is listening to.

Even with the host names in place, Administration Console was unable to properly deploy composite applications to SOA server instances. The applications were not deployed to the entire cluster, but only to the target server. That was caused by *Coherence,* which essentially is a distributed in memory data storage [16], which defaults to *multicast*, which is a type of data transmission with multiple recipients, when caching data. It was soon found out that EC2 does not support multicast. Coherence is used by Administration Console when deploying applications. This was confusing, because SOA Suite installer asks for the communication method and unicast was chosen. Oracle documentation has instructions for fixing this issues afterwards. The instructions were changing the Weblogic's parameters used for starting up Coherence. Apparently due to a software bug, the parameter configuration did not stop Coherence from using multicast. Coherence's configuration files were found inside `coherence.jar`. Application server host names were added to `tangosol-coherence.xml` as well-known-addresses (WKA). Repacking `coherence.jar` with modified configuration finally removed the multicast deployment issue. After his modification the applications were successfully deployed on all running nodes.

4.3.6 Load balancing and monitoring

To properly take advantage of EC2, the system needed to use some of the promises cloud computing offers. The plan was to implement a simple scaling algorithm to compensate for the load. Before any scalability could be achieved, the system needed load balancing and some information about the actual load on the application servers.

Load balancing was done with EC2's load balancer. The cluster start up script was modified to include launching a load balancer. Then when starting and stopping AS nodes they are added to or removed from the load balancer as end points. Load balancer uses simple round robin algorithm for actual balancing between the nodes.

When load balancer was in place, it was time to monitor load on the nodes. EC2 offers monitoring functionality that can be enabled for each node separately for extra fee. This feature offers monitoring of CPU, network IO and disk IO usages. Monitoring is only used with application server nodes to track their CPU usage in this cluster configuration.

4.3.7 Automated scaling

The scalability feature was implemented by hand instead of using EC2's Auto Scaling, because the cluster required running scripts outside EC2. Scalability feature automatically starts when the cluster is started. Scaling up is triggered whenever the cluster is running on at least 50% average load. At least one node is running on all times. This node runs the Administration Console and one instance of SOA server. The other nodes that only run SOA server are subject to scaling. When they are started they run at least 50 minutes until it is determined if they are still needed. EC2 charges for each started hour so running almost an hour costs the same as running for few minutes. The extra 10 minutes are left for SOA server's shut down script that waits for any running processes to finish.

Scaling down threshold is reached when the average load has been below 20% for the last 10 minutes on the cluster. This way there is a nice stable gap between when one node running at 50% capacity and running two nodes on 25% capacity. The gap is somewhat larger when adding a third node. The gap reduces the unnecessary shutting down and starting up routine when the load stays near the threshold.

4.3.8 Start up script

The development of the scripts started as an easier way to start up a cloud instance than the management console. The scripts were first created when setting up the database node to simplify the process. The database configuration requires setting up elastic IP address and setting the database listening address each time the instance is started. Doing it manually made no sense. So a script was created to automate this process. Later on steps were added to the script whenever they were needed. The script had started as a start up script, but it became a true management script when the automated scalability was introduced.

