Analysis and Performance Optimization of E-mail Server

Dissertação apresentada ao Instituto Politécnico de Bragança para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Sistemas de Informação, sob a supervisão de Prof. Dr. Rui Pedro Lopes.

Eduardo Manuel Mendes Costa
Outubro 2010

Preface

The e-mail service is increasingly important for organizations and their employees. As such, it requires constant monitoring to solve any performance issues and to maintain an adequate level of service. To cope with the increase of traffic as well as the dimension of organizations, several architectures have been evolving, such as cluster or cloud computing, promising new paradigms of service delivery, which can possibility solve many current problems such as scalability, increased storage and processing capacity, greater rationalization of resources, cost reduction, and increase in performance.

However, it is necessary to check whether they are suitable to receive e-mail servers, and as such the purpose of this dissertation will concentrate on evaluating the performance of e-mail servers, in different hosting architectures. Beyond computing platforms, was also analze different server applications. They will be tested to determine which combinations of computer platforms and applications obtained better performances for the SMTP, POP3 and IMAP services. The tests are performed by measuring the number of sessions per ammount of time, in several test scenarios.

This dissertation should be of interest for all system administrators of public and private organizations that are considering implementing enterprise wide e-mail services.

Acknowledgments

This work would not be complete without thanking all who helped me directly or indirectly to complete it.

Firstly I would like to thank Prof. Rui Lopes, for his knowledge, his help, advices and all the support.

I also thank Prof. José Rufino for making available the resources of the Laboratory for Computer Science, necessary for establishing the test scenarios.

I also would like to thank Prof. Tiago Pedrosa for all the support and advices.

My final thanks go to God and to my family: my wife for all the support, patience and encouragement you always gave me, which helped me in difficult moments. To my parents for all the support and encouragement to finish this dissertation.

Abstract

Nowadays the use of electronic services and Internet communications are increasingly common among citizens and thus the demand for better services and better solutions is constantly growing. In recent years we have seen the emergence of new infrastructures and computing platforms as well as the improvement of the existing ones. The need to improve services and electronic communications is compelling and it requires constant monitoring and studying new solutions towards new infrastructures and platforms.

To cope with the increase of traffic as well as the dimension of organizations, several architectures have been evolving, such as cluster or cloud computing, promising new paradigms of service delivery, which can possibility to solve many current problems such as scalability, increased storage and processing capacity, greater rationalization of resources, cost reduction, and increase in performance. However, there it is not clear if they are suitable to host e-mail servers. In this dissertation we perform the evaluation of the performance of e-mail servers, in different hosting architectures. Beyond computing platforms, was also analyze different server applications.

Tests were run to determine which combinations of computer platforms and applications obtained better performances for the SMTP service and for services POP3/IMAP. The tests are performed by measuring the number of sessions per ammount of time, in several test scenarios. We used Qmail and Postfix as SMTP servers and Qmail, Courier and Dovecot for POP and IMAP services.

Resumo

Nos dias de hoje, o uso de serviços de comunicações electrónicas e de Internet é cada vez mais comum entre os cidadãos. Neste sentido, a demanda por melhores serviços e melhores soluções está em constante crescimento. Nos últimos anos tem-se assistido ao aparecimento de novas infra-estruturas e plataformas de computação, bem como a melhoria das já existentes. A constante necessidade de melhorar os serviços e comunicações electrónicas exige um constante acompanhamento e estudo de novas soluções para novas infra-estruturas e plataformas.

Para lidar com o aumento do tráfego, bem como a dimensão da organizações, várias arquitecturas foram evoluindo, tais como o *cluster* ou *cloud computing*, promissores de novos paradigmas de prestação de serviços, que podem possibilitar a resolução de muitos dos problemas actuais, tais como escalabilidade, maior armazenamento e capacidade de processamento, uma maior racionalização de recursos, redução de custos e aumento no desempenho. No entanto, não está claro se estes estão adequados para os servidores de e-mail. Nesta dissertação realizamos a avaliação do desempenho dos servidores de e-mail em diferentes arquitecturas. Para além das plataformas de computação, também foram analisadas diferentes aplicações servidoras.

Foram realizados testes para determinar que combinações de plataformas de computação e aplicações obtêm melhor desempenho para o serviço SMTP e para os serviços POP3/IMAP. Os testes são realizados através da medição do número de sessões por quantidade de tempo, em vários cenários de teste. Optou-se por usar o Qmail e o Postfix como serviço de SMTP e servidores Qmail, Courier e Dovecot para os serviços POP e IMAP.

Contents

List of Tables

List of Figures

List of Acronyms

ACL Access Control List

AIX Advanced Interactive eXecutive

AMI Amazon Machine Image

AMS Andrew Messaging System

API Application Programming Interface

ARP Address Resolution Protocol

AWS Amazon Web Services

AoE ATA over Ethernet

BSD Berkeley Software Distribution

BOINC Berkeley Open Infrastructure for Network Computing

CDB Constant DataBase

CC Cluster Controller

CLC Cloud Controller

CPU Central Processing Unit

CRAM Challenge-Response Authentication

CRM Customer Relationships Management

CaaS Communication as a Service

DKIM DomainKeys Identified Mail

DR Direct Routing

DNS Domain Name System

DaaS Data Storage as a Service

EBS Elastic Block Storage

EC2 Elastic Compute Cloud

FTP File Transfer Protocol

GAE Google App Engine

GB Gigabyte

GPL General Public License

GRAM Grid Resource Access and Management

GRIP Grid Resource Information Protocol

GSSAPI Generic Security Services Application Program Interface

HACMP High Availability Cluster Multiprocessing

HA High-Availability

HP-UX Hewlett Packard UniX

HPC High-Performance Computing

HTTP Hypertext Transfer Protocol

IMAP Internet Message Access Protocol

IP Internet Protocol

IPL IBM Public License

IRC Internet Relay Chat

iSCSI Internet Small Computer System Interface

IT Information Technology

IaaS Infrastructure as a Service

JDBC Java Database Connectivity

KPOP Kerberized Post Office Protocol

KVM Kernel Based Virtual Machine

LAN Local Area Network

LDAP Lightweigh Directory Access Protocol

LVS Linux Virtual Server

MB Megabytes

MD5 Message-Digest Algorithm 5

MDA Mail Delivery Agent

MHS Message Handling Services

MIME Multipurpose Internet Mail Extensions

MIT Massachusetts Institute of Technology

MSA Mail Submission Agent

MTA Mail Tranfer Agent

MUA Mail User Agent

MX	Mail Exchanger
NAS	Network-Attached Storage
NAT	Network Address Translation
NC	Node Controller
NFS	Network File System
NNTP	Network News Transfer Protocol
NTP	Network Time Protocol
OS	Operating System
OSI	Open Systems Interconnection
OTP	One-time Password
PAYG	Pay as you Go
PCRE	Perl Compatible Regular Expressions
POP	Post Office Protocol
PaaS	Plataform as a Service
QMQP	Quick Mail Queuing Protocol
QMTP	Quick Mail Transport Protocol
QoS	Quality of Service
RAC	Real Application Clusters
RAM	Random Access Memory
REST	Representational State Transfer
RFC	Request for Comments
S3	Simple Storage Service
SAN	System Area Network
SASL	Simple Authentication and Security Layer
SC	Storage Controller
SDK	Software Development Kit
SLA	Service Level Agreements
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
SSO	Single Sign On
STD	Internet Standard

SaaS	Software as a Service
TB	Terabyte
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UPS	Uninterruptible Power Supply
VIP	Virtual IP
VM	Virtual Machine
VoIP	Voice over IP
WAN	Wide Area Network
WSFC	Microsoft Windows Server Failover Clustering
WS	Web Services
XML	eXtendable Markup Language

Chapter 1

Introduction

Nowadays the use of electronic services and Internet communications are increasingly common among citizens and thus the demand for better services and better solutions is constantly growing. Recent years have seen the emergence of new infrastructures and computing platforms as well as the improvement of the existing ones. The need to improve services and electronic communications is compelling and it requires constant monitoring and studying new solutions towards new infrastructures and platforms.

Computing platforms such as clusters and more recently cloud computing bring new paradigms of service delivery, and also the possibility to solve many current problems such as scalability, increased storage and processing capacity, greater rationalization of resources, cost reduction, and increase in performance.

One of the services critical to the operation of companies, institutions as well as to the communication between individuals is the e-mail. It is present in several devices, from mainfraim computers to mobile phones, using several protocols and architectures. It is, by far, the most ubiquitous service in the Internet. For this reason we chose to analyse the behavior of this service in several environments, to evaluate the performance of e-mail servers.

1.1 Objectives

The aim of this dissertation is to study architectures and computing platforms hosting e-mail services, in relation to the applications that implement this services, architectural details, vendors, platforms and infrastructure and advantages and disadvantages that comes with their adoption. More attention is given to open-source computing plataforms, due to the its' wide availability and flexibility.

The main contribution aims to test and check which platforms and e-mail services applications, or combinations, are more suitable, thus performing better.

1.2 Structure of this Document

This dissertation is composed by five chapters, including the present chapter:

- Chapter 1: introdution and objectives.
- Chapter 2: makes a presentation on the state of the art of Cluster, Grid and Cloud Computing, as well as E-mail server applications.
- Chapter 3: presents a case study focusing on the details needed to run an E-mail server in in a cluster and cloud environments and comparing them with single computer environment. It also defines test scenarios and tests to be carried out.
- Chapter 4: presents and discusses the results.

- Chapter 5: presents the conclusions of work and future evolution.

Chapter 2

Performance Optimization

The dependency on electronic communication services make it increasingly necessary to keep a high rate of throughput as well as high availability and fault tolerance. Computer systems have evolved with this requirements in mind and today there are systems that are suitable to cope with the modern day demands. Clusters, as well as cloud computing, provide capability on demand, making it easy to provide resources beyond the hardware limits.

2.1 Cluster Computing

A cluster computer is a set of independent computers connected by a high performance network [Baker, 2000]. All the subsystems are supervised within a single administrative domain, usually residing in a single room and managed as a single computer system. This subsystem components are called nodes and can incorporate a single microprocessor as well as multiple microprocessors in a symmetric multiprocessor configuration. The nodes are commonly connected to each other through fast Local Area Networks (LANs) or System Area Networks (SANs) technologies that may be structured hierarchically. The cluster network should be dedicated, thus separated from the external world. Clusters are usually deployed to improve performance and/or availability in relation to a single computer, while typically being much more cost-effective than single computers of comparable speed or availability [Bader and Pennington, 2001]. The nodes which compose the cluster can be homogeneous or heterogeneous, an important factor for the analysis of cluster performance [Naiouf et al., 2006]. Homogeneous nodes are those that have the same architecture, the same OS (including the same OS version) and the same key component libraries, such as libc or glibc on Linux and freeware BSD OSs. Heterogeneous nodes are those that can not be defined as homogeneous [lam, 2006]. Figure 3.2 shows an example of a cluster with N nodes.

A cluster can be used in many modes: high capacity and sustained performance, high capacity and throughput, high availability through redundancy, or high bandwidth through a multiplicity of disks and disk access channels or I/O. According to its role, clusters are classified in: High-Availability (HA), Load-Balacing and Computing Clusters. It is also possible to have Hybrid Clusters, combinations of the above.

2.1.1 High-Availability (HA) Clusters

HA clusters, also knows as Failover Clusters, are implemented primarily for the purpose of improving the availability of services. To ensure high-availability of services, the cluster requires several mechanisms and services to operate properly, such as a monitor mechanism, a backup mechanism, a recover mechanism and a failover mechanism [Li-Gu et al., 2006]. The mode of operation is to have redundant nodes which are used when some primary node fail. In pratice, if a server with a particular application crashes, the application will be unavailable until someone fixes the crashed server. With HA this situation is remedied by detecting hardware/software faults, and immediately restarting the application on another system

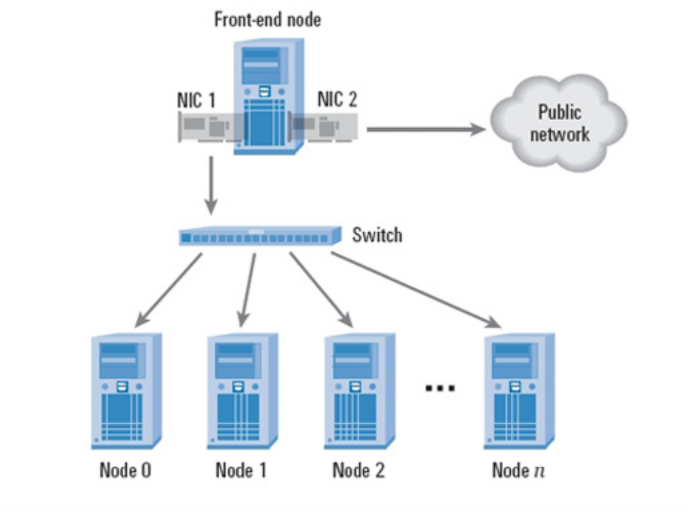

Figure 2.1: Cluster Layout

without administrative intervention, a process know as *Failover*. As part of this process, the cluster software can configure the node before starting the application on it. For example, file systems may have to be imported and mounted, network hardware may have to be configured, and some applications customizations maybe needed. The HA clusters are typically used for critical databases, file sharing, networking, business applications and customer services such as e-commerce sites. Applications of HA cluster try to build redundancy to eliminate single points of failure, using multiple network connections and data storage which is multiply connected via SAN [Yves and Patrice, 2008]. To monitor the health and status of each node, the HA clusters usually use a heartbeat private network. A special condition that the software devices cluster must have is being able to deal with the split-brain. This occurs when all of the private links go down simultaneously, but the cluster nodes are still running. When this happens, each node in the cluster may mistakenly decide that every other node has gone down and attempt to start services that other nodes are still running, resulting in duplicate instances of service may cause data corruption on shared storage. Access to services provided by the cluster is done through a network address that is called the logical host or cluster logical host. This logical host identity is not tied to a single cluster node. It is actually a network address/hostname that is linked with the service(s) provided by the cluster. If a cluster node that is running a service goes down, this service will be restarted on another node in the cluster and it will be associated with the network address/hostname of service, the logical host.

Two nodes is the minimum size for this type of clusters, as it is the minimum requirement to provide redundancy [Chee-Wei and Chen-Khong, 2007]. However, tipically, many clusters are composed of many nodes, sometimes dozens or hundreths of nodes. These configurations can be classified in one of the following models: [Robertson, 2000]:

- **Active/Active** - Traffic destined to the failed node is passed to an existing node or load is balanced by the other nodes. This is usually only possible when the nodes utilize a homogeneous software configuration.
- **Active/Passive** - Provides a fully redundant instance for each node that is only brought online when the primary node fails. This configuration typically requires an extra amount of hardware that is not used until needed.

Not all applications can run in a clustered environment for high availability, and the design decisions need to be made in the early design phase of software. The applications must satisfy at least the following technical requirements in order to run in a HA cluster environment:

- Should be a relatively easy way to start, stop, force-stop, and check the status of the application. In practice, the application must have a line interface commands or scripts to monitor the application, and must support multiple instances.
- The application must support the use of shared storage (NAS/SAN).
- The application should store as much data as possible in a non-volatile shared storage. It is also important that it can restart another node using the last saved state.
- The application should not corrupt the data when crash or restart from the last saved state.

HA cluster typically use all techniques to make the individual systems and shared infrastructure as reliable as possible. These include:

- Disk mirroring avoiding system failures due to failures of internal disks.
- Redundant network connections, this means redundancy of all active and passive components of network.
- Redundant Storage Area Network data connections.
- Redundant electrical power inputs on different circuits, usually both or all protected by Uninterruptible Power Supply (UPS) units.

These techniques help to minimize the need for clustering failover between systems. The HA cluster products commonly used comercialy or research/academic are:

- **HP ServiceGuard** for HP-UX and Linux
- **IBM High Availability Cluster Multiprocessing (HACMP)** for AIX and Linux - recently rebranded PowerHA for AIX
- **Linux-HA** - a commonly used free software HA package for the Linux OS.
- **Microsoft Cluster Server (MSCS)** and **Microsoft Windows Server Failover Clustering (WSFC)** for Server 2008.
- **Oracle Clusterware** - multi-platform, provides the infrastructure for Oracle Real Application Clusters (RAC) and enables the protection of both Oracle and non-Oracle applications.
- **Red Hat Cluster Suite** - Linux only.
- **Sun Cluster** - Solaris/OpenSolaris only.
- **Veritas Cluster Server** - multi-platform.
- **VMware HA** - HA for virtual machines.

2.1.2 Load-Balancing Clusters

Load-Balancing it is a technique used to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get a optimal resource utilization, maximize throughput, minimize response time, and avoid overload [Bourke, 2001]. The use of load-balancing stategies can improve the performance of parallel and distributed applications, by dividing the workload among the machines, to take proper advantage of the resources [Laine and Midorikawa, 2007]. We can use different approaches to this objective, both for homogeneous and heterogeneous environments. Generally in homogeneous environments, such as homogeneous clusters, a static strategy is used to divide the work into a number of pieces equal to the number of nodes or processors in the environment, to evenly distribute the computing local nodes. In heterogeneous environments, such as heterogeneous clusters or computational grids, the workload is divided according to the computational power of each host, and then, all the nodes received a workload proportional to its processing capacity an load status. This static workload partition and distribution creates a low overhead compared with dynamic strategies [Laine and Midorikawa, 2007]. These strategies are normally based on the divide-to-conquer techniques.