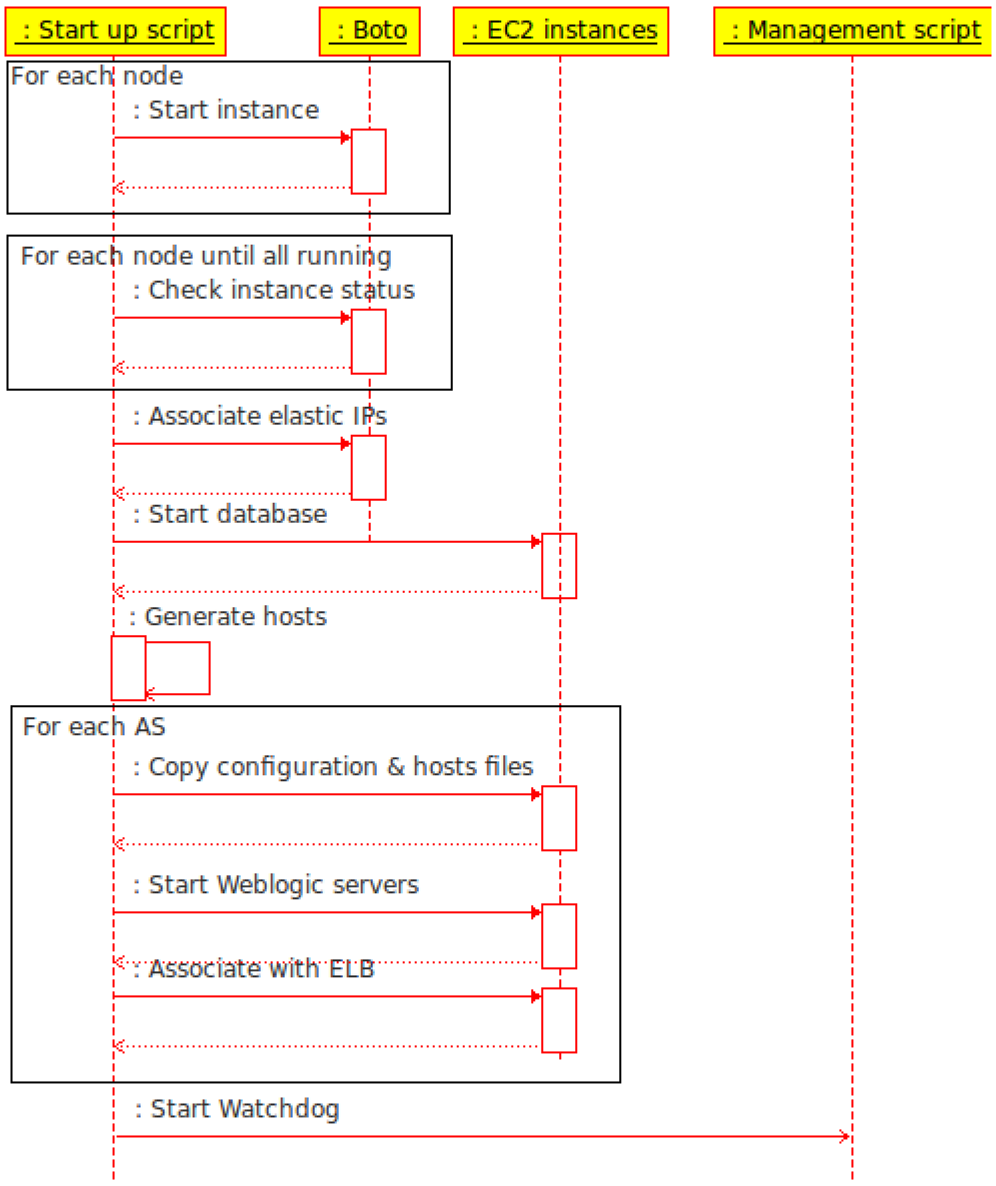

Figure 4.5: The start up script performs all the necessary routines needed to start the cluster and the management script.

Starting up the cluster configuration has many steps. The main steps of the start up script are included in Figure 4.5. It shows that start up script calls out EC2 via Boto interface to start up instances. This step starts up a NFS, database and up to 3 application server instances, depending on parameters. The script polls the status of each instance until they are all running. The database and NFS node use elastic IP addresses that will be associated with the instances. At this step the database is started. Weblogic servers use host names defined in hosts file to find other cluster members. Next stage of the script generates the hosts from the AS nodes' IP addresses. Then the script copies configuration files and hosts file to the AS nodes, starts Weblogic and associates the instances IP address with ELB. The last step is starting the management script known as Watchdog.

4.3.9 Management script

The management script is responsible for runtime scaling of the cluster. Depending on the load, the script either adds or removes AS nodes. However, it does not add nodes beyond three nor reduce to less than one.