One of the most common applications of load balancing is to provide a single Internet service from multiple servers, sometimes known as a server farm. Some examples are: popular Web Sites, large Internet Relay Chat (IRC) networks, high-bandwidth File Transfer Protocol (FTP) sites, Network News Transfer Protocol (NNTP) servers, Domain Name System (DNS) servers and Email servers. At Internet services, the load balancer is usually a software program that is listening on the port that clients use to access the service. The load balancer foward the request to one of cluster nodes. This allows the load balancer to reply to the client without the client ever knowing who really answered. It also prevents clients contacting nodes clusters directly, which results in security benefits by hiding the structure of internal network and may prevent attacks in case of vulnerabilities. To select the node, a load balancer must know which ones are available. For this, it will periodically perform a “health check”. In practice it will periodically send echo-replies, connection attemps, requests, or anything that the administrator considers a valid measure to qualify their state.

At Internet services the easiest way to perform load-balancing is to dedicate servers to predefined groups of users. This approach does not required a dedicated software or a extra hardware node and is called DNS round robin [Tarreau, 2006]. In this technique, a DNS server has a multiple Internet Protocol (IP) address for a given hostname, it will return all in a rotating order. This technique unlike dedicated load balancer, is not “transparent” to clients, because there will be different address for the same hostname. This is usually used by load balancers for multi-site, but requires that the application is not affected by the lack of server context. For this reason, this is generally used by search engines, POP servers, or to deliver static contents. This method requires additional checks the status of servers, so that in case of failure to proceed to change the IP to another server [Tarreau, 2006]. For this reason it is often used as a complementary solution and never as a first choice. Depending on the degree of control over the DNS server and the granularity of load balancing desired this technique may have other advantages or disadvantages.

The load balancers use a variety of scheduling algorithms to determinate which server node should send the request. The most common scheduling algorithms used in load-balancing clusters are [clu, 2009]:

- **Round-Robin** - Distributes each request sequentially around a pool of cluster nodes. Using this algorithm, all the cluster nodes are treated as equals without regard to capacity or load.
- **Weighted Round-Robin** - Distributes each request sequentially around a pool of cluster nodes proportionaly to its capacity. Capacity is indicated by a user-assigned weight factor, which is then adjusted up or down by dynamic load information. This is a preferred choice if there are significant differences in the capacity of cluster nodes.
- **Least-Connection** - Distributes more requests to cluster nodes with fewer active connections. This is a type of dynamic scheduling algorithm, making it a better choice if there is a high degree of variation in the request load.
- **Weighted Least-Connections** - Distributes more requests to cluster nodes with fewer active connections relative to their capacities. Capacity is indicated by a user-assigned weight, which is then adjusted up or down by dynamic load information. The addition of weighting makes this algorithm ideal when the cluster nodes pool contains hardware of varying capacity.
- **Locality-Based Least-Connection** - Distributes more requests to cluster nodes with fewer active connections relative to their destination IPs. This algorithm is for use in a proxy-cache server cluster. It routes the packets for an IP address to the server for that address unless that server is above its capacity and has a server in its half load, in which case it assigns the IP address to the least loaded cluster node.

- **Locality-Based Least-Connection with Replication Scheduling** - Distributes more requests to servers with fewer active connections relative to their destination IPs. This algorithm is also used in a proxy-cache server cluster. It differs from Locality-Based Least-Connection Scheduling by mapping the target IP address to a subset of cluster nodes. Requests are then routed to the server in this subset with the lowest number of connections. If all the nodes for the destination IP are above capacity, it replicates a new server for that destination IP address by adding the cluster node with the least connections from the overall pool of cluster nodes to the subset of cluster nodes for that destination IP. The most-loaded node is then dropped from the cluster node subset to prevent over-replication.
- **Source Hash Scheduling** - Distributes requests to the pool of cluster nodes by looking up the source IP in a static hash table. This algorithm is for load balancers routers with multiple firewalls.

Generally, a hardware load balancer [Tarreau, 2006] and some software load balancers [clu, 2009] will work at the network packets level and will act on routing, using one of the following methods:

- **Direct routing**: allows the cluster nodes to process and route packets directly to a requesting user rather than passing outgoing packets through the load balancer router. In this method the load balancer and the cluster nodes must be in same network segment and must share the same address for all services.
- **Tunnelling**: this mode of operation is identical to direct routing, except that tunnels are established between load balancer and cluster nodes.
- **Network Address Translation (NAT)**: the load balancer receive the request and routes it to the appropriate cluster node. The latter processes the request and returns the packets to the load balancer. The load balancer uses network translation to replace the address of the cluster node with the public IP address. The main disadvantage of this method is that the load balancer become a bottleneck in a large deployments because it must process outgoing and incoming requests.

In certain situations, it may be desirable for a client to reconnect repeatedly to the same cluster node (multi-screen web forms, cookies, Secure Sockets Layer (SSL), and FTP connections). In those situations a client may not work properly unless the transactions are being handled by the same cluster node to retain the context. The load balancing clustering provides features to help solve this problem, and the most common are Persistence and Firewall Marks [clu, 2009].

Most of the products that implement load balancing are hardware appliances and the most used are:

- Barracuda Load Balancer
- Array Networks APV Series
- A10 Networks AX Series
- Cisco LocalDirector 400 Series
- Brocade Global Sever Load Balancing
- F5 Networks
- Citrix NetScaler
- Kemp Technologies Load Master Series
- CoyotePoint Equalizer Family

There are also software implementations Load Balancing and the most used are:

- Zeus Load Balancer
- RedHat Cluster Suite
- LVS

2.1.3 High-Performance Computing (HPC) Clusters

High-Performance Computing (HPC) cluster is a set of thousands of independents nodes interconnected through specialized fast networks. Each node run independent operating system kernel, thus synchronization between nodes is demanded for user mode programs. This means that temporal synchronization of the nodes is a dificult task [Betti et al., 2009]. Moreover, the HPC cluster applications often require an accurate time synchronization in activities such as performance analysis, debugging code, or checkpoint data. Therefore, the performance of an HPC parallel application can be severely hampered by the absence of time synchronization between the activities of the cluster nodes, thus putting a limit on scalability of these architectures. The HPC Clusters are used by many research areas and fields that require a very high computational power such as theoretical physics, weather forecasting, cryptoanalysis. Historically, supercomputers were dedicated machines to solve specific kinds of computationally difficult problems, but only a few institutions were able to own one of them, because these machines were really expensive. Nowadays with the progress in computer industry and in computer science, it is possible to build a supercomputer by means of high-end computers grouped in a cluster [Betti et al., 2009]. This modern supercomputers are cheaper than old ones and generally have more computing power. In November 2009 the TOP500 Supercomputing sites list includes 410 HPC Clusters out of 500 total entries. Most of these supercomputers running a version of Linux kernel (446 entries out 500 int the TOP500 list of November 2009). Generally each node of cluster runs a Linux kernel that handles all cores in the node [top, 2009].

2.2 Grid Computing

Grid computing is a form of parallel and distributed computing based on an infrastructure that integrate and collaborates the use of heterogeneous collections of high end computers, storage systems, sensors, scientific instruments, databases, etc., all interconnected by a wide, geographically dispersed network [Kouvatsos and Mkwawa, 2003, Al-Khannak et al., 2007]. These resources are owned and managed by several organizations, usualy called administrative domains. A Grid is built from multi-purpose protocols and interfaces that address fundamental issues such as authentication, authorization, resource discovery, and resource access. Grid computing is oriented to high performance computing, but because of their geographical separation may be a delay in communications between the nodes. This uses its constituent resources in a coordinated manner to offer different qualities of service, like the reponse time, throughput, availability, security and/or co-allocation of multiple resources.

Grid computing allows monitoring and interacting with hosted applications. It is also possible to monitor and interact with power systems and network system using online distributed system. Medicine, natural sciences, computer science, economics, and energy management and other challenging problems are the research areas and fields where grid computing has been mainly used [Al-Khannak et al., 2007]. Examples of grand challenge problems that Grid computing offer a way to solve are protein folding, financial modeling, earthquake simulation, and climate/weather modeling. Web technologies and standards (e.g. eXtendable Markup Language (XML) or Web Services (WS) are the foundation for scalability and adaptability in grid computing.

2.2.1 Architecture

The main layers of Grid architecture are Fabric, Connectivity, Resource, Collective, Application (Figure 2.2) [Foster et al., 2001]. Grid middleware is the core system of grid computing and users rely on it to use the available resources. This core system encompasses the layers Connectivity, Resource and Collective.

The Fabric layer provides the resources to shared and these are mediated by Grid protocols, such as computational resources, storage systems, catalogs, network resources, scientific instruments and sensors. A "resource" can be a logical entity, such as a computer cluster, distributed file system, or a distributed computer pool. In some cases a resourece implementation can involve internal protocols (e.g. Network File System (NFS)), but these are not a concern of the Grid architecture. Fabric components implement

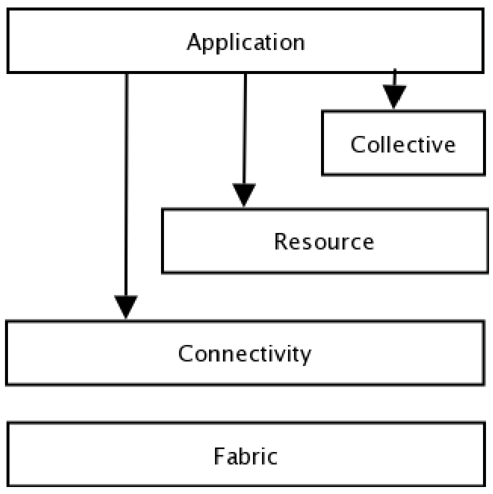

Figure 2.2: Grid Architecture

locally, resource-specific operations that occur on specific resources (physical or logical) as a result of sharing operations at higher levels. There is a close interdependence between the functions implemented at the Fabric level, and the sharing operations supported at higher levels. The more richer the fabric functionality, the more sophisticated are the sharing operations, and if we place few demands on Fabric elements, then deployment of Grid infrastructure is simplified. For example, resource-level support for advance reservations makes it possible for higher-level services to aggregate (coschedule) resources in interesting ways that would otherwise be impossible to achieve. According to some authors [Foster et al., 2001], at a minimum, resources should implement discovery mechanisms that allow describing its structure, status and capabilities (e.g. whether support advance reservation). Moreover, mechanisms for resource management should provide some quality control of services delivered. Some resource-specific examples are: computational resources, storage resources, network resources, code repositories and catalogs.

The Connectivity layer defines core communication and authentication protocols required for Grid-specific network transactions. Communication protocols enable the exchange of data between the resources of the fabric layer, and the authentication protocols provide secure mechanisms to communicate and to verify the identity of resources and users. Communication requirements include transport, routing, and naming. Authentication solutions should have the following features: Single Sign On (SSO), delegation, integration with various local security solutions (such as Kerberos and Unix security) and User-based trust relationships. Grid security solutions should also provide flexible support for communication protection (e.g. control over the degree of protection, independent data unit protection for unreliable protocols, support for reliable transport protocols other than Transmission Control Protocol (TCP)) and enable stakeholder control over authorization decisions, including the ability to restrict the delegation of rights in various ways [Foster et al., 2001].

The Resource layer builds on Connectivity layer to define protocols (and Application Programming Interfaces (APIs) and Software Development Kit (SDK)) for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. This protocols implementations call Fabric layer functions to access and control local resources. Resource layer protocols are only concerned with individual resources and thus ignore the issues of global state and atomic actions in distributed collections, which are handled by the collection layer. Two primary classes of Resource layer protocols can be distinguished [Foster et al., 2001]:

- Information protocols are used to obtain information about the structure and state of a resource, for example, its configuration, current load, and usage policy (e.g. cost).

- Management protocols are used to negotiate access to a shared resource, such as resource requirements (including advanced reservation and quality of service) and the operation(s) to be performed, such as process creation, or data access.

The main standards-based of protocols which were adopted by this layer are:

- A Grid Resource Information Protocol (GRIP) (currently based on the LDAP) is used to define a standard resource information protocol and associated information model.
- The HTTP-based Grid Resource Access and Management (GRAM) protocol is used for allocation of computational resources and for monitoring and control of computation on those resources.
- An extended version of the FTP, GridFTP, is a management protocol for data access. FTP is adopted as a base data transfer protocol because of its support for third-party transfers and because its separate control and data channels facilitate the implementation of sophisticated servers.
- LDAP is also used as a catalog access protocol.

Unlike the Resource layer, the Collective layer contains protocols, services, APIs and SDKs that are not associated with a specific resource but rather are global in nature and capture interactions across collections of resources. As Collective layer is built on the Resource layer can implement a variety of sharing behaviors without placing new demands on the resources shared. For example:

- Directory services;
- Co-allocation, scheduling, and brokering services;
- Monitoring and diagnostics services;
- Data replication services;
- Grid-enabled programming systems;
- Workload management systems and collaboration frameworks;
- Software discovery services;
- Community authorization servers;
- Community accounting and payment services;
- Collaboratory services.

The last layer is the Application layer that contains the user applications. Applications are built by calling the services defined at any layer. In each layer, we have well defined protocols that guarantee access to some useful service (resource management, data access, resource discovery, and so on). In each layer, APIs may also define which implementation (ideally provided by third-party SDKs) exchange messages with the appropriate service(s) to perform the desired actions.

2.2.2 Implementations

Midleware implementations can provide a set of service solutions, such as data management, resource management, information discovery and secure comunication infrastrutures, even using the Internet [Esposito and Tarricone, 2003, Al-Khannak et al., 2007]. There are several projects which implement grid midleware and the most used are Globus Toolkit, Berkeley Open Infrastructure for Network Computing (BOINC), Legion, Condor-G, Accessgrid, Alchemi, Sun Grid Engine, Gridbus, and gLite.

2.3 Cloud Computing

Cloud computing is an emerging model which refer to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services [Armbrust et al., 2009]. This services are referred as SaaS and the hardware and software of data center is called cloud. Some authors [Gens, 2008] divide this into two concepts: cloud computing and cloud services. Since then the following definitions:

- **Cloud Services** - Consumer and Business products, services and solutions that are delivered and consumed in real-time over the Internet
- **Cloud Computing** - an emerging Information Technology (IT) development, deployment and delivery model, enabling real-time delivery of products, services and solutions over the Internet (i.e., enabling cloud services)

2.3.1 Cloud Computing Models

Clouds are commonly classified as public and private. Public clouds are available for anyone, from an Internet access, which may access and start using almost immediately. On the other hand private clouds are usually located within a private infrastructure and managed by the users' organization [Maxey, 2008]. Each orgazination should independently analyze each situation and decide between models public, private or hybrid. This choice may be important to their users in terms of availability of service(s) application(s), contract types and prices. A number of factors should be taken into account:

- **Initial Investment** - The private cloud has an associated initial investment, which depends on the desired solution, but that is usually higher than the initial investment in a public cloud;
- **Amount of Data** - Typically, organizations start with small amounts of data and increasing demand for these will grow. Private clouds typically get a range of a few Terabytes (TBs) and are easily scalable by adding additional nodes or disks. On the other hand public cloud get a range of a few Gigabytes (GBs) and can grow lease more capacity and that the cost increases linearly.
- **Longevity of Data** - The amount of time that we plan to keep the data may have an impact on decision. Within a public cloud the longer storage is more expensive, so it is good for frequent publication of short duration or small content. Within private cloud the longevity of data does not increase the cost of solution, so it is good for archive or content repository applications;
- **Required Performance** - Private clouds are deployed locally and accessed over Ethernet LAN at wire speed, tipically at least 100 MB/s per node. Public clouds are accessed over Internet and limited by the your and provider bandwidth connection. Typically these approaches are around 10 MB/s but to increase the performance can be increased, also increasing the cost.
- **Access Paterns and Locations** - Public clouds offerings tipically include replication of data to multiple geographically dispersed locations, but sometimes it is a extra. This can be a beneffit when yours users are global.Private clouds are usually deployed locally for LAN-based access. Remote users can access using a Wan, but work with Internet type latencies. Major private cloud deployments can use multiple sites, thus coming closer to the approach of public cloud, but with a high initial investment;
- **Security and Data Isolation** - Public clouds as the name suggests, are shared by different contexts and their security capacity is only achieved by virtualization of resources and the firewall of the service provider. If the security issue is critical this option is not the right one. Private clouds are owned, deployed and managed by internal organization employees, so security and data isolation is based in your internal definitions for that;

- **Confidentiality and Destruction of Data -** In public clouds must be considered matters of confidentiality to the service provider of storage, as this may be required to keep, disclose or answer questions about the information they hold and call into question the confidentiality of the organization, in case of subpoena or legal questions. In private clouds, control and knowledge of the data and verifying the legality of these is done internally. When you need to delete data at the discretion of the organization;
- **Service Level Agreements (SLA) -** Public clouds have contracted SLAs with their customers with guaranteed availability of service. This alone does not guarantee the absence of faults, in the case of unavailability of the service may affect critical business activity of the organization. In addition it must take into account another possible point of failure the Internet provider. In private clouds are different mechanisms for data availability and access to, such as multiple copies of files on multiple nodes and treat each node as a failure domain. In case of individual server failures do not gring down the cloud or create data loss, so the SLA agreements are usually satisfied;
- **In House Technical Resources -** Public clouds remove the need for allocation of resources of the organization for the management and maintenance of infrastructure. But with the use of newer protocols such as WebDav or Rest for access, the applications will have to be adapted, and for doing will require knowledge and technical stuff.