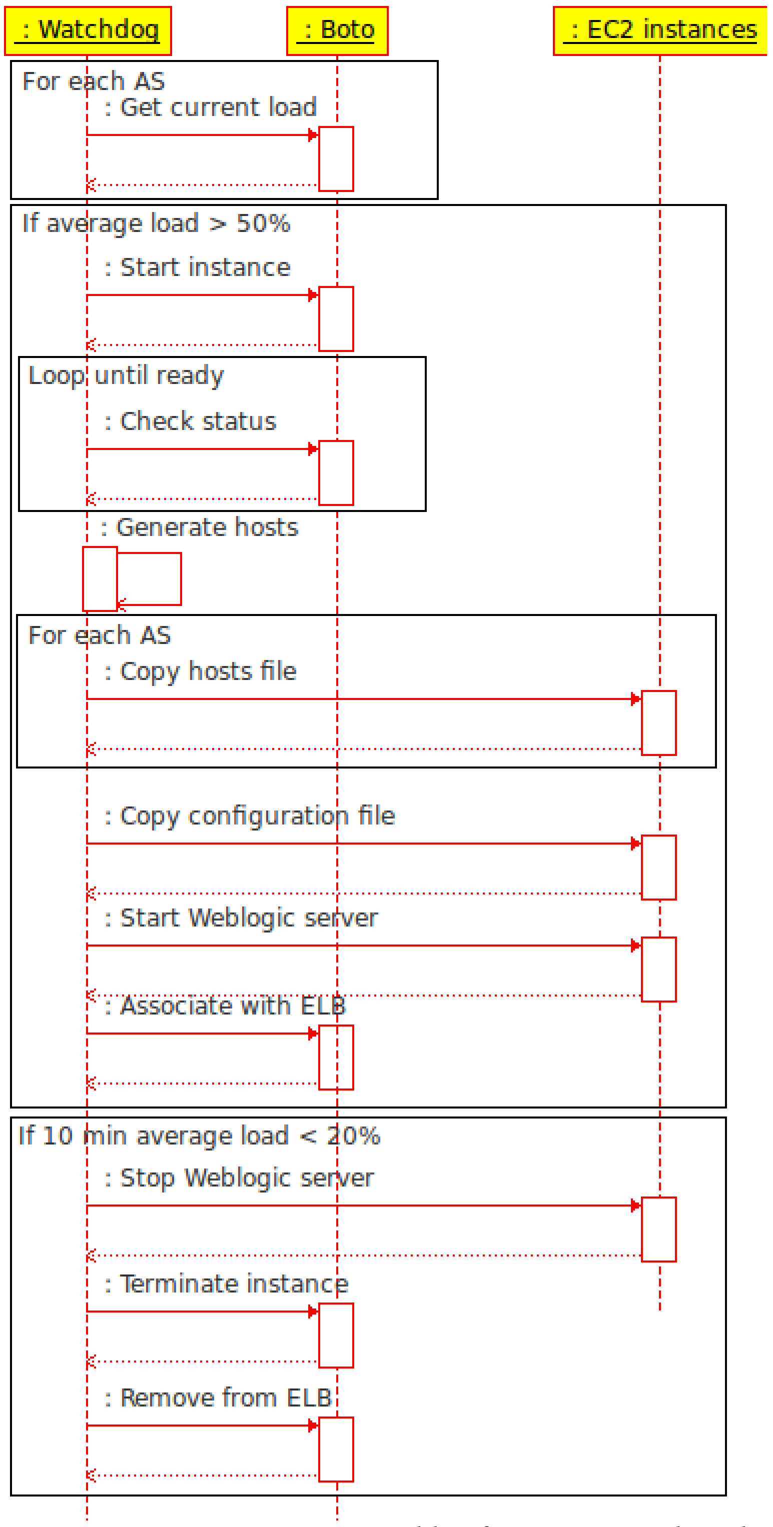

Figure 4.6: The Management script is responsible of automatic scaling during operations.

Management script shown in Figure 4.6 runs in an eternal loop. The loop runs through once every minute, because EC2 monitoring updates the instance load status once a minute.

The script gets the load of each AS instance and calculates the average. If the average load is above 50% and less than 3 AS nodes are running, another AS node is started. Then the script waits until the server is started. Now script generates new hosts file that contains the names and IP addresses of all the AS instances. This updated hosts file is then sent to each AS instance currently running. Script copies configuration files to the new instance and starts Weblogic server. When it is started, the server is associated with ELB that balances some of the cluster load to the server.

However, if the average server load has been below 20% for the last 10 minutes and more than one server is running, then it is reasonable to shut down additional servers. First the servers are removed from ELB and then Weblogic servers are shut down. After that it is safe to terminate the instance.

5 CLUSTER IMPLEMENTATION ISSUES AND FUTURE DEVELOPMENT

Implementing the cluster was not a straightforward process, because the distributed nature of the cluster caused some networking related issues. The software stack is clearly designed for traditional hardware configuration. For example, the configuration assumes that the IP addresses are static.

This chapter lists all the major cloud computing or EC2 related issues faced during the implementation. The first section lists abandoned approaches. These approaches were found misfit and were replaced with better solutions. Unsolved issues are described in the second section. The third section describes the limitations the configuration has. They limit the potential of the configuration in the cloud. The fourth section lists recommendations for future development. These are development time decisions that are not practical for future development.

5.1 Abandoned approaches

Abandoned approaches are parts of the work that were planned at first, but removed from implementation. The changes were due to compatibility issues or simply involving too much unanticipated work.

The listed approaches are in the order they appeared. The first subsection is about Oracle's RAC. This problem appeared in early planning. The rest appeared during implementation. The second subsection covers the change from Fedora to CentOS as an application server OS. The last one describes why NFS was chosen over GFS2 and NBD.

5.1.1 RAC Database

The original plan was to use Oracle RAC database. RAC FAQ's had info that RAC would not work in EC2 [15]. While the FAQ did not specify the reason, it was determined that the reason is the limitations in EC2's network capabilities. In addition to normal network connection, RAC requires a additional network connection between the network nodes. Because EC2 is limited to one network connection, this approach was abandoned and replaced by a Oracle 11g Enterprise Edition Database.

Oracle 11g Enterprise Edition Database is a single server database. It does not have database cluster capabilities. In other words, performance scalability and active high availability configuration are not available. Performance is now limited to the performance of a single EC2 instance. Performance scaling is limited to scaling up limits. So it can still be stretched somewhat via running higher performance instance.

5.1.2 SOA Suite on Fedora

During planning it was decided that Fedora would be used as an operating system for the application server. This would save time because EC2 had pre-made Fedora images. While SOA Suite is not supported on Fedora, it was thought that SOA Suite could run on it. Fedora appeared to have all the necessary packages that are required for using SOA Suite. Only one package was declared as deprecated. The functionality of that packages was contained in another packages and Fedora did not allow the installation.

SOA Suite's installer has a pre-check that complained about the deprecated package. This pre-check was skipped. The installer froze in the middle of the installation with no error messages. This approach was abandoned and the focus moved towards CentOS. While CentOS is also not supported it is a clone of RHEL which makes it compatible.