These factors should be taken into account to choose the model that best fits the organization needs (public or private). However, the solution may also involve using a hybrid model.

Public Clouds

Public cloud, also know as external cloud, is referred to when its use is made in a pay-as-you-go manner [Armbrust et al., 2009, Razak, 2009]. The pay-as-you-go billing model applies charges for the actually used resources per unit of time. This way, a business can optimize its IT investment and improve availability and scalability.

Private Clouds

Private cloud is reffered when the cloud is not available to the public, like internal data centers of business or other private organizations [Razak, 2009]. The private clouds provide local users with a flexible and agile private infrastructure to run virtualized service workloads within the administrative domain. For organizations with existing data center investments, it allows reducing operating cost while increasing the agility and reability of their critical information services. In this scenario, internal and external cloud resources are consumed and controlled centrally by the organization itself.

Hybrid Clouds

A hybrid cloud is an extension of a private cloud to combine local resources with remote resources from cloud providers. The cloud providers could be a commercial cloud service or a partner infrastruture running a different instance of a private cloud.

2.3.2 Cloud Computing Architecture

Cloud computing is an emerging paradigm, yet with little maturity, and suffering from a number of uncertainties, related to approaches and contexts that are not always consistent. Currently there is no entity that benefits users and providers of cloud computing that has the responsibility to manage, to organize, and standardize the cloud computing universe. Entities most respected and involved in the development of solutions, which highlights the Amazon, Google, IBM, Sun, 3Tera, and some entities from the academic world, have been responsible for some specifications in this area. It is normal to find different approaches to the taxonomy of structure and architecture of the cloud computing.

Cloud computing is composed of three layers: Software as a Service (SaaS), Plataform as a Service (PaaS) and Infrastructure as a Service (IaaS) (Figure 2.3) [Keith, 2009].

Software as a Service
Platform as a Service
Infrastructure as a Service

Figure 2.3: Cloud Computing Taxonomy or Cloud Computing Stack [Keith, 2009]

Infrastructure as a Service (IaaS)

Iaas is the provision of computer infrastructure (typically a platform virtualization environment) as a service. IaaS clouds provide very easy and affordable access to resources, such as servers, connections, storage and related tools. However, it is needed to build an application environment from scratch on demand. This infrastructure is the underlying of PaaS and SaaS clouds. IaaS clouds are very complex because it is possible to obtain a great degree of flexibility. Servies are, generally, of low-level, which have to deal with virtual machines, operating systems, patches. For this reason, it may be necessary to have skilled labor, or resort to outsourcing. The IaaS can be divided into three components: computational resources, data storage and communication.

The most common form for providing computational resources at IaaS layer it is through VM [Youseff et al., 2008]. Virtualization is a leading technology in this component of the cloud, enabling users to with a secure environment and with unprecedented flexibility in configuring their settings, thereby protecting physical infrastructure of the center of the data provider. This was specifically enabled by two technologies of virtualization: paravirtualization and hardware-assisted virtualization. However the lack of strict isolation between the performance of virtual images which share the same physical infrastructure has resulted in the difficulty in providing assurances of quality of service. Furthermore, the emergence of multicore machines aggravated this problem. To overcome these problems and to safeguard the providers use a number of SLA levels that translate into different levels of service with competitive prices. Amazon Elastic Compute Cloud (EC2), and Enomalism elastic computing infrastructure are undoubtedly the two most popular examples of commercial systems available in this cloud category. In this category, there are also several academic open-source cloud projects, such as Eucalythus, which provides a computing infrastructure for deploying private or hybrid clouds, with support for integration with Amazon EC2. This is a tool often used to test Cloud Computing solutions in controlled environments.

Data Storage allows users to store their data on remote virtual disks, and once it is stored this data can be accessed from anywhere. This service is offen called Data Storage as a Service (DaaS), and it facilitates cloud applications to scale beyond their limited servers. In cloud environments it is required that Data Storage Systems comply with requisites such as: high availability, reliability, performance, data replication and consistency, but we ought to contradictory nature of these requisites will not be expected that the system can meet all at once. For example, availability, scalability and data consistency can be considered as three conflicting objectives. Although these features are difficult to be satisfied with the systems of data storage in general, each DaaS provider defines which it implements to the detriment of others, reflecting that choice in their SLA. Some examples of commercial DaaS-systems are Amazon's Simple Storage Service (S3) and EMC Storage Managed Service [Youseff et al., 2008].

In the cloud the need to guarantee quality of service increases, and consequently the network component and communications becomes vital to the infrastructure. So, cloud systems are required to provide capacity-oriented communication services, configurable, scalable, predictable and reliable. Given these needs, the concept of Communication as a Service (CaaS) was created, to support such requirements as well as network security, dynamic provisioning of virtual overlays for traffic isolation or dedicated bandwidth, guaranteed message delay, communication encryption, and network monitoring. Voice over IP (VoIP) phone systems, collaboration videoconferencing applications using fixed and mobile devices, and instant messaging applications are candidates for cloud applications that can be composed of CaaS and can in turn provide combinable cloud solutions for other common applications [Youseff et al., 2008].

Plataform as a Service (PaaS)

The users of this layer are cloud applications developers, implementing their applications for and deploying them on the cloud [Youseff et al., 2008]. PaaS provide developers with a programming environment with high-level language and a well-defined set of APIs to facilitate interaction between environments and cloud applications, and to accelerate the deployment and scalability needed to support these applications cloud.

Two common examples of a system in this category are the Google App Engine and the Salesforce Apex language. The Google App Engine provides a pynthon runtime environment and APIs for applications to interact with Google's cloud platform. Salesforce Apex language allows the design, along with logic applications, page layout, work flow and customer reports.

Developers have the benefits of automatic scalability, load balancing, and integration with other services (e.g. authentication services, email services, user interface) provided through the PaaS. This way much of the overhead of developing cloud applications is simplified, accelerating the time of issuance or amendment of new solutions and minimizes the problems affecting the business logic of applications. The use of these proprietary platforms can bring some disadvantages in interoperability and application migration between them.

Software as a Service (SaaS)

This layer is the most visible to the end-users off cloud, and are know by Cloud Application Layer or Software as a Service (SaaS). Typically, user access to services provided by this layer is done through web-portals, and sometimes these services are paid. Recently this model has proved attractive to many users, because it reduced the burden of software maintenance and support, and export the computational work of user terminals for data centers where the cloud applications are developed. This last feature allows the end user to obtain better performance to some of their cpu-intensive and memory-intensive workloads without the need for massive capital investments in local machines [Youseff et al., 2008].

For SaaS providers, this model allows them to simplify the work of testing and updating the code, thus protecting their intellectual property. Once the applications are developed under the provider computing infrastructure, the developers of the application are able to roll smaller patches to the system and add new features without disturbing the users with requests to install major updates or service packs.

Despite all the advantages of SaaS, several deployment issues hinder its wide adoption. The major issues in this model are the security, the availability of the cloud applications and the QoS, but that are usually avoided by using soft SLAs. Apart from these there are other issues with which the end users and SaaS providers have to deal with, such as service interruptions due to breaks in the network and other possible faults in the systems, and integration/migration of applications and end-user data to the cloud. These issues can cause delays in the adoption of SaaS. In order to persuade end users to migrate from desktop applications to applications cloud, cloud application vendors need to respond to the concerns of end users on security, QoS, security, storage of confidential data on the cloud, authentication and authorization users, availability, backup and data recovery in case of need and an SLA of confidence for their applications in the cloud [Youseff et al., 2008].

The SaaS provider, in addition to being responsible for the provision of applications to end users, is also responsible for providing computing power, storage, and networking infrastructure necessary to run the application [Keith, 2009]. It can use a third party IaaS or PaaS vendor or implement the application themselfs. Figure 2.4 shows the different environments that can be used to host a SaaS Application.

The most common examples of an application in this category are Customer Relationships Management (CRM) of Salesforce, Google Apps, Microsoft Online Services and LotusLive of IBM.

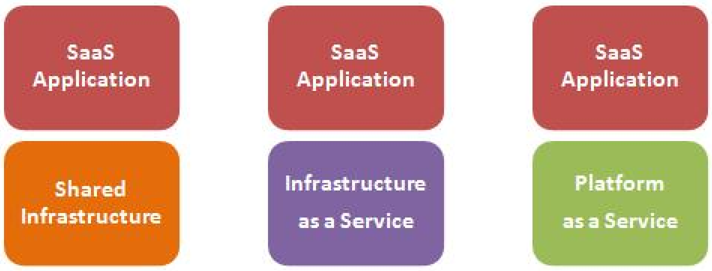

Figure 2.4: Three ways to host a SaaS Application [Keith, 2009]

2.3.3 Main Suppliers and Services

The early cloud computing suppliers were companies that discovered that they could get an extra return by making available its infrastructure to third parties, thereby optimizing the use and sharing costs. Google and Amazon are two examples because due to the volatility of their services had to scale its infrastructure, and it represented a very high investment. Then there was a second wave which was led by housing companies, because they also have large data centers and experience in Information Technology (IT) management. These were the pioneers who traced the course of Cloud Computing, followed by many others. Nowadays more and more companies have an opportunity for providing services in this market area. Table 2.1 shows a comparison between the main suppliers/solutions of Cloud Computing focusing on the main aspects that distinguish them.

Table 2.1: Comparison between suppliers/solutions of Cloud Computing

Supplier/ Solution	**Focus**	**Business Model**	**Infrastructure**	**Infrastructure Control**	**SLA**	**Notes**
Amazon EC2 and S3	IaaS	PAYG	Own	Web, REST API	yes	A pioneer and market reference. Besides EC2 and S3 offers a set of Cloud Computing services that makeAWS the more complete IaaS solution.
GoGrid	IaaS	PAYG or month/annual plans	Own	Web, REST API	yes	Is a Cloud Infrastructure Hosting supplier with extensive expertise and experience running complex, on-demand cloud, dedicated and mixed server infrastructures. Public Cloud.

Continued on next page

Table 2.1 – continued from previous page

Supplier/ Solution	**Focus**	**Business Model**	**Infrastructure**	**Infrastructure Control**	**SLA**	**Notes**
RackSpace	IaaS	PAYG or month plans	Own	Web, REST API	yes	Like GoGrid is a Cloud Infrastructure Hosting supplier a hybrid hosting that is to combine cloud computing with traditional hosting. Public Cloud
FlexiScale	IaaS	PAYG	Own	Web, REST API	yes	European supplier. Allows the creation of virtual datacenter. Public Cloud
Joyent	IaaS, PaaS, SaaS	Month/Annual plans	Own	Web	no	Joyent implements IaaS, PaaS and SaaS. Part of his platform is open-source, and can be used to implement private clouds.
3Tera AppLogic	Implements IaaS	Depends on the partners business model	Own, partners to use the OS AppLogic	Web, managment console	yes	AppLogic is a turn-key cloud computing platform for running and scaling distributed applications.
Layered Technologies	IaaS	Month plans	Own	Web, scripting console	yes	Using 3Tera AppLogic OS to implment grid on cloud. Allows the creation of virtual datacenter.
Eucalyptus	Implements IaaS	Free	Independent	Web, REST API compatible with EC2 and S3	yes	Is an open-source software platform that implements IaaS-style cloud computing using the existing Linux-based infrastructure found in the modern data center. Also works with most of the currently available Linux distributions including Ubuntu, Red Hat Enterprise Linux, CentOS, SUSE Linux Enterprise Server, openSUSE, Debian and Fedora [euc, 2010].
Google App Engine	PaaS	Free	Own	not available	no	Google App Engine (GAE) its a market reference for PaaS. GAE has an SDK available for Java and Python.
Google App Engine for Business	PaaS	Per user or per month	Own	Administration console	yes	Version for business of GAE.
Salesforce	PaaS, SaaS	Month plans	Leased - Equinix	not available	yes	It provides a platform for developing and integrating itself, the Force.com, and is assumed as a reference company in providing Saas.

Continued on next page

Table 2.1 – continued from previous page

Supplier/ Solution	Focus	Business Model	Infrastructure	Infrastructure Control	SLA	Notes
IBM LotusLive	SaaS	Depends on the partners business model	-	not avaiable	yes	LotusLive ss a suite of business networking and collaboration cloud-based services hosted by the Lotus Software. The integrated services include social networking for businesses, online meetings, file sharing, instant message, data visualization and e-mail.
Microsoft Azure	SaaS	Pay as you Go (PAYG) or Depends on the partners business model.	Partners with Microsoft Datacenters	.Net Services, REST API	yes	The Windows Azure platform is a set of cloud computing services that can be used together or independently. The main component is Windows Azure what is a a cloud services operating system that serves as the development, service hosting and service management environment [azu, 2010].

Each of these suppliers/solutions have a set of characteristics that distinguishes them. In the case of IaaS suppliers there is a big difference in market share among the top three (Amazon, GoGrid and RackSpace). The Amazon is a leader with a large difference, offering flexibility and potential to control the infrastructure that leads her to take as a reference in this field. In the past did not offer SLAs, which was a feature often essential in choosing a supplier. Currently SLAs with guaranteed 99.95% availability annual for the EC2 and 99.9% monthly for S3 [aws, 2010]. The GoGrid and RackSpace because of his background have an extensive expertise and experience running complex, on-demand cloud, dedicated and mixed server infrastructures [gog, 2010, rac, 2010]. Both guarantee strong SLAs to the availability of their services and have a great flexibility to build solutions with the client for Cloud Computing hybrid at the infrastructure level. The flexibility with which Amazon Machine Images (AMIs) are defined goes far beyond the equivalent of its competitors. As for the management of IPs and DNS and GoGrid RackSpace are leaders in relation to the Amazon Web Services (AWS).

Google positions itself in the market either in the category of SaaS, with Office applications like Google Docs and Gmail, either as a supplier of PaaS with Google App Engine (GAE). Its use is free until a certain quota of resource utilization. GAE platform is a somewhat limited because it only supports applications in Java and Python. Beyond this limitation is added the fact that applications are either designed specifically for this purpose (GAE offers an SDK that facilitates the process of development and integration tests), or will have to be processed in order to run on the this computing platform. Google most recently launched the GAE for business, that offers centralized administration, SLAs, more security and some other enterprise features like hosted SQL databases, SSL clients company's domain for secure communications, and access to advanced Google services [gae, 2010]. Compared to Google, Joyent with the SmartPlatform only supports Java applications. The Joyent SmartPlatform is recent, it is open-source and is likely to have much potential.

Microsoft Windows Azure is an OS for cloud computing and aims at the transition between operating systems to operate in data centers, such as Windows Server and other Microsoft solutions like SQL Server and .Net, to an environment compatible with the paradigm of cloud computing. Because of that Azure will be a market reference for all who use developed applications that use Microsoft technology, as this will ensure their smooth integration. It can be used for construction of public or private clouds and the guarantee of interoperability between different service providers that adopt the Azure.

Today there are several software solutions that implement IaaS and that convert traditional harware in an infrastructure for providinf IaaS. The comercial reference is the 3Tera AppLogic and the open-source reference is the Eucalyptus. The strength of AppLogic is the possibility of converting a set of servers in a distributed system of grid computing in which control of the infrastructure is done centrally. The Eucalyptus provides an abstraction layer of hardware resources by converting them into virtual resources. In being able to define private cloud or hybrid due to its compatibility with the API of AWS causes this to be adopted for test environments or academic. The AppLogic and Eucalyptus run on Linux operating systems and both allow the installation of virtual images that span different operating systems and software customized by the programmer. Azure lacks this flexibility.

2.4 E-Mail Service

The implemention of an e-mail service requires taking into account several factors, such as the format for storing users accounts (mbox or maildir format), the filesystem the mail will be stored in, the computing platform, and others.

2.4.1 Mail Tranfer Agent (MTA)

In the context of Internet Message Handling Services (MHS), a Message Tranfer Agent or Mail Tranfer Agent (MTA) or Mail Relay is a computer process or software agent that transfers electronic mail messages from one computer to another, in single hop application-level transactions. A MTA implements both the client (sending) and server (receiving) portions of the SMTP. In the context of the DNS, the term Mail Exchanger (MX) refers to an IP address assigned to a device hosting a mail server, and by extension also indicates the server itself. The SMTP first definition is in Request for Comments (RFC) 821 Internet Standard (STD) 15 in 1982 and the last updated by RFC 5321 in 2008, which includes some optional extensions. The SMTP is the most used protocol today and its objective is to transfer mail reliably and efficiently. This is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel [Klensin, 2008]. The RFC 5321 specifically describes transport over TCP, but other transports are possible (some of them are described in the appendices of the RFC 821. SMTP mail relaying is usually referred to as an important feature of SMTP, which consists of its capability to transport mail across multiple networks. Those networks, consisting of mutually-TCP-accessible hosts on the public Internet or mutually-TCP-accessible hosts on a firewall-isolated TCP/IP Intranet, or hosts in some other LAN or Wide Area Network (WAN) environment utilizing a non-TCP transport-level protocol.