5.1.3 Network Block Device and Global filesystem

The cluster configuration required a shared filesystem. A plan was to use GFS2 with NBD. NBD would allow mapping of drive over the network. Then GFS2 would be used on it to allow multiple systems to access it without corrupting the filesystem.

First a NBD drive was setup. It was briefly tested with a single user configuration using ext3 filesystem. Setting up GFS2 for use proved to be more difficult. Running GFS2 required support from kernel. This is provided with kernel modules. Unfortunately, EC2's default kernel did not support this. It was determined that resolving this would take more time than using NFS as a replacement.

5.2 Open Issues

Some non-essential issues were not resolved during the implementation. They are listed in this section. The issues are described here with initial thoughts of why they appeared. The fixes for these issues were left out of the implementation, because they were not deemed essential for this proof of concept. The issues are listed here for possible future investigation.

The first issues is about a long timeout appearing sometimes when running a partial cluster. The other issue manifests itself in a form of an error message that appears when a previously undeployed component is redeployed.

5.2.1 Timeout in cluster configuration

When the cluster configuration is running with less than three nodes, some Administration Console's operations take several minutes when ran for the first time.

The same operations take no more than few seconds when running a full cluster.

Although no error messages are generated, this looks like a timeout related issue. Administration Console is probably trying to reach all three nodes and it waits until they respond or a timeout is reached. The timeout does not appear when all three servers are running.

5.2.2 Deployment error message

An error message appeared in Administration Console when deploying a composite application, which had been undeployed before, to the cluster. The error message first stated that the deployment had failed, but on the last line it declared that the deployment was successful. After closing the error message, the deployment application was left open and it could be exited by pressing cancel. The composite application deployed successfully regardless of the error message.

This error appeared after adding WKAs to Coherence's configuration file. Although it appeared at that time, it may not be related. This configuration is the only way the cluster deployment worked in EC2 with no point of comparison. Nevertheless, the issue was briefly looked into with no results.

5.3 Limitations

The implementation has some undesirable traits that limit its potential. For these limitations no solution was found during the implementation. These limitations limit the potential of the SOA cluster in EC2. They limit the size of the cluster, therefore limiting Rapid Scalability. The limitations did not prevent this project and were just noted as inconveniences.

The first subsection describes the problem with preconfigured application server cluster. The configuration limits the scalability of the cluster. The other subsection is about non-scalable database. The database does not offer a cluster configuration option.

5.3.1 Preconfigured application server cluster

The application server cluster configuration was preconfigured to use three nodes. The cluster can be used with less than the maximum amount of nodes. This allows a limited scalability. The automated scaling is explained in Subsection 4.3.7.

This configuration is the probable cause to issues described in Subsection 5.2.1. When the cluster is not running at full capacity, there are timeout issues. The other aspect of this configuration is that the scalability itself is limited to the preconfigured number of nodes. When running performance demanding applications, this configuration could cause a performance bottleneck. Chancing the configuration to include more preconfigured nodes lessens the chance of bottlenecking the system and allow a truly scalable system. However, this ensures the continuity of the timeout issues.

For proper use of the system, either the cause of the timeout or alternative scaling method for the cluster needs to be found.

5.3.2 Scalability of the database

Oracle enterprise edition is a non-scalable database. In other words it can only be run on a single instance. This version was used because the clustered database, RAC, did not work in EC2 as explained in Subsection 5.1.1.

The application server cluster has, with proper configuration, seemingly unlimited scalability. Meanwhile, the database works as dehydration storage for the SOA Suite cluster. A non-scalable database could cause a performance bottleneck for data intensive operations. A properly scalable configuration would need a scalable database.

5.4 Recommendations

There were some development decisions in the implementation phase that are not sensible for any production environment. These decisions have other known solutions that are better suited for production. The development decisions are explained here along with a possibly better solution.

First subsection describes the use of hosts file. This method should be replaced with a proper use of DNS service. Next subsection deals with reasons why cluster management scripts were ran outside the cloud. This creates an unnecessary dependency outside the cloud. The last subsection explains why regular AMIs were used instead of EBS based ones. Better performance and persistence could be gained by using EBS based images. These solutions were not implemented, because of limited time frame of the project.

5.4.1 Communication in DHCP environment

The use of generated hosts file with DHCP addresses was a simple way to get application servers to communicate. There was already a script for sending configuration files to servers, which made moving the hosts file easy. A custom hosts file for the servers was generated with python scripts on management server. The management server had access to Boto that provided the IP addresses for cluster application server nodes. Generating and replacing the hosts file every time a server is booted, removes the possibility of manually configuring the hosts file.

A sensible way to deal with DHCP related issues would be to configure a custom DNS-server. That DNS could resolve cluster address queries. For any other DNS-queries, that server can forward them to an actual DNS-server. A DNS server configuration removes the need to generate hosts file.