Regarding the mode of operation, the MTA works behind the scenes, while the user usually interacts directly with the MUA. Mail Submission Agent (MSA) is the process which accepts message submissions from Mail User Agents (MUAs), and the TCP port 587 is used for that communication [Klensin and Gellens, 1998]. The communication between MTAs, or from an MSA to an MTA using the TCP port 25.

The process of sending an email to the receipt is as follows: it is sent by an e-mail client (Mail User Agent (MUA)) to a mail server (Mail Submission Agent (MSA)) using SMTP on TCP port 587 (most email providers also allow sending the standard port 25). Then the MSA forwards the email to his MTA. In most cases, these two agents are just different instances of the same software released with several options on the same machine. Local processing can be done on a single machine, or divided among several units in the first case, the processes involved can share files, in last case the SMTP is used to transfer messages internally, with each host configured to use the next unit as a smart host. Each process is an MTA, that is an SMTP server. The edge MTAs to find the target host uses the DNS to find the MX record to the recipient's domain (the part of the address on the right of @). The returned MX record contains the name of the destination host. The MTA then look for the **A** record to the name in order to obtain the IP address, then connects to the server as an SMTP client. We may have several MX records for a single domain, which allows administrators to specify an order in which they should be tried. Once the target MX accepts the incoming message, it passes it to a Mail Delivery Agent (MDA) for local mail delivery. The MDA is able to save messages in the format supported (mbox or maildir). As in the sending, receiving mail can be done using several computers or just two different instances of the same software released with several options on the same machine. Ended delivered to the mail server location, the mail is stored so it can be retrived by authenticated MUA. Mail is retrieved by the end user applications, called e-mail clients using the Internet Message Access Protocol (IMAP), or Post Office Protocol (POP), or proprietary systems (Exchange, Lotus Domino/Notes), or Webmail clients, which may use any method, or a non standard protocol.

The IMAP is an application layer internet protocol that allows an e-mail client to e-mail retrieval from a remote mail server, and is one of the two most prevalent Internet standard protocols for this function. The last version of this protocol are IMAP4rev1, and permits manipulation of mailboxes (remote message folders) in a way that is functionally equivalent to local folders [Crispin, 2003]. IMAP4rev1 also supports both on-line and off-line modes of operation, and provides the capability for an offline client to resynchronize with the server. Usually e-mail clients that use IMAP leaves the messages on the server until the user explicitly deletes them. This characteristic of IMAP operation allow multiple clients to manage the same mailbox. IMAP is used by some Webmail clients, such as Gmail, Atmail, Roundcube. IMAP server listens on well-known port 143. Encrypted IMAP traffic is either by IMAPS, which connects to the server using Transport Layer Security (TLS) or SSL on well-known TCP port 993, or requested after protocol initiation, using the STARTLS command, if supported [Newman, 1999].

The POP is an application layer internet protocol that allows an e-mail client to e-mail retrieval from a remote mail server, and is one too of the two most prevalent Internet standard protocols for this function. The last version and current standard of this protocol are POP version 3 (POP3). POP supports simple download-and-delete requirements for access to remote mailboxes, although most POP clients have an option to leave mail on server after download. POP is used too by some Webmail clients, such as Gmail, Yahoo, POP server listens on well-known port 110. Encrypted POP traffic is either by POPS, which connects to the server using TLS or SSL on well-known TCP port 995, or requested after protocol initiation, using the STLS command, if supported [Newman, 1999].

2.4.2 E-mail Service Implementations

There are many implementations of email service, some free and open-source, and other proprietary or open-source that must be paid. Someone can run only in Windows family, and other in Unix/Linux world. Some services implement SMTP, POP and IMAP, and others only implement SMTP, or SMTP and POP, or POP and IMAP.

The most popular proprietary and paid e-mail services implementations are Microsoft Exchange Server, IBM Lotus Domino and GroupWise of Novel. There are some open-source implementations, although requiring some form of paiment, and the most popular are Zimbra, Scalix and Atmail. The Zimbra has a open-source version that is free, but has fewer features. The Zimbra Collaboration Suite (now part of Yahoo!), is an collaboration suite, email and calendar server. This version it is Software as a Service (SaaS) ready [zim, 2010]. The Scalix Collaboration Platform is owned by Linux distributor Xandros and is an collaboration platform and an e-mail server implementation SaaS ready too [sca, 2010].

The most popular open-source and free SMTP implementations are: Postfix, Qmail, Exim, and Sendmail. The Qmail implements POP server too. The most polular open-source and free implementations of POP and IMAP are: Dovecot, Courier Mail Server and Cyrus IMAP.

Sendmail

Sendmail was written by Eric Allman as a derivative of delivermail early in the year 1980 at University of California, Berkeley. It was shipped with Berkeley Software Distribution (BSD) 4.1c in 1983, the first BSD version which included TCP/IP protocol. Allman projected Sendmail to have great flexibility, but this may be an enemy for beginners. The current version of Sendmail is version 8 and consists of about

118k lines of code, plus the M4 scripts used to generate the config file and milters. With M4 much of the configuration complexity is hiden. The Sendmail milter is an API to enable third-party programs to access mail messages as they are being processed by the MTA. This allows them to examine and modify message content and meta-information during the SMTP transaction [mil, 2010]. The goal of the project in the latest sendmail version is to be compatible with its predecessors. The sendmail has a monolithic main configuration file. The releases are regular, have many commiters and major contribs, and have a good documentation. Sendmail is not officially supported to interact with MySQL or LDAP, but only to use the ldap for aliases database.

Sendmail is known for its lack of security and many bugs. Perhaps because this originates in the early days of the Internet, an era when considerations of security did not play a primary role in the development of network software. Since about 2000 it has improved in security and performance, and has a large number of new features (milters). It is a default MTA on most commercial Unix versions and some Linux distributions. It has been ported to many systems, including Windows and is probably the most well-known MTA.

The official site for the open-source community of sendmail is http://www.sendmail.org/. The site is sponsored by Sendmail, Inc., which provides appliance-based products, business applications and SLA support and services for the open-source Sendmail MTA [sen, 2010].

Qmail

Qmail was writen, at the end of 1995, by Daniel Julius Bernstein as a more secure replacement for the popular Sendmail program. The last version of Qmail is version 1.03 and was released on June 15, 1998 [qma, 2010]. Its author has not released since then, and does not permit others to make releases. In November 2007 Qmail was released to the public domain, but is usable within very tight restrictions.

The main features of Qmail are security, performance, simplicity, modularity and some good inovations. At the first release, qmail was the first security-aware mail transport agent, prior to others who have appeared. Contrasting with Sendmail, Qmail has a modular architecture composed of mutually untrusting components; for instance, the SMTP listener component of Qmail runs with different credentials than the queue manager, or the SMTP sender. qmail was also implemented with a security-aware replacement to the C standard library, and as a result has not been vulnerable to stack and heap overflows, format string attacks, or temporary file race conditions [qma, 2010]. Qmail's performance was also superior to sendmail, particularly for bulk mail tasks such as mailing list servers. The configuration is easy and is distributed by many simple files. Because it is modular with respect to its main functions is possible to replace any part of Qmail system with a different module as long as it maintains the same interface. When Bernstein created Qmail one of his concerns was to solve the problems of the BSD mailbox format, and thus created the maildir message format. Other innovations were the user-controlled wildcards, the Quick Mail Transport Protocol (QMTP) and Quick Mail Queuing Protocol (QMQP) protocols and POP3 protocol support. With the user-controlled wildcards, out of the box, mail addressed to "user-wildcard" on qmail hosts is delivered to separate mailboxes, allowing users to publish multiple mail addresses for mailing lists and spam management. Other features are supports host and user masquerading, full host hiding, virtual domains, null clients, list-owner rewriting, relay control, double-bounce recording, arbitrary RFC 822 address lists, cross-host mailing list loop detection, per-recipient checkpointing, downed host backoffs, independent message retry schedules, and qmail also includes a drop-in "Sendmail" wrapper. Today those benefits have already been matched and exceeded by some MTAs and and Qmail is not so good option, because it is not maintained and is impossible someone else maintain it, do not suport IPv6 protocol and the modern mail standards.

To resolve some of these problems and fill other needs,some communities of users of Qmail, have developed some patches. With the amount of patches that have appeared, without much scrutiny, it became necessary to know how to choose them and how to conjugate them. Some group of Qmail experts created distributions of sets of patches, and the most popular are: netqmail (http://qmail.org/netqmail/), qmailrocks (http://qmailrocks.org/) and qmail-ldap (http://www.qmail-ldap.org/). Due to the patches Qmail supports interaction with some databases such as Mysql, PostgreSQL and Openldap, the latter being supported by the community qmail-ldap. In addition to the patches there are some applications that is usually advisable to its installation along with Qmail, such as ezmlm-idx, ucspi-tcp, daemontools, checkpassord. The ezmlm-idx is an easy-to-use, high-speed mailing list manager, the ucspi-tcp is an application created by Bernstein and implement tcpserver and tcpclient command-line tools, the daemontools, also created by Bernstein, is a collection of tools for managing UNIX services, and the checkpassword, also created by Bernstein, provides a simple, uniform password-checking interface to all root applications, such as pop3d. In last version of qmail-ldap patch the checkpassword is not necessary because the authentication function of the pop3d is already implemented. There is much documentation for Qmail and the use of their patches and the best website to start is http://www.qmail.org/, or netqmail, or qmailrocks or qmail-ldap websites.

The website of Qmail author is http://cr.yp.to/qmail.html, and the website where it is the Qmail source code, documentation and all the patches is http://www.qmail.org/.

Postfix

As Qmail, Postfix was written by a prolific Unix security software author, Wietse Zweitze Venema. The first release was in 1997 and this project had as the objectives to be fast, efficient, easy to administer, and secure. Postfix is a work largely of Wietse, with occasional contributions from isolated areas such as integration of the TLS libraries. The management of the release is made by Wietse, and these come in bursts, with some releases containing only very small improvements. The last stable version of Postfix is 2.7.1. The Postfix is under the IBM Public License (IPL).

Postfix fits being Qmail and Exim, consists of several programs (less than qmail), has a monolithic configuration file, like Exim and Sendmail, handles regular expressions in many contexts, using the Perl Compatible Regular Expressions (PCRE) developed by Phil Hazel for Exim, and it is table-driven, meaning that everything is a table and that table can be represented in all kinds of ways, from simple text files to relational databases. It supports databases such as SQLite, PostgresSQL, MySQL,LDAP, DBM, CDB and Berkeley DB. Other popular features of Postfix are IPv6 support, SASL authentication, TLS encryption and authentication, QMQP server, Multipurpose Internet Mail Extensions (MIME) (including 8BITMIME to 7BIT conversion), DomainKeys Identified Mail (DKIM), DomainKeys and SenderID authentication (via Milter plug-in), junk mail control, address manipulation, mailbox support (mbox and maildir) [pos, 2010]. Postfix is a drop-in replacement for Sendmail (it implements the Sendmail command line interface and is compatible with Sendmail milters).

Postfix has a tiny development community (one commiter and three major contribs), medium-size, but very active, user community, good documentation, and your website is http://www.postfix.org/.

Exim

The Exim is an MTA used on Unix-like operating systems, and its first version was written in 1995 by Philip Hazel for use in the University of Cambridge Computing Services e-mail systems. The name initially stood for EXperimental Internet Mailer. It was originally based on an older MTA, Smail-3, but it has since diverged from Smail-3 in its design and philosophy [Golanski, 2000]. The objective of this project is to be a general-purpose MTA for Unix machines. The latest version of Exim is 4.72, and is distributed under the terms of the GNU General Public Licence. Exim has been ported to most Unix-like systems, as well as to Microsoft Windows using the Cygwin emulation layer. Exim 4 is currently the default MTA on Debian GNU/Linux systems.

With respect to security Exim3 had some flaws and, consequently, a pair of serious security flaws, solved with the release of 4.

Exim, like Smail and Sendmail, has a single binary that controls all the facilities of the MTA, a monolithic configuration file (at version 4.x, the configuration file can be split into several files). Exim is highly configurable, with features that are lacking in other MTAs. It had always had substantial facilities for mail policy controls, providing facilities for the administrator to control who may send or relay mail through the system. Version 4.x included many important control features such as Access Control List (ACL) based system and the integration of a framework for content scanning. With ACL based system allows very detailed and flexible controls and with framework for content scanning allows for easier integration of anti-virus and anti-spam measures [exi, 2010]. There are two main schools of configuration style for Exim. The first and native school, strongly influenced by Philip Hazel, keeps the Exim configuration in one file and external files are only used as data sources. The second commonly style is the Debian style which is designed to make it easier to have an installed application automatically provide mail integration support without having the administrator edit configuration files. Due to differences of approach of the two, is normal to have flat support for the Debian approach in the Exim mailing lists, but there is Debian-specific list.

The Exim documentation is extensive and exhaustive, and when a feature or some behaviour is not documented then this is classed as a bug. There are 3 main sets of documentation for exim, all of which are also available online in HTML, pdf and text-file [exi, 2010]. The Exim Specification which is the master documentation for exim containing all required detail to install, configure and use Exim, the Exim Filter Specification which is additional information on the Exim filter language, and HOWTO Documentation.

With respect to performance, Exim has been deployed in busy environments, often handling thousands of emails per hour efficiently. Exim is designed to deliver email immediately, without queueing. However, its queue processing performance is comparatively poor when queues are large (which happens rarely on typical low-traffic sites, but can happen regularly on high-traffic sites). Qmail and Postfix have a central queue manager (qmail-send, qmgr), which Exim do not have, which makes it difficult to perform centralized load balancing, either of queue processing (leading to disproportionate amounts of time being spent on processing the same queue entries repeatedly) or of system-wide remote transport concurrency (leading to a "thundering herd" problem when multiple messages addressed to a single domain are submitted at once) [Wikipedia, 2010].

Like Postfix, Exim is a Sendmail drop-in replacement. In 2007 Philip Hazel was retired from the University of Cambridge, and the maintenance of Exim transitioned to a team of maintainers. Since then the release rate has slowed. The Exim website is http://www.exim.org/.

Dovecot

Dovecot is an open-source IMAP and POP3 email server for Linux/UNIX-like systems, written with security in mind [dov, 2010]. Dovecot was first released in July 2002, and is developed by Timo Sirainen. A part of code is under Massachusetts Institute of Technology (MIT) license, and everything else is LGPLv2.1. The last stable version is 2.0.0. Dovecot is an excellent choice for both small and large installations. It is fast, simple to set up, requires no special administration and it uses very little memory. The website of Dovecot is http://www.dovecot.org/. Good documentation and source code can be found in this website.

Its main features are support the standard mbox and Maildir formats, is standards compliant, the indexes are self-optimized, is self-healing, tries to be admin-friendly, support IMAP4rev1 and POP3, IPv6, SSL and TLS, as well as multiple commonly used IMAP extensions, plugins (quota and ACLs), plaintext and non-plaintext (CRAM-MD5, DIGEST-MD5 and others) authentication mechanisms, interaction with databases (LDAP, PostgresSQL, MySQL, SQLite), works with NFS and clustered filesystems, support Maildir++ quota.

The design and implementation is highly focused on security. The author offer 1000 EUR of own money to the first person to find a security hole from Dovecot [dov, 2010].

Courier IMAP

The Courier IMAP server is a module of the Courier Mail server and is distributed under the terms of the GNU General Public License (GPL). The last stable release is 4.8.0, available at http://www.courier-mta.org/imap. Good documentation and source code can be found in the same location. This module implements IMAP and POP protocols. The main features are support IMAP4rev1 and POP3, IPv6 support, IMAP extensions, shared folders, IMAP over SSL and STARTTLS, authentication using SSL certificates, authenticating mail accounts using the module Courier Authentication Library, ability to restrict the maximum number of IMAP logins, and the maximum number of logins from the same IP address, virtual mailboxes, maildir support only, maildir++ quota support, IMAP/POP3 proxying. The Courier IMAP server do not support maildir++ quota over 2GB - 1 byte.

The module Courier Authentication Library suport plaintext and non-plaintext authentication mechanisms. This has various authentication modules such as authpwd(looks up userids and passwords in your /etc/passwd file), authshadow (use shadow password files), authpam, authuserdb (uses GDBM or DB database files), authcram (CRAM-MD5 authentication), authldap, authmysql, authpgsql.