5.4.2 Running management scripts inside the cloud

During the implementation management scripts were kept and ran on a server outside the cloud. In the beginning of the implementation phase there were doubts on EC2's security. Running the management scripts requires EC2's access keys. These keys were kept on a remote server behind a closed firewall. The management scripts were used to start up and manage the EC2 instances.

Running management scripts on a node outside the cloud makes the configuration dependent on that node. Because the management scripts constantly monitor the nodes, interruptions in connection are undesirable. Moving the control to the cloud also removes maintenance need for the external server. The cluster still needs to be started externally. This can still be done via scripting, but that script should only start the node that runs the management script. Then the management script kicks in and manages the nodes. This way the runtime dependency to an external node is removed.

5.4.3 Using EBS based AMIs

During the implementation phase AWS introduced EBS based AMIs. This type of AMI offered several advantages over the S3 based AMI. Most importantly it offers persistence. The nodes can be stopped without losing the changes made while running. That makes development easier, because new images do not have to be created whenever an instances is stopped in order to persist the changes.

The other aspect is start up time. The EBS based AMI starts considerably faster than the traditional S3 based AMI. When S3 based AMI is started, the drive image is first copied from S3 to a temporary drive in EC2. EBS drive already exist on the cloud's local filesystem and is therefore considerably faster to start. Actual IO performance was not tested.

The implemented system used S3 based AMIs for each node except for NFS node. That node was implemented last. At this stage EBS based AMIs had become available and it was taken into use. Converting the other nodes to use EBS was out of scope of this project. However, that is recommended for future development.

6 SOA TRAINING ENVIRONMENT

After implementing the cluster configuration, there was a need for a SOA training environment. It was decided that cloud computing could be used for this purpose. The benefit of this is that the trainees would not need to install the whole software stack to their laptops. The installation is a time consuming process and the software has high system requirements.

The first section describes the purpose of the training environment. It fills a need for a pre-configured training environment. The second section describes the environment and its components. The training environment consists of almost the same software stack as the cluster environment with few differences. The third section explains how the environment was implemented. As the process was similar to the cluster, only the different parts are noted here.

6.1 Purpose

During the implementation a need arose for a SOA training environment, where trainees learn to use Oracle SOA stack. NSN has training events for the SOA project that this thesis is for. A common problem these events is that the trainees use the software stack on their own laptop. This situation has some drawbacks such as high system requirements and time consuming installation.

The high system requirements are imposed on the trainees laptop by the Oracle SOA stack. Also in addition to database, application server, and administration server the system needs to run *Jdeveloper*, which is a multi-purpose integrated development environment. In these trainings, Jdeveloper is used for creating the SOA applications. This software configuration requires roughly 3 gigabytes of RAM, somewhat modern CPU and several gigabytes of hard drive space to run smoothly. The requirements, mostly the RAM, are simply too much for many laptops the trainees have.

To overcome this obstacle, cloud computing would be used to provide the necessary hardware requirements while using the trainees laptop merely as a terminal to access the system via remote desktop software. The cloud computing SOA training environment would have the software stack pre-installed and configured. The trainees would only have to install and configure the remote desktop software on their laptops.

6.2 Description

The training environment has its software stack installed on a single large EC2 instance, because it has enough RAM to run the whole stack. Even though large instances are not the cheapest around, the training costs are still manageable, because the instances are only needed for during the training period.

The training environment was developed on an EBS based AMI. This is important, because it makes possible to shut down the instances without losing any data. The training may last several days so cutting the expenses during the night is also vital.

The software stack consists mostly the same software that was used in the cluster environment with two exceptions. The similar parts of the configuration are single instances of Oracle 11g Database, Oracle Weblogic and Oracle SOA suite. The first exception is that Red Hat Enterprise Linux was used as an OS for the environment. The RHEL image was still unavailable in EC2 EU region, but at this point getting it there was there was given a try. The other exception was the plan to use FreeNX server, a remote desktop server application. This server allows the trainees to utilize the training environment via NX client application.

6.3 Base system implementation

The implementation started by researching the RHEL instance offerings in US region. However, it was soon discovered, that the official RHEL image had two major flaws. It would not work in EU region even if it was transferred there, because its use is blocked outside US region due to licensing. The block however only applies to the official supported version of RHEL in EC2.

The other option was to build RHEL image outside the cloud and then upload it there. RHEL was installed on local hardware, which was then used to create an AMI. EC2 tools offer an option to convert a locally running Linux into an AMI. This option was used for the locally running RHEL installation. With some configuration, including installing the kernel modules as in Subsection 4.3.2, this image was ready to be transferred to S3. Once the image was uploaded to S3, it was registered as an AMI.

As the image was a regular S3 based AMI, it needed to be transformed into EBS based AMI. A new and large enough EBS drive was created for the task. The transformation was done by running the S3 based AMI as an instance and then synchronising its data into the EBS drive. To actually make the EBS based AMI usable a snapshot was taken from the drive. The snapshot was then registered as bootable AMI.