Cyrus IMAP

The Cyrus IMAP server is a module of Cyrus Project developed at Carnegie Mellon University, which prior to 1994 is exclusively used the locally-developed and non-standard Andrew Messaging System (AMS) for its email communication needs [cyr, 2010]. The last stable version is 2.3.16 and is under Carnegie Mellon University License. The website of Project Cyrus is http://cyrusimap.web.cmu.edu/. The Cyrus IMAP server provides access to personal mail and system-wide bulletin boards through the IMAP protocol. This server is a scalable enterprise mail system designed for use from small to large enterprise environments using standards-based technologies. This module full implementation allows a seamless mail and bulletin board environment to be set up across multiple servers. It differs from other IMAP server implementations in that it is run on ”sealed” servers, where users are not normally permitted to log in. The mailbox database is stored in parts of the filesystem that are private to the Cyrus IMAP system. All user access to mail is through software using the IMAP, POP3, or Kerberized Post Office Protocol (KPOP) protocols. The private mailbox database design gives the server large advantages in efficiency, scalability, and administratability. Multiple concurrent read/write connections to the same mailbox are permitted. The server supports ACLs on mailboxes and storage quotas on mailbox hierarchies, IMAP4rev1 protocol, IMAP extensions, SIEVE(RFC3028) for server side email filtering, any authentication mechanism available from the Simple Authentication and Security Layer (SASL) library (KERBEROS V4, GSSAPI, CRAM-MD5, DIGEST-Message-Digest Algorithm 5 (MD5), OTP, PLAIN, and STARTTLS), IMAP and POP3 over SSL [cyr, 2010].

2.4.3 Mail Storage Format

In the Unix/Linux world there are two ways to store e-mail: mbox and maildir format. Some MTAs implementations support both ways, such as postfix, qmail, exim.

Mbox Format

Mbox is the traditional way of storing mail messages and is one standard of the Unix world. The mbox format (figure 2.5) use a regular text file which serves as the mail users mailbox. Mbox is a text based storage system and is used in a lot of mail clients on a wide variety of platforms. For example Eudora (Windows, MacOS), Mozilla Thunderbird (Windows, Linux, MacOS) and others, uses this format for storing its mail. Some MTAs uses this format as well, such as Sendmail and Microsoft Exchange Server.

How Mbox works [mbo, 2009]:

- **Receiving and storing a mail**
  1. Lock the mailbox.
  2. Append the header (usually “From [senders email address] [date and time received]”) and the mail into the mailbox file.
  3. Unlock the mailbox.
- **Retrieving a mail**
  1. Lock the mailbox.

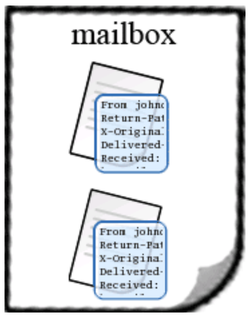

Figure 2.5: Mbox storage format

2. Locate and read the mail.
3. Update the mail status flag.
4. Unlock the mailbox.

- **Deleting a mail**
  1. Lock the mailbox.
  2. Move the contents of the mailbox, beginning from the position right after the mail to be deleted until the end of the mailbox, into the position of the mail to be deleted.
  3. Reduce the size of the mailbox file by the size of the deleted mail.
  4. Unlock the mailbox.
- **Searching a mail**
  1. Lock the mailbox.
  2. Search the mailbox.
  3. Unlock the mailbox.

The main advantages include: universally supported format, the addition of a new mail into the mailbox file and the search for text inside a single mailbox file are fast. The disadvantages are: file lock problems, problems when used with network file systems and this format is prone to corruption.

Maildir Format

The Maildir is a more recent format than Mbox, and was created by Bernstein when he created Qmail. This format creates a directory for each mail user, usually named *Maildir*. Under this directory there are three more directories named new, cur and tmp (figure 2.6) [mbo, 2009].

How Maildir works:

- **Receiving and storing a mail**
  1. Create a unique file in the tmp directory.
  2. Write the mail into the newly created file.
  3. Move the completely written mail into the new directory.
- **Retrieving a mail**
  1. Locate and read the mail.

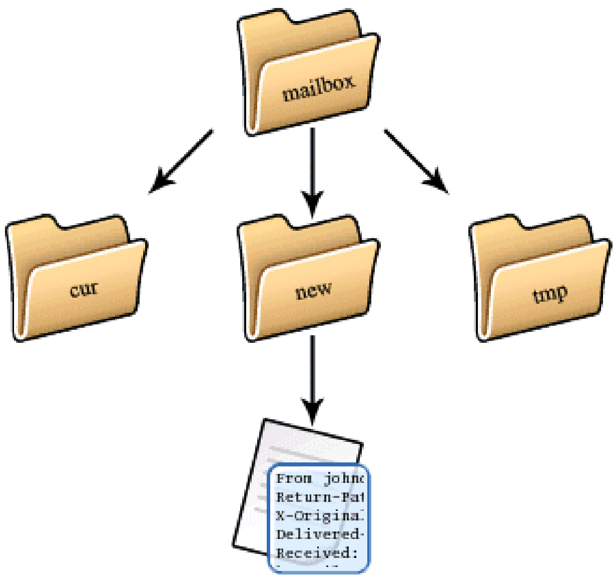

Figure 2.6: Maildir storage format

2. Move the mail from new into the cur directory and append the mail status flag into the filename.

- **Deleting a mail**
  1. Delete the file containing the mail.
- **Searching a mail**
  1. Search each and every mail file.

The main advantages are: the locating, retrieving and deleting a specific mail is fast, minimal to no file locking needed, can be used on network file system and is immune to mailbox corruption (assuming the hardware will not fail). The disadvantages are: some filesystems may not efficiently handle a large number of small files, and the searching text, which requires all mail files to be opened is slow [mbo, 2009].

Chapter 3

Test Scenarios

The main contribution of this dissertation is related to the analysis of the performance of e-mail services. To achieve some conclusions, it is necessary to define a test scenarios and to retrieve information about the behaviour of several e-mail server applications. As such, a set of tests must be defined, as well as the architectures that are hosting the e-mail servers. We intend to conduct tests to verify the scalability and performance of the SMTP on different kinds of hosting architecture (single computer, cluster and cloud) and with different applications (Qmail, Postfix). Since they are related, we also intend to perform tests to verify the scalability and performance of the POP3 and IMAP on the same architectures (single, cluster, cloud), whith the applications Qmail, Courier and Dovecot.

3.1 Tests

The tests are performed by measuring the number of sessions per ammount of time, in several test scenarios. The test period is fixed, thus varying the number of client machines, the number of connections and the number of e-mail destinations (only for SMTP. For POP3 and IMAP the e-mail address is fixed, as well). Therefore we have chosen to perform tests with the following scenarios:

- SMTP
  - 1C 1S - one client machine and one SMTP sessions injector intance, with 1, 5 and 10 e-mail address destinations.
  - 3C 1S - three clients machines, one SMTP sessions injector intance in each machine, with 1, 5 and 10 e-mail address destinations.
  - 3C 2S - three clients machines, two SMTP sessions injector intances in each machine, with 1, 5 and 10 e-mail address destinations.
- POP3 and IMAP
  - 1C 1S - one client machine, one POP3 and IMAP sessions injector intance.
  - 3C 1S - three clients machines, one POP3 and IMAP sessions injector intance in each machine.
  - 3C 2S - three clients machines, two POP3 or IMAP sessions injector intances in each machine.

To inject the traffic SMTP, POP3 and IMAP we used the tool SLAMD Distributed Load Generation Engine. This application is Java-Based and is designed for stress testing and performance analysis of network-based applications [sla, 2010]. Initial development of SLAMD was performed at Sun Microsystems, with recent development sponsored primarily by UnboundID Corp [sla, 2010]. It is available under the terms of the Sun Public License, which is an Open Systems Interconnection (OSI) approved open source license, and its main site is www.slamd.com [sla, 2010]. It was originally developed for the purpose of benchmarking and analyzing the performance of LDAP directory servers, however, it is also well-suited for testing other kinds of network applications and has been used for Web servers and Web-based applications, relational databases, and e-mail servers stress analysis. It can also be used for non-network based applications (and in fact, it is used for comparing things like CPU power and memory latency across a number of different kinds of systems), although it's distributed nature makes it ideal for systems that can be accessed remotely [sla, 2010]. It also provides a Java-based API to make it possible to quickly develop custom workloads, and it also contains an embedded scripting engine that can make it easy to stress applications using protocols like LDAP, HTTP, SMTP, IMAP, and POP, or any database that can be accessed via JDBC [sla, 2010]. The SLAMD has a server and a client versions. The server version has a Web interface for management where tests are defined and where the test results are received. The client version has two variants: client and monitor client. The client receives the server's definition of the tests, executes them and returns the results to the server. The client monitor monitors the client and the target servers and sends monitoring data to the server. To define a test, called a job in SLAMD, several parameters must be set. The most important parameters for defining an SMTP job are the following (those with an * are required):

- Start Time (YYYYMMDDhhmmss)
- Duration (seconds) - the duration will always be 60 seconds
- Number of Clients * - the number of client will be 1, 3 or 6.
- Threads per Client * - the number of threads will always be 20
- Statistics Collection Interval (seconds) - the interval will always be 10 seconds
- SMTP Server Address * - MX IP address
- SMTP Server Port * - 25
- From Address - raposo@ipb.pt
- Recipient Address * - depending on the number of destinations: teste1@wifi.ipb.pt teste[1-5]@wifi.ipb.pt or teste[1-10]@wifi.ipb.pt
- Minimum Number of Recipients * - equal to the number of destinations
- Maximum Number of Recipients * - equal to the number of destinations
- Message Body Size (bytes) * - will always be 1024
- Time Between SMTP Sessions (ms) - this time will always be 0
- Max Request Rate (Requests/Second/Client) - this value will always be -1, which indicates that the client should attempt to perform requests as quickly as possible.
- Max Rate Enforcement Interval (Seconds) - this value will be 0, which indicates that it should be equal to the statistics collection interval.

The most important parameters to define a POP3 job are (those with a * are required):

- Start Time (YYYYMMDDhhmmss)
- Duration (seconds) - the duration will always be 60 seconds
- Number of Clients * - the number of client will be 1, 3 or 6.
- Threads per Client * - the number of threads will always be 20

- Statistics Collection Interval (seconds) - the interval will always be 10 seconds
- POP Server Address * - MX IP address
- POP Server Port * - 110
- User ID * - teste1@wifi.ipb.pt
- User Password * - (password-of-teste1)
- Time Between POP Sessions (ms) * - this time will always be 0
- Max Request Rate (Requests/Second/Client) - this value will always be -1, which indicates that the client should attempt to perform requests as quickly as possible.
- Max Rate Enforcement Interval (Seconds) - this value will be 0, which indicates that it should be equal to the statistics collection interval.

The most important parameters to define a IMAP job are (those with a * are required):

- Start Time (YYYYMMDDhhmmss)
- Duration (seconds) - the duration will always be 60 seconds
- Number of Clients * - the number of client will be 1, 3 or 6.
- Threads per Client * - the number of threads will always be 20
- Statistics Collection Interval (seconds) - the interval will always be 10 seconds
- IMAP Server Address * MX IP address
- IMAP Server Port * - 143
- User ID * - teste1@wifi.ipb.pt
- User Password * - (password-of-teste1)
- Time Between IMAP Sessions (ms) * - this time will always be 0
- Max Request Rate (Requests/Second/Client) - this value will always be -1, which indicates that the client should attempt to perform requests as quickly as possible.
- Max Rate Enforcement Interval (Seconds) - this value will be 0, which indicates that it should be equal to the statistics collection interval.

Between each test, some maintenance operations are performed:

- Maildirs are cleaned,
- SLAMD clients are restarted,
- Services under test are restarted,
- Caches are erased in all the computers.

3.2 Applications

We will have two main groups of applications, SMTP applications and POP3/IMAP applications. These applications mentioned above are dependent on other applications, such as openldap, db, daemon-tools, uscpi-tcp. The description of the steps to install and configure these applications can be found in Appendix A.

3.2.1 SMTP Applications

For SMTP, we will install and configure the applications Qmail and Postfix. Qmail will be installed from source code, so it is necessary to download the source code, configure the Makefile, compile and install. Before compiling and installing it is necessary to install the dependencies (openldap,daemon-tools, uscpi-tcp) and create local users and groups that Qmail needs. Once installed it has to be configured. The configuration is contained in several small text files, housed in the directory */var/qmail/control*. It is also necessary to create a set of scripts needed to manage the service. These scripts are housed in the directory */var/qmail/supervise* and will allow starting, stoping, restarting, checking the status, checking the queue and logging (more detailed installation and configuration can be found in Appendix A). Postfix will be installed from the available packages, as its' dependencies (openldap). The configuration is spread over two main files (*main.cf*, *master.cf*) and two small files that contain settings related to authentication and LDAP search. Since Postfix has mechanisms that protect against the exceeding of requests coming from an IP or range of IP addresses, it is necessary to add the test network to the local networks in the configuration. More details can be found in Appendix A.

3.2.2 POP3/IMAP Applications

For POP3/IMAP, we will install and configure the applications Qmail, Courier and Dovecot. Qmail has already been installed for the SMTP, so it is only necessary create the scripts to manage this services. Courier will be installed from the available packages, as its' dependencies (openldap). The configuration is spread over four main files (*imapd*, *pop3d*, *authdaemonrc*, *authldaprc*). The maximum number of connections per IP was increased to 50. More details can be found in Appendix A. Dovecot will be installed from source code, so it is necessary to download the source code, configure the Makefile, compile and install. Before compiling and installing it is necessary to install the dependencies (openldap). Once installed it has to be configured. The configuration is contained in several small text files, housed in the directory */usr/local/etc/dovecot*. The maximum number of connections per IP was increased to 50. More details can be found in Appendix A.

3.3 Architecture

In this work, we will address the following architectures:

- single computer – will be the basis for comparison,
- computer cluster,
- cloud computing.

Computers with the following characteristics will be used to build the previous test scenarios, organizing them in cluster and in cloud, as well as using a single station for comparison:

- CPU - Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz
- RAM - 4GB
- 100Mbps ethernet connection
- Operating System (OS) Linux Ubuntu Lucid Server - version 64bits
- System partitions with ext4 filesystem

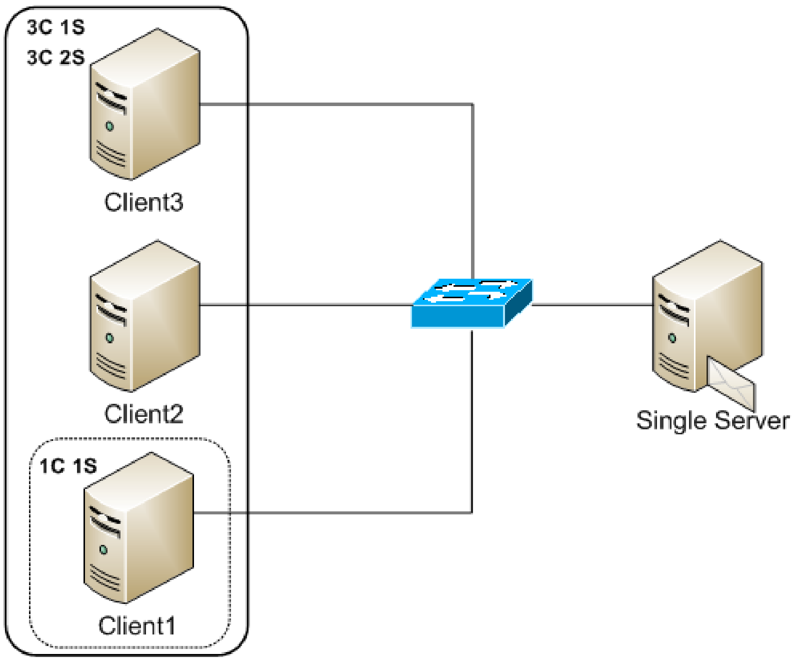

Figure 3.1: Single Server

3.3.1 Single Computer

In this architecture we will have a single computer and three computers, that will operate as clients in testing, connected all to the same switch (Figure 3.1).

All computers have one IP address of the same network 193.137.106.0/25. The gateway will be 193.137.106.126, dns servers will be 193.136.195.220 and 193.136.195.219 and the domain search will be wifi.ipb.pt.

3.3.2 Cluster

In this architecture we will use a load balancing cluster with direct routing. We will have two versions with different number of nodes: one will have two nodes and the other three (Figure 3.2). In addition, we also have three computers that will operate as clients. All computers will be connected to the same switch and share the same IP network (193.137.106.0/25). The gateway will be 193.137.106.126, dns servers will be 193.136.195.220 and 193.136.195.219 and the domain search will be wifi.ipb.pt.