6.4 Installing the software stack

The next step began by running the newly created EBS based AMI. FreeNX server was installed to allow remote desktop usage. This was done early in order to use desktop to download and install the rest of the required software more easily.

The rest of the software stack was downloaded from the Oracle website. There were some minor differences compared to the cluster installation process. For this configuration the database software was installed locally. The step included installing the software and creating the database. As the system was installed on a persistent EBS drive, the database could also be placed on that drive.

The Weblogic and SOA Suite software were installed mostly in the same manner as for the cluster configuration. However, this configuration needed to be configured as a single server instance, unlike the three server cluster. Another difference was the database reference which referred to localhost address, because the database was running on the same instance.

The Jdeveloper IDE was then installed without issues. In order to develop SOA projects the Jdeveloper requires a plug-in. In order to help the developers, the plug-in allows most of the SOA XML-files to be created and edited with graphical tools. The plug-in was installed and the connection between Jdeveloper and the application server was set up. Additionally, the training required some sample data and configuration for the Weblogic server and the database. These were configured into the final training environment image to save time from the training.

6.5 Benefits and drawbacks

The benefits of using a cloud computing environment for temporary tasks like this are quite clear. The users are ensured to have always enough resources for the training, which may otherwise prove difficult. Trainings in particular, usually requires the trainees to use their laptops which tend to have some limitations.

In the cloud computing scenario, the trainees only need a laptop with enough computing power to run a lightweight remote desktop application and a board band internet connection. The environment is provided by a trainer who gives access rights for the trainees.

Compared to installing a virtual machine and running the pre-configured image, the cloud computing approach requires less time to install. In the cloud computing scenario, the trainee only installs the NX client software compared to installing the whole software stack or a virtualization environment that includes the software stack.

The cloud computing training environment is not without cons. Running multiple training environments is not free of charge. Although, it is most likely cheaper to use the cloud than acquire such hardware by other means. In addition, there is also an actual drawback. A nasty latency which hinders the user experience. As in this case the cloud computing servers were running in another country, this causes some unwanted latency.

7 CONCLUSIONS

The first goal of the project that theoretical part of cloud computing was to be grasped at high level with the focus on the aspects surrounding EC2 cloud. The second goal, the implementation part, was to have a working SOA cluster in EC2. The cluster is scalable and consists of multiple SOA application servers and multiple databases. This cluster runs in EC2 and benefits from cloud computing by utilizing rapid elasticity and environmentalism while minimizing costs.

Next, the first section determines what parts of the goal were successfully implemented. The goals of the cluster environment are described and the success of their implementation is estimated. The second section deals with the benefits gained from cloud computing. These benefits are viewed from the perspective of this micro project. The third section explains experienced cloud computing drawbacks. The last section is about final thoughts on cloud computing. It paints a cloudy vision for the future of cloud computing.

7.1 Achieving the goal

Many aspects of cloud computing were given thought. The benefits of cloud computing were identified and explained. Most notably cloud computing offers flexibility that is reflected as lowered costs and ease of use. The drawbacks mostly restrict special scenarios and raise some security and legislative concerns. Legislation permitting, the benefits of cloud computing outweigh its drawbacks.

The desired software stack was installed with the exception of Oracle RAC database, which was replaced with Standard edition database. The main installed components included SOA Suite, Weblogic application server, and Oracle Database. Each of these components required some special consideration to work properly in the cloud. Single server test configuration appeared work without issues.

The cluster configuration prototype was more or less successfully installed on EC2 cloud. The configuration can run the test application, but it still has some undesirable limitations. For future purposes the issues listed in this thesis need to be addressed. Nevertheless, the configuration could demonstrate benefits of cloud computing.

7.2 Gained cloud computing benefits

This project's implementation could utilize the benefits of cloud computing quite well. The project itself may not have been possible without pay-as-you-go payment method in

EC2. The setup of a multi node environment would have required some company internal assistance even if the hardware would have been ready or virtual space would be available. In this cloud computing scenario the project just needed acceptance to cover the relatively small costs of running a up to five instance cluster in EC2.

The implemented cluster gained limited benefit from rapid elasticity. The resources for development were immediately at disposal and the cluster development began. The cluster was configured in a way that it could be automatically, albeit limitedly, scaled. The cluster scales out during high load and shrinks back to one application server configuration when the load dissipates. The scaling is limited by issues faced during implementation. For the training environment, rapid elasticity is the most important aspect. It makes the relatively short training periods possible and cost effective.

Community support was experienced when setting up a database. The community provided a pre-made Oracle database AMI that only needed configuration. This helped to save a lot of time.

7.3 Experienced cloud computing drawbacks

The implementation faced some issues that can be noted as cloud computing related. These issues did not stop the project from progressing, but they did cause delays. The issues were encountered at different stages of the project.

One performance related issue was that EC2's hardware is also somewhat heterogeneous. This was noticed by chance while testing the cloud environment. Two instances were started with the same configuration. One of them was running on a AMD Opteron CPU and the other one was running on Intel Xeon CPU. The performance of the two systems were tested later with `super_pi` benchmark. Although the synthetic benchmark does not properly reflect the performance in real applications, it gives some hint on the performance variation between the CPUs. According to the benchmark, the Intel CPU has roughly twice the performance of the AMD CPU. Appendix 1 contains outputs of `super_pi` on two different instances. Appendix also has an output of the CPU architecture the system is running.

The problem with Oracle RAC database described in Subsection 5.1.1 was clearly hardware related. The EC2 could not be configured with multiple networks. This issue could not be solved which lead to choosing a non-clustered database. Another hardware related problem was the lack of multicast support in EC2 as described in Subsection 4.3.5. While the issue could be circumvented in a relatively easy manner, finding out the root cause of the problem took its time.

7.4 Final thoughts

It remains to be seen, if this implementation or others like it will truly invade the clouds. As for cloud computing, its benefits will outweigh the drawbacks. For small to medium sized companies using public cloud computing will reduce costs. They benefit from not having to obtain server-side hardware support. Meanwhile, big companies, that can build cost effective large computing systems, have the opportunity to use private centralized cloud computing environments that have low per server maintenance costs. There are some issues preventing the use of some software in the cloud. These issues will be cleared out with time as cloud computing providers work out the few restrictions they may have and the software providers play along to make their products cloud capable.

The future of cloud computing will no doubt smooth some of the rough edges cloud computing still has. This will make the hardware invisible and at the same time remove most restrictions the current implementations of cloud computing have. However, at that time cloud computing has been prominent part of computing for so long, maybe the utility computing will come true and the term "cloud computing" becomes meaningless.

The utility computing vision brings cloud computing also to homes. Desktop PCs will be replaced by low-maintenance cloud computing terminals. These terminals offer a desktop computing through broad band access to a VM running in a cloud.

REFERENCES

1. Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/, 27.7.2010
2. Amazon Web Services Developer Community, http://developer.amazonwebservices.com/, 18.8.2010
3. Amazon Web Services, http://aws.amazon.com/, 27.7.2010
4. AWS Management Console, http://aws.amazon.com/console/, 18.8.2010
5. boto, http://code.google.com/p/boto/, 18.8.2010
6. Daniel Nurmi, Rich Wolski, Chris Grzegorczyk , Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov , The Eucalyptus Open-source Cloud-computing System, *CCGRID Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 124-131
7. Google App, http://code.google.com/appengine/, 27.7.2010
8. Hakan Erdogmus, Cloud Computing: Does Nirvana Hide behind the Nebula? , IEEE Software, vol. 26 no. 2, March/April 2009 , pp. 4-6
9. Hao He, What Is Service-Oriented Architecture, http://www.xml.com/, 2003, 27.7.2010
10. Katarzyna Keahey, Mauricio Tsugawa, Andrea Matsunaga, and Jose A.B. Fortes, Sky Computing, *IEEE Internet Computing,* Volume 13, Issue 5, 2009, pp. 43-51
11. Lavanya Ramakrishnan and Dennis Gannon, A Survey of Distributed Workflow Characteristics and Resource Requirements, Department of Computer Science, School of Informatics , Indiana University, Bloomington, IN , 2008
12. Marios D. Dikaiakos, George Pallis, Dimitrios Katsaros, Pankaj Mehra, Athena Vakali, Distributed Internet Computing for IT and Scientific Research, *IEEE Internet Computing,* Volume 13, Issue 5, 2009
13. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia , Above the Clouds: A Berkeley View of Cloud Computing, UC Berkeley Reliable Adaptive Distributed Systems Laboratory, February 10, 2009
14. OpenSSH, http://openssh.org/, 18.8.2010
15. Oracle Cloud Computing Center, http://www.oracle.com/technology/tech/cloud/ , 27.7.2010
16. Oracle Coherence, http://www.oracle.com/technology/products/coherence/ , 27.7.2010
17. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer†, Ian Pratt, and Andrew Warfield , Xen and the Art of Virtualization, *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003, pp. 164-177
18. Peter Mell and Tim Grance, The NIST Definition of Cloud Computing, National Institute of Standards and Technology, Information Technology Laboratory,

Version 15, June 10, 2009

19. Python Programming Language, http://python.org/, 18.8.2010
20. Rajkumar Buyya and Anthony Sulistio, Service and Utility Oriented Distributed Computing Systems: Challenges and Opportunities for Modeling and Simulation Communities, 41st Annual Simulation Symposium, 13-16 April 2008, pp. 3-3, 2008
21. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic, Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, *Future Generation Computer Systems*, Volume 25, Issue 6, June 2009, Pages 599-616
22. S3Fox Organiser, http://s3fox.net/, 18.8.2010
23. Salesforce, http://www.salesforce.com/, 27.7.2010
24. Shyam Kumar Doddavula and Amit Wasudeo Gawande , Adopting Cloud Computing: Enterprise Private Clouds , SETLabs Briefings , VOL 7 NO 7 , 2009
25. Stavros Harizopoulos, Mehul A. Shah, Justin Meza, Parthasarathy Ranganathan, Energy Efficiency: The New Holy Grail of Data Management Systems Research, HP Labs and UCLA, 2009
26. Yefim V. Natis, Service-Oriented Architecture Scenario, Gartner Inc, 2003