To implement the load balancer we will use the LVS-DR. The LVS consists of two programs, the ipvsadm and the ldirectord. Ipvsadm is used to set up, maintain or inspect the virtual server table in the Linux kernel and the ldirectord which is used to monitor and manage real servers. The ipvsadm makes load-balancing of services and routing to each node while ldirectord monitors the status of the service on each real server and in case of service failure in one of us it removes the service/real server in the ipvs routing table, returning to replace it once it is back up again. LVS-DR will be used, so that each real server and load balancer will have a Virtual IP (VIP), which is the IP associated with the service(s) we want to balance. In the load balancer the VIP will be assigned to a virtual interface associated with the physical interface, while in the real servers will be assigned to a virtual loopback interface. Both IPs and VIPs need to belong to the same network segment. It is also necessary to disable Address Resolution Protocol (ARP) for VIP of real servers, because, otherwise, race conditions in ARP response could happen, causing the load balancer and the real servers to ARP response for VIP simultaneously. As a consequence, the router might send requests, intended for VIP, to real servers instead of the load

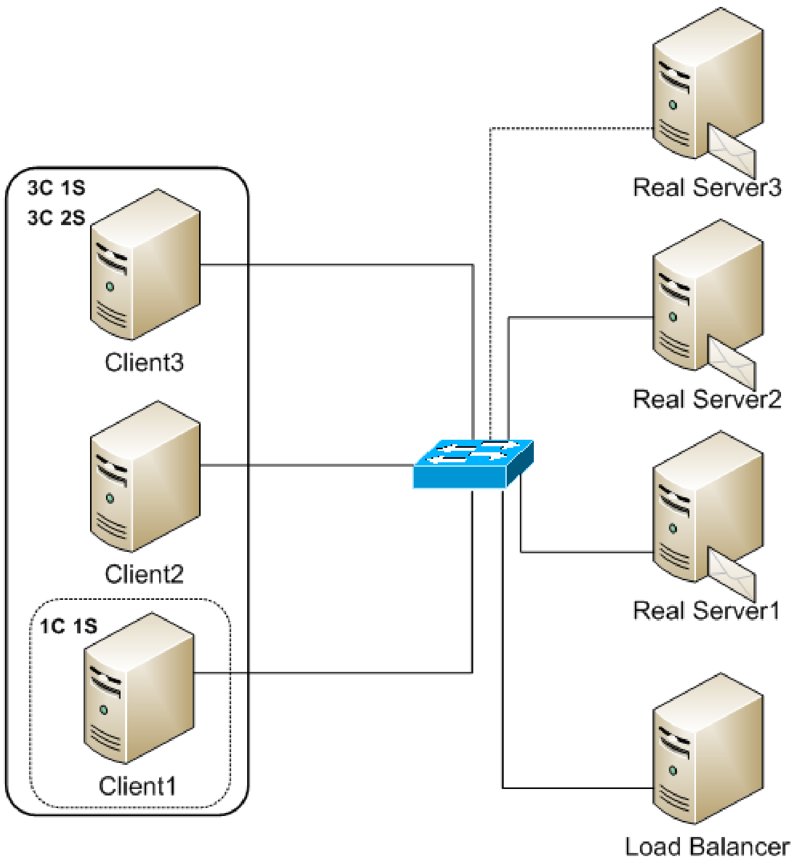

Figure 3.2: Cluster

balancer. With this architecture (Figure 3.2) when a client tries to contact a service, the request sent to VIP and is received by load balancer, which will forward the request to one of the real servers. The later will respond directly to the client thus avoiding the bottleneck effect in the load balancer.

3.3.3 Cloud

For testing the same scenarios in a cloud environment, we chose to use a private cloud, based on a cluster architecture and on Ubuntu UEC (Eucalyptus). In this architecture we installed four computers all connected to the same switch. One will be the cloud frontend and the others will be the nodes. The network cards of nodes are configured in bridge mode. We chose to se a single VMs in each node, with the following characteristics:

- Four Central Processing Units (CPUs) (QEMU Virtual CPU version 0.12.3 - 2666.220 MHz)
- 2048 Megabytes (MB) of RAM
- 20 GB of Disk
- 100Mbps ethernet connection
- Operating System (OS) Linux Ubuntu Lucid Server - version 64bits
- System partitions with ext4 filesystem

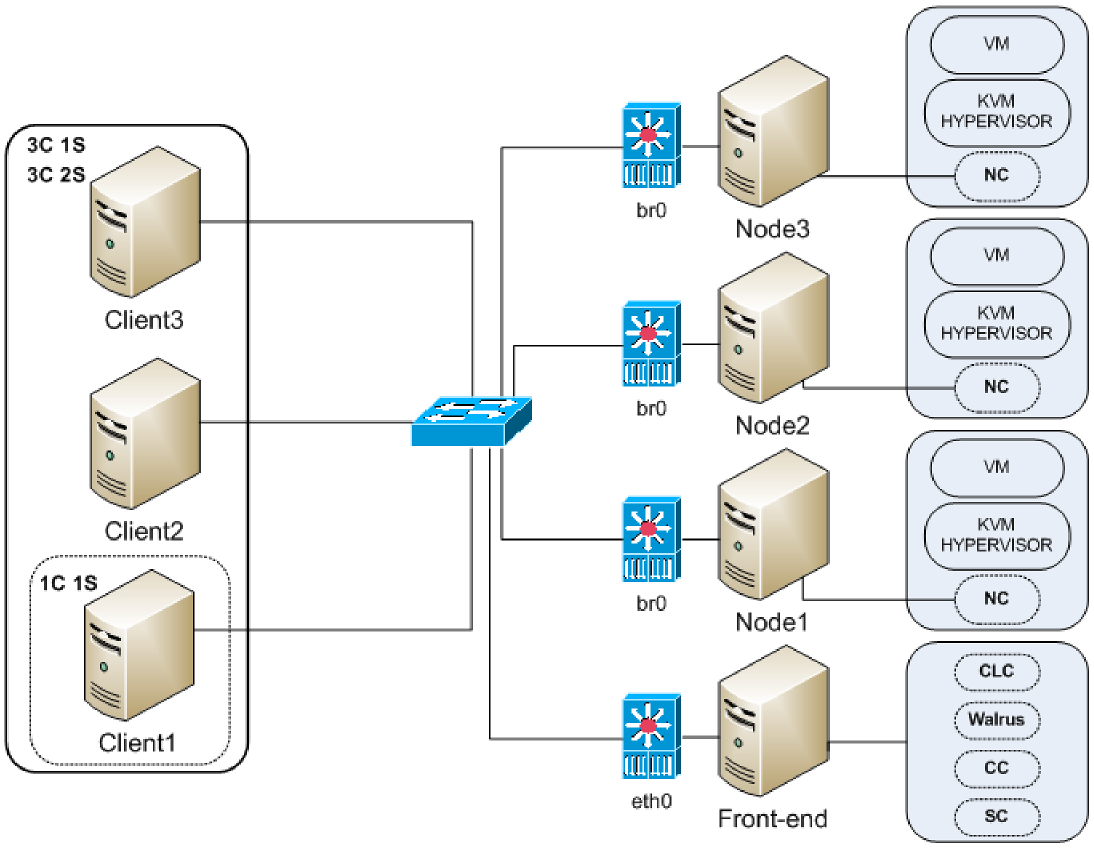

Figure 3.3: Cloud

One of the VM will be the load balancer and the other two will be the real servers in the cluster.

Eucalyptus requires a dedicated machine for deploying a private cloud infrastructure. In our scenario we have four machines: a frontend and three nodes (Figure 3.3). The frontend will host one Cloud Controller (CLC), one Cluster Controller (CC), one Walrus (the S3-like storage service) and one Storage Controller (SC). Each node will host a single Node Controller (NC). The NC main functions are Collection of data related to the resource availability and utilization on the node and reporting the data to CC, and instance life cycle mangement. The CC main functions include receiving requests from CLC to deploy instances, deciding which NCs to use for deploying the instances, controling the virtual network available to the instances, and collecting information about the NCs registered with it. The Walrus main functions include storing the machine images and snapshots, storing and serving files using S3 API. The SC main functions include creating persistent Elastic Block Storage (EBS) devices, providing the block storage over AoE or iSCSI protocol to the instances, and allowing creation of snapshots of volumes. The CLC main function is to have a knowledge of the availability and usage of resources in the cloud and the state of the cloud. The nodes must have Intel VT or AMD-V enabled support, because they will run a hypervisor, which by default is Kernel Based Virtual Machine (KVM). In the configuration of Eucalyptus VM profiles can be defined and, based on these profiles, each node can run one or more of these VMs. Since it was not feasible to implement clustering identical to the second scenario, because the frontend of the cloud is using NAT to associate a Public IP to Private IP of VMs, HAproxy was used to make the reverse proxy. In this architecture the clients' requests for services are received by CC and based in the target public IP and the NAT table is sent to the VM with the private IP corresponding. At that VM runs the HAproxy which will distribute the requests to the real servers VMs, using a round robin algorithm, and then they respond back via HAproxy which in turn responds by passing CC.

Chapter 4

Results

To be able to draw some conclusions, the tests presented in the previous chapter where performed and structured in two groups: SMTP and POP3/IMAP. The values where collected and processed, so that comparisons could be more meaninglful. The whole set of raw results are not presented in the this dissertation, because the sheer extension of data would increase the document length to over 600 pages.

4.1 SMTP Results

Production email servers usually have SMTP as well as POP and/or IMAP active. The behaviour of each protocol in terms of scalability is independent of the others, since it is active in different moments. For SMTP we are interested to know how it behaves in two ways: how each application perform when the hosting architecture changes and the differences of server applications in the same hosting architecture. For this, the following sections will cover:

- Qmail results obtained in the architectures: single, cluster and cloud.
- Postfix results obtained in the architectures: single, cluster and cloud.
- Qmail vs. Postfix results in each architecture: single, cluster and cloud.

As presented on the previous chapter, the tests are performed for 1C 1S for {1D, 5D, 10D}, 3C 1S for the same number of destination addresses as well as 3C 2S (Section 3.1).

4.1.1 Qmail Results

Qmail was tested for the previous three architectures, starting with the single server scenario. In this set, traffic generated from a single client with 20 threads, was injected, followed by traffic injected by three clients with 20 threads each and three clients with 40 threads each (Figure 4.1).

We can see that, in general there is an increase in the number of sessions from the single server scenario to the cluster based architecture. In the second scenario the introduction of another node in the cluster in general also led to a rise in the number of SMTP sessions. Increasing the number of destinations in general decreases the number of SMTP sessions but increase the number of e-mail recipients. Increasing the number of destinations decreases the number of sessions, but increases the number of emails delivered. Qmail achieved better results in the cluster environment with three nodes. For the cloud architecture, results are inferior to the other two. The inferior results for the third scenario (cloud) is due to the fact that the communications between clients and real servers in both directions pass through the CC and the load balancer, creating a bottleneck.

In all tests, and in all environments, it was observed that after ending the traffic injection, the server continued to process the incoming queue. In other words, it was not able to process all the requests,

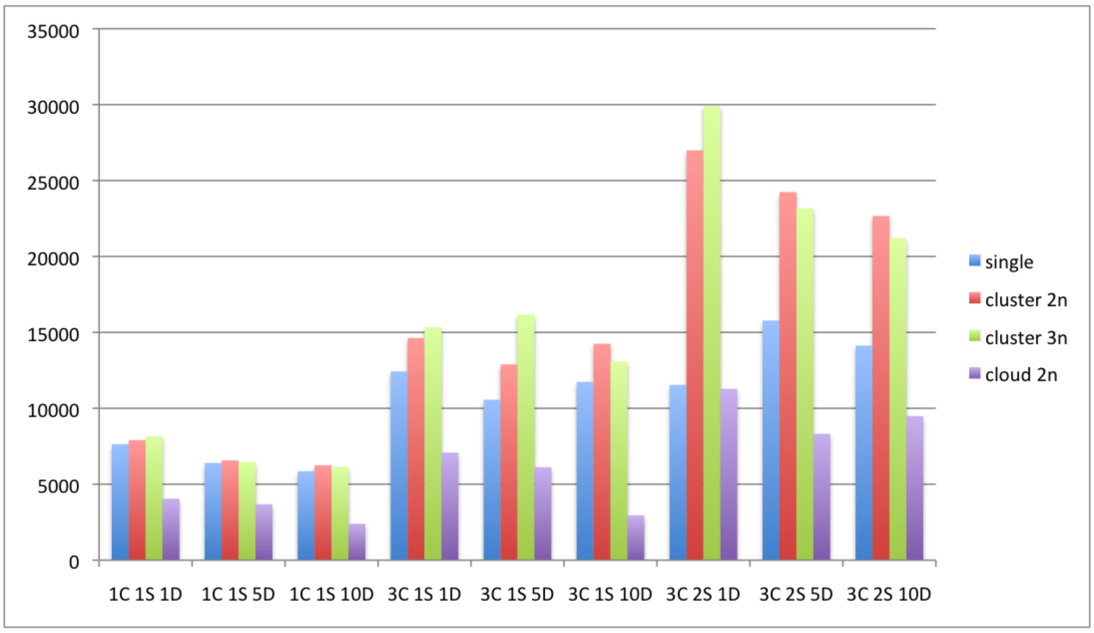

Figure 4.1: Qmail - Total SMTP sessions by scenario

continuing to deliver all e-mails to the mailbox for several minutes (from 10 to 25 minutes in the first scenario, 4 to 20 minutes in the second scenario with two nodes in cluster, 3 to 17 minutes in the second scenario with three nodes in the cluster and 3 to 15 minutes in the third scenario).

4.1.2 Postfix Results

With Postfix in general there is an increase in the number of sessions from the first to the second scenario. On the third scenario the results are inferior to the other two (Figure 4.2). This inferior results in the cloud scenario confirm the same problem found with Qmail – the bottleneck caused by the CC. In the second scenario the introduction of another node in the cluster in general also led to a rise in the number of SMTP sessions. Increasing the number of destinations in general decreases the number of SMTP sessions but increase the number of e-mail recipients. As in the previous case, Postfix achieved better results in the cluster environment with three nodes.

In all tests, and in all environments, Postfix managed to deliver almost all the e-mails during the injection time. There was a slight delay in just a few tests, revealing a better performance processing the queue:

- Single
  - 3C 1S 5D - delay of 2 minutes
  - 3C 1S 10D - delay of 4 minutes
  - 3C 2S 5D - delay of 3 minutes
  - 3C 2S 10D - delay of 7 minutes
- Cluster - two nodes
  - 3C 1S 10D - delay of 1.5 minutes
  - 3C 2S 5D - delay of 2 minutes
  - 3C 2S 10D - delay of 3 minutes

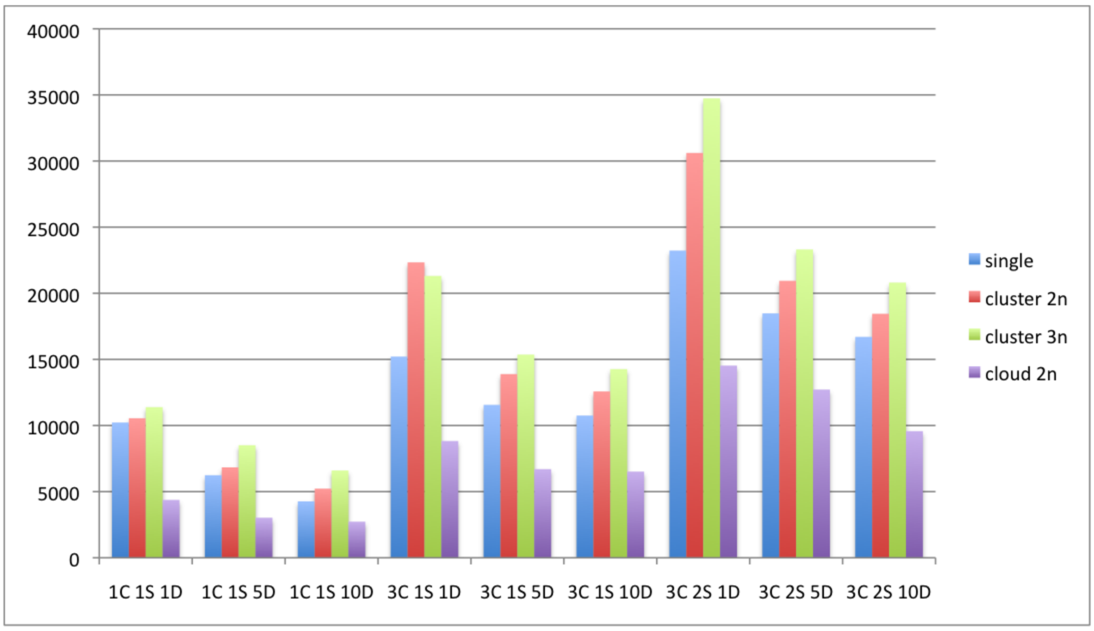

Figure 4.2: Postfix - Total SMTP sessions by scenario

- Cluster - three nodes
  - 3C 1S 10D - delay of 0.5 minutes
  - 3C 2S 5D - delay of 0.5 minutes
  - 3C 2S 10D - delay of 2minutes
- Cloud - two nodes
  - 3C 1S 5D - delay of 1 minute
  - 3C 1S 10D - delay of 1 minute
  - 3C 2S 5D - delay of 1 minute
  - 3C 2S 10D - delay of 3 minutes

4.1.3 Qmail vs. Postfix Results

To compare the performance of Qmail and Postfix in the test environments, we compared the results in each hosting architecture. In the single server environment (Figure 4.3), Postfix achieved better results for tests with just one destination. In tests with 1C 1S Postfix results where lower, but very close to Qmail. In tests with 3C 1S for five destinations the Postfix is overcame by Qmail and in tests with 3C 2S it achieved better results for all number of destinations, probably due to a drop in performance of Qmail. This was unexpected, due to the fact that Qmail code has been originally designed as a way for manage large mailing lists [qma, 2010].