APPENDIX 1: PERFORMANCE TESTS

First EC2 performance test on a small instance

```
[root@ip-10-224-42-99 ~]# ./super_pi 20
 Version 2.0 of the super_pi for Linux OS
 Fortran source program was translated into C program with version
19981204 of
 f2c, then generated C source program was optimized manually.
  pgcc  3.2-3  with  compile  option  of  "-fast  -tp  px  -Mbuiltin
-Minline=size:1000  -Mnoframe  -Mnobounds  -Mcache_align  -Mdalign
-Mnoreentrant" was used for the
 compilation.
 ------ Started super_pi run : Mon Jun 7 07:08:14 EDT 2010
 Start of PI calculation up to 1048576 decimal digits
 End of initialization. Time=       0.376 Sec.
 I= 1 L=       0        Time=       1.228 Sec.
 I= 2 L=       0        Time=       1.408 Sec.
 I= 3 L=       1        Time=       1.400 Sec.
 I= 4 L=       2        Time=       1.404 Sec.
 I= 5 L=       5        Time=       1.420 Sec.
 I= 6 L=      10        Time=       1.504 Sec.
 I= 7 L=      21        Time=       1.500 Sec.
 I= 8 L=      43        Time=       1.468 Sec.
 I= 9 L=      87        Time=       1.476 Sec.
 I=10 L=     174        Time=       1.428 Sec.
 I=11 L=     349        Time=       1.408 Sec.
 I=12 L=     698        Time=       1.404 Sec.
 I=13 L=    1396        Time=       1.416 Sec.
 I=14 L=    2794        Time=       1.412 Sec.
 I=15 L=    5588        Time=       1.496 Sec.
 I=16 L=   11176        Time=       1.500 Sec.
 I=17 L=   22353        Time=       1.480 Sec.
 I=18 L=   44707        Time=       1.336 Sec.
 I=19 L=   89415        Time=       1.272 Sec.
 End of main loop
 End of calculation.    Time=      28.458 Sec.
 End of data output.    Time=       0.132 Sec.
 Total calculation (I/O) time=      28.590 (       1.396) Sec.
 ------ Ended super_pi run : Mon Jun 7 07:09:20 EDT 2010

[root@ip-10-224-42-99 ~]# cat /proc/cpuinfo | grep model
model       : 65
model name  : Dual-Core AMD Opteron (tm) Processor 2218 HE
```

Second EC2 performance test on another small instance

```
[root@ip-10-227-103-162 ~]# ./super_pi 20
 Version 2.0 of the super_pi for Linux OS
 Fortran source program was translated into C program with version
19981204 of
 f2c, then generated C source program was optimized manually.
  pgcc  3.2-3  with  compile  option  of  "-fast  -tp  px  -Mbuiltin
-Minline=size:1000  -Mnoframe  -Mnobounds  -Mcache_align  -Mdalign
-Mnoreentrant" was used for the
 compilation.
 ------ Started super_pi run : Mon Jun 7 07:21:55 EDT 2010
 Start of PI calculation up to 1048576 decimal digits
 End of initialization. Time=       0.228 Sec.
 I= 1 L=       0        Time=       0.644 Sec.
 I= 2 L=       0        Time=       0.736 Sec.
 I= 3 L=       1        Time=       0.776 Sec.
 I= 4 L=       2        Time=       0.736 Sec.
 I= 5 L=       5        Time=       0.740 Sec.
 I= 6 L=      10        Time=       0.744 Sec.
 I= 7 L=      21        Time=       0.728 Sec.
 I= 8 L=      43        Time=       0.760 Sec.
 I= 9 L=      87        Time=       0.732 Sec.
 I=10 L=     174        Time=       0.728 Sec.
 I=11 L=     349        Time=       0.752 Sec.
 I=12 L=     698        Time=       0.732 Sec.
 I=13 L=    1396        Time=       0.736 Sec.
 I=14 L=    2794        Time=       0.732 Sec.
 I=15 L=    5588        Time=       0.732 Sec.
 I=16 L=   11176        Time=       0.720 Sec.
 I=17 L=   22353        Time=       0.748 Sec.
 I=18 L=   44707        Time=       0.696 Sec.
 I=19 L=   89415        Time=       0.648 Sec.
 End of main loop
 End of calculation.    Time=      14.605 Sec.
 End of data output.    Time=       0.088 Sec.
 Total calculation (I/O) time=      14.693 (       1.096) Sec.
 ------ Ended super_pi run : Mon Jun 7 07:22:35 EDT 2010

[root@ip-10-227-103-162 ~]# cat /proc/cpuinfo | grep model
model       : 23
model name  : Intel (R) Xeon (R) CPU           E5430  @ 2.66GHz
```