In the cluster environment with two nodes (Figure 4.4), in test with 1C 1S compared to the previous environment there is an increase in the number of sessions, and Postfix is overcame by Qmail for five destinations. In tests with 3C 1S, there was only an increase in the number of sessions. Qmail has a better performance for one destination but it is still bellow that of Postfix. In tests with 3C 2S, Qmail greatly improved by recovering the best results for five and ten destinations, as expected.

In cluster with three nodes environment (Figure 4.5), in test with 1C 1S compared to the previous environment there is an increase in the number of sessions of Postfix, Qmail managed a lower number of

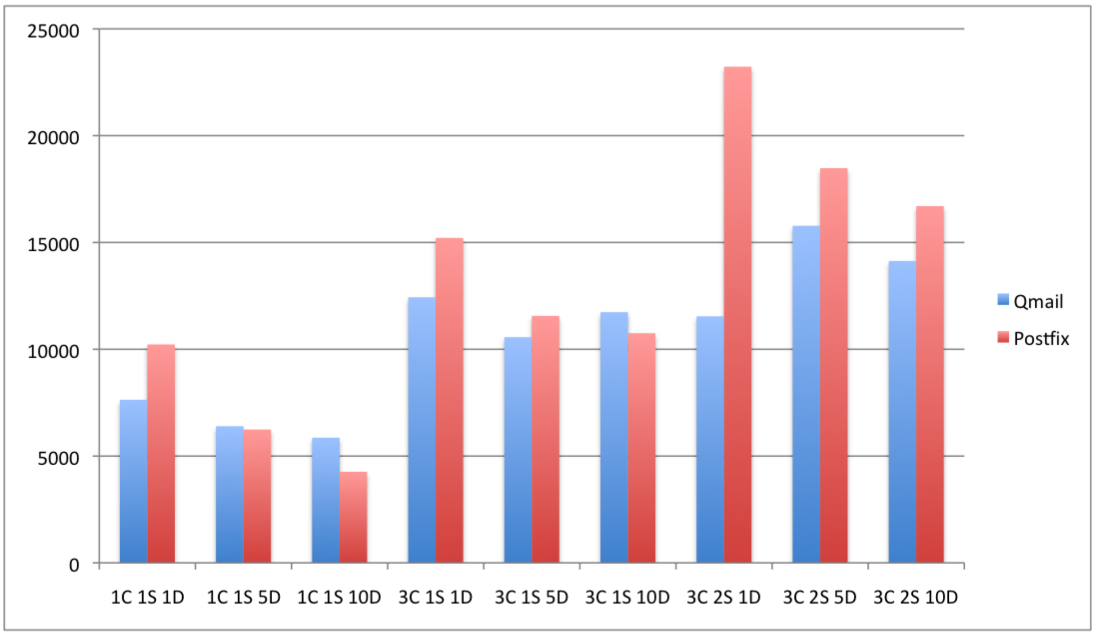

Figure 4.3: Qmail vs. Postfix - Single - Total SMTP sessions

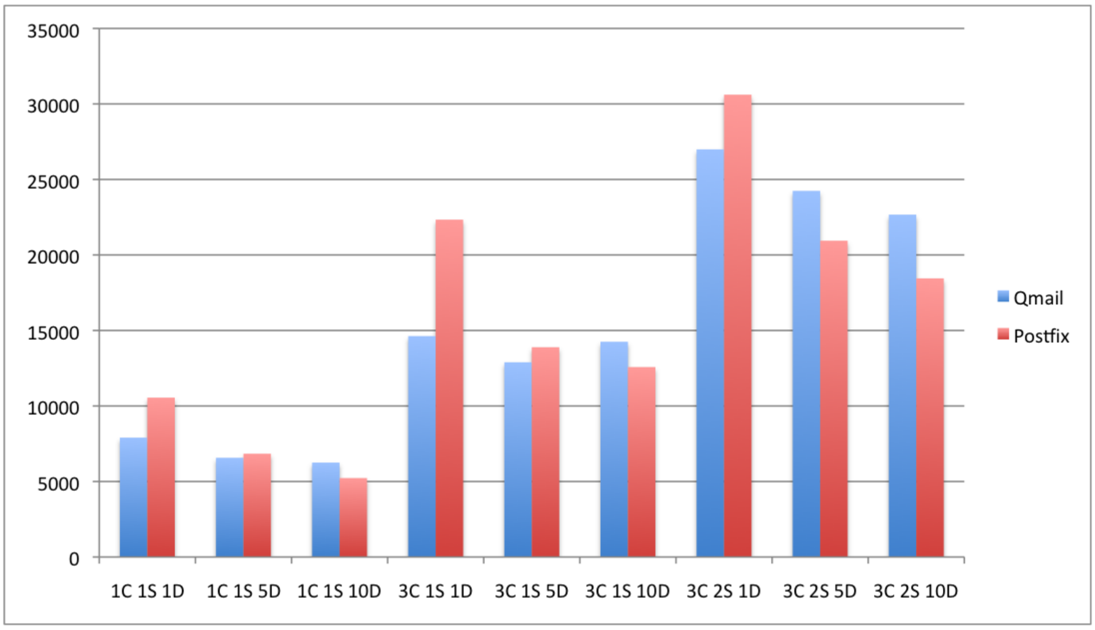

Figure 4.4: Qmail vs. Postfix - Cluster 2 Nodes - Total SMTP sessions

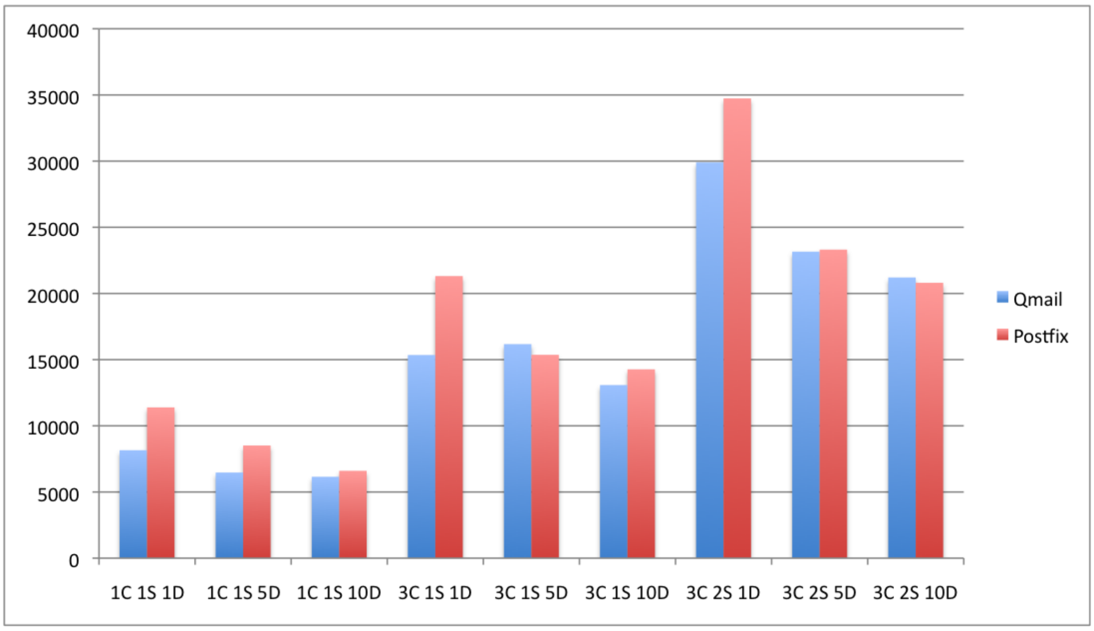

Figure 4.5: Qmail vs. Postfix - Cluster 3 Nodes - Total SMTP sessions

sessions to five and ten destinations. Postfix achieved better ressults for all number of destinations. In test with 3C 1S Postfix was less sessions to one destination, but more than Qmail. For five destinations was better and for ten destinations was Postfix. In tests with 3C 2S Qmail was less sessions to five and ten destinations. Postfix was better for five destinations, but Qmail is very close to them, and for ten destinations Qmail was better, but Postfix is very close to them.

In cloud with two nodes environment (Figure 4.6), compared to the previous environments both Qmail as Postfix had poorer results. Qmail only got better results than Postfix in tests with 1C 1S for one and five destinations.

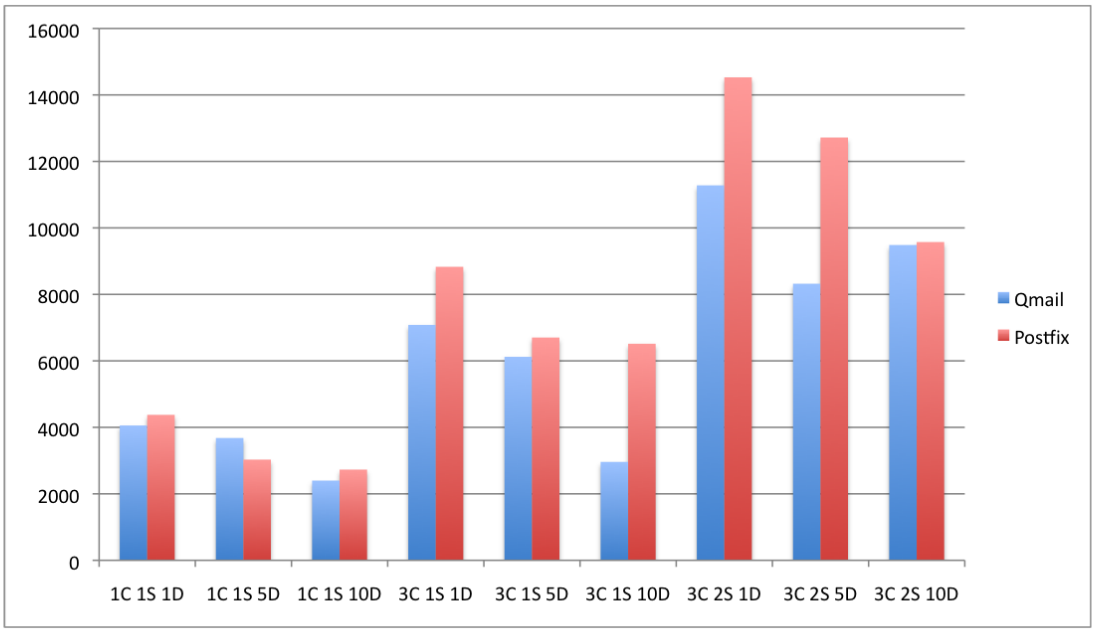

Figure 4.6: Qmail vs. Postfix - Cloud 2 Nodes - Total SMTP sessions

4.2 POP3/IMAP Results

For POP3/IMAP the results will be based on the server applications Qmail, Courier and Dovecot in the three architectures (single, cluster and cloud). The presentation of results will be distributed as follows:

- Qmail POP3 results obtained in architectures single, cluster and cloud.
- Courier POP3 results obtained in architectures single, cluster and cloud.
- Courier IMAP results obtained in architectures single, cluster and cloud.
- Dovecot POP3 results obtained in architectures single, cluster and cloud.
- Dovecot IMAP results obtained in architectures single, cluster and cloud.
- Qmail vs. Courier vs. Dovecot POP3 results for each architecture.
- Courier vs. Dovecot IMAP results for each architecture.

4.2.1 Qmail Results

In general, the POP3 results achieved by Qmail reveal an increase in the number of sessions from the first to the second and to the third scnenario. However, the results of the third scenario are inferior to the second (Figure 4.7). This inferior results confirm the problem found in SMTP, but, in this case, results are higher than the baseline scenario. Qmail achieved better POP3 results in the cluster three nodes environment.

4.2.2 Courier Results

The POP3 results achieved by Courier reveal an increase in the number of sessions from the first to the second and to third scnenarios. However, the results in the third scenario are inferior to the second (Figure 4.8). Courier achieved better POP3 results in the cluster three nodes environment.

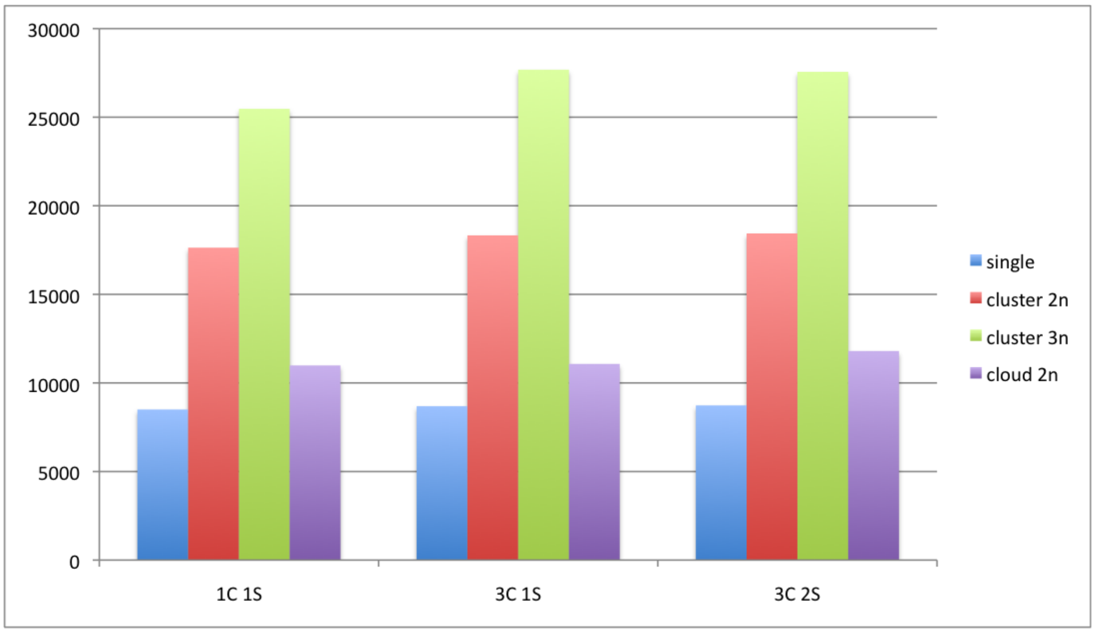

Figure 4.7: Qmail - Total POP3 sessions

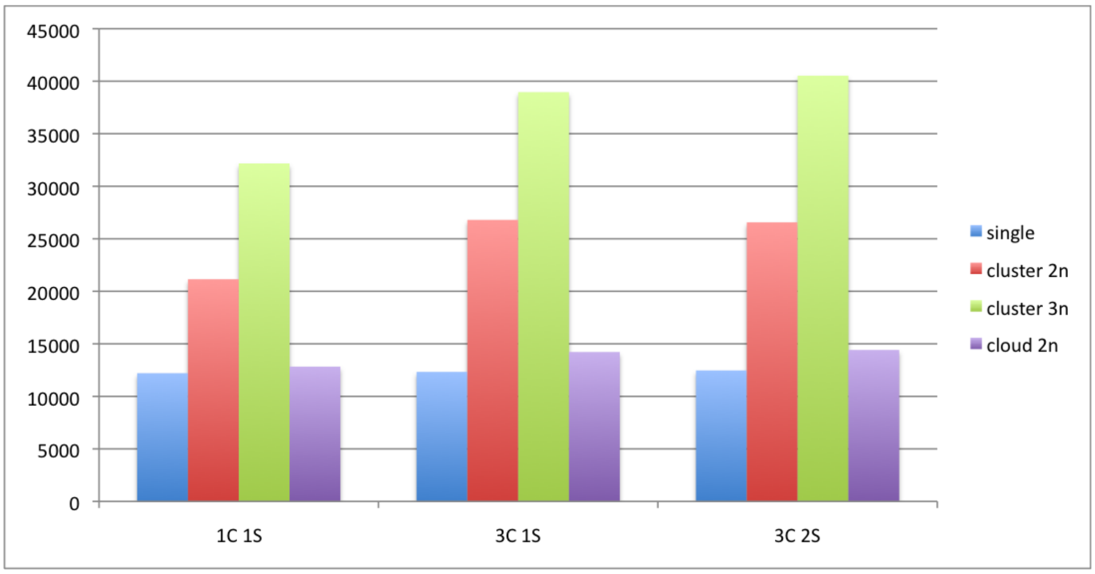

Figure 4.8: Courier - Total POP3 sessions

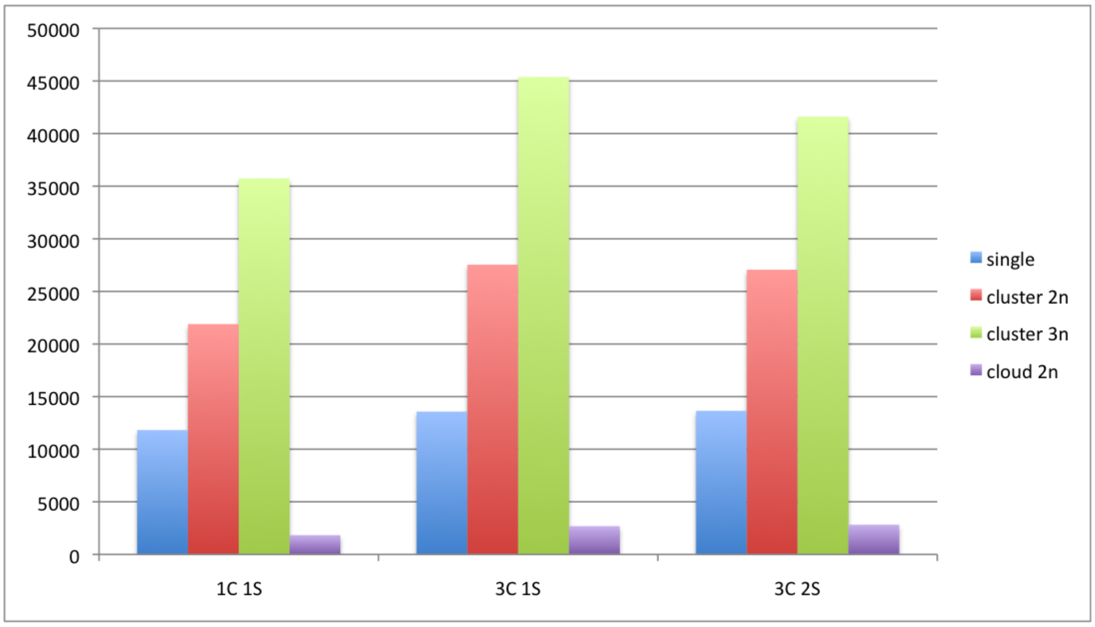

Figure 4.9: Courier - Total IMAP sessions

For IMAP, it was necessary to increase the maximum number of connections per IP address from 10 to 50. As expected, Courier performed better for IMAP in the cluster environment (Figure 4.9). Courier achieved better IMAP results in the cluster three nodes environment.

4.2.3 Dovecot Results

The POP3 results achieved by Dovecot follow the previous patterns (Figure 4.10). Dovecot achieved better results in the cluster three nodes environment. The same can be said for the IMAPimplementation of Dovecot (Figure 4.11). Dovecot achieved better IMAP results in the cluster three nodes environment.

4.2.4 Qmail vs. Courier vs. Dovecot POP3 Results

In single environment (Figure 4.12), Dovecot achieved better results than Courier and Qmail for all tests, but Qmail and Courier with the increasing number of clients increased slightly the number of sessions while Dovecot slightly decreases the number of sessions.

In cluster with two nodes environment (Figure 4.13), compared to the previous environment there is an increase in the number of sessions for all. Courier results are very close to the results of Dovecot, overtaking it slightly in test 3C 1S.

In cluster with three nodes environment (Figure 4.14), compared to the previous environment there is an increase in the number of sessions for all and the Courier was just better than Postfix in test with 3C 2S, but in other tests is very close to Postfix.

In cloud with two nodes environment (Figure 4.15), compared to the two previous environments there is an decrease in the number of sessions for all, but Qmail and Courier was better results than in single environment. Poor results of Dovecot is due to the fact that this defect have as a limitation on the number of POP3 connections per login. The default is 256. Also all connections come from the same IP, so it is very easy to reach this limit.

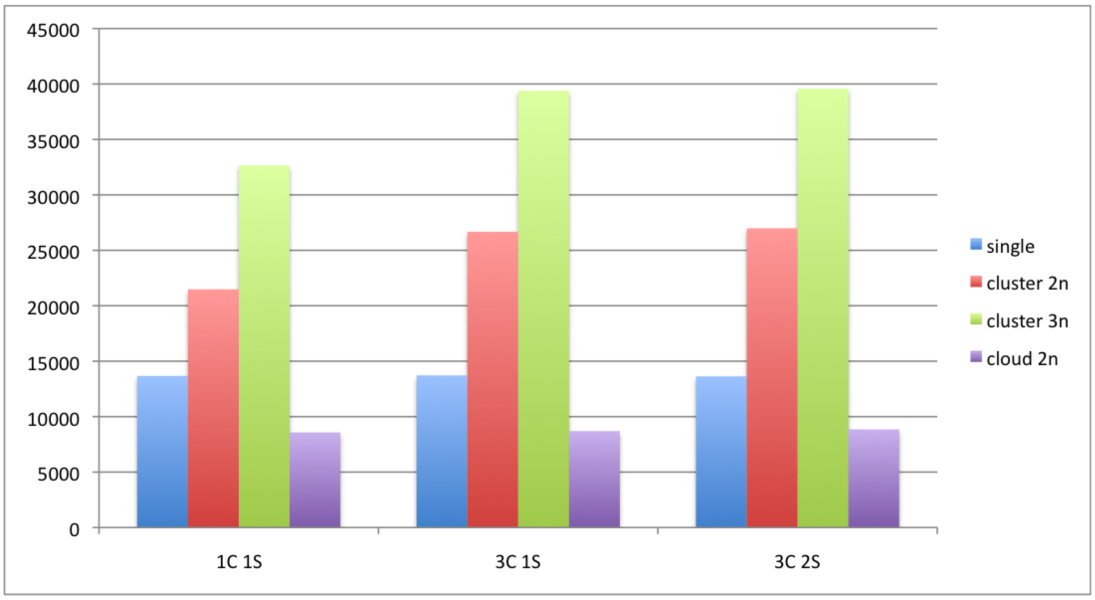

Figure 4.10: Dovecot - Total POP3 sessions

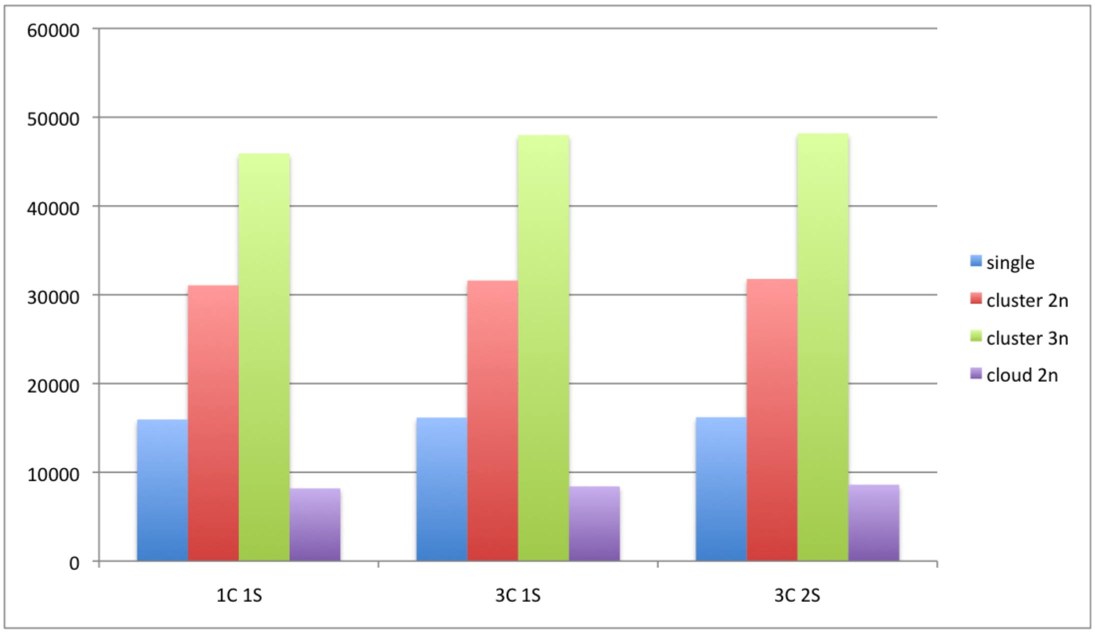

Figure 4.11: Dovecot - Total IMAP sessions

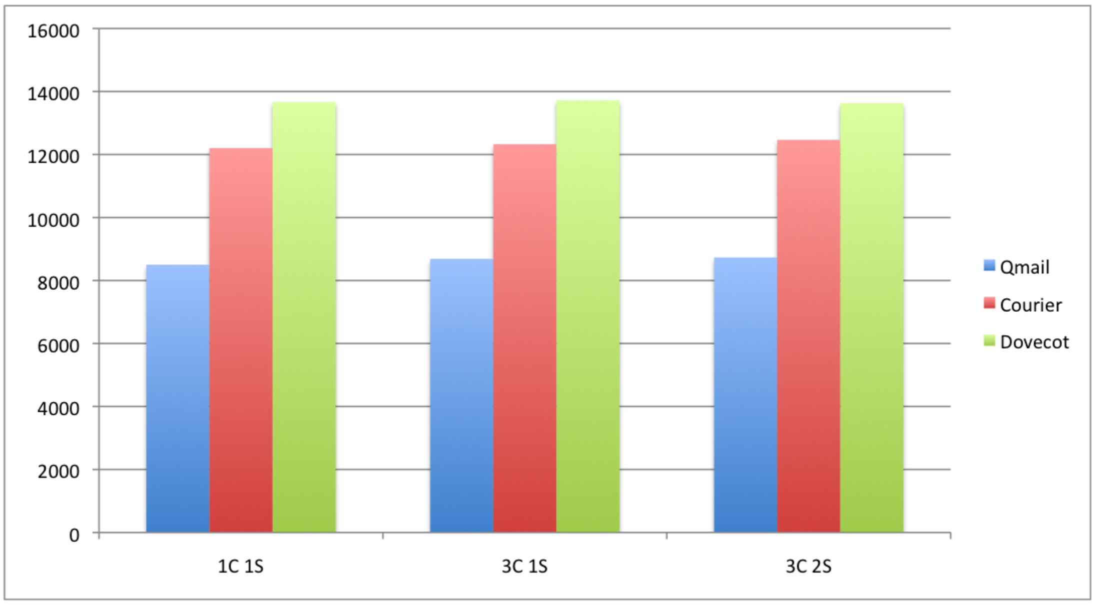

Figure 4.12: Qmail vs. Courier vs. Dovecot - Single - Total POP3 sessions

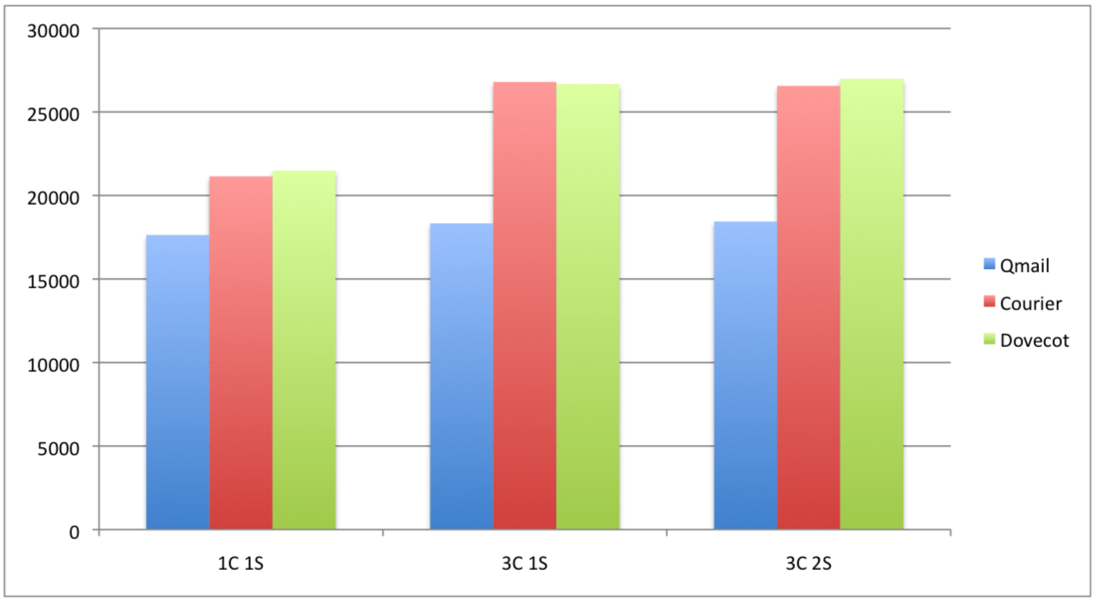

Figure 4.13: Qmail vs. Courier vs. Dovecot - Cluster 2 Nodes - Total POP3 sessions

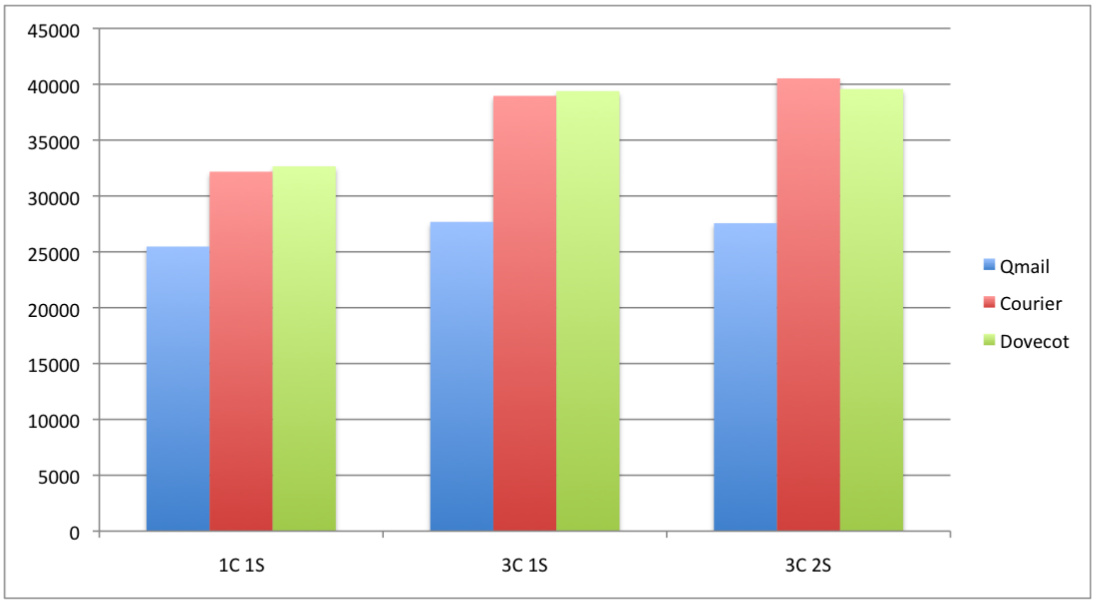

Figure 4.14: Qmail vs. Courier vs. Dovecot - Cluster3 Nodes - Total POP3 sessions

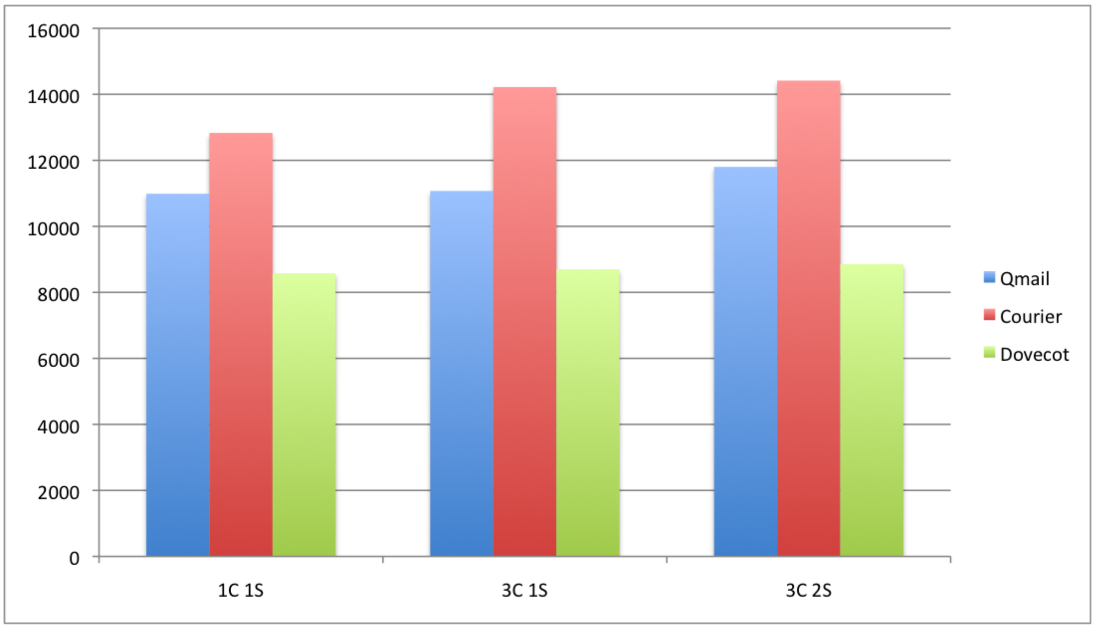

Figure 4.15: Qmail vs. Courier vs. Dovecot - Cloud 2 Nodes - Total POP3 sessions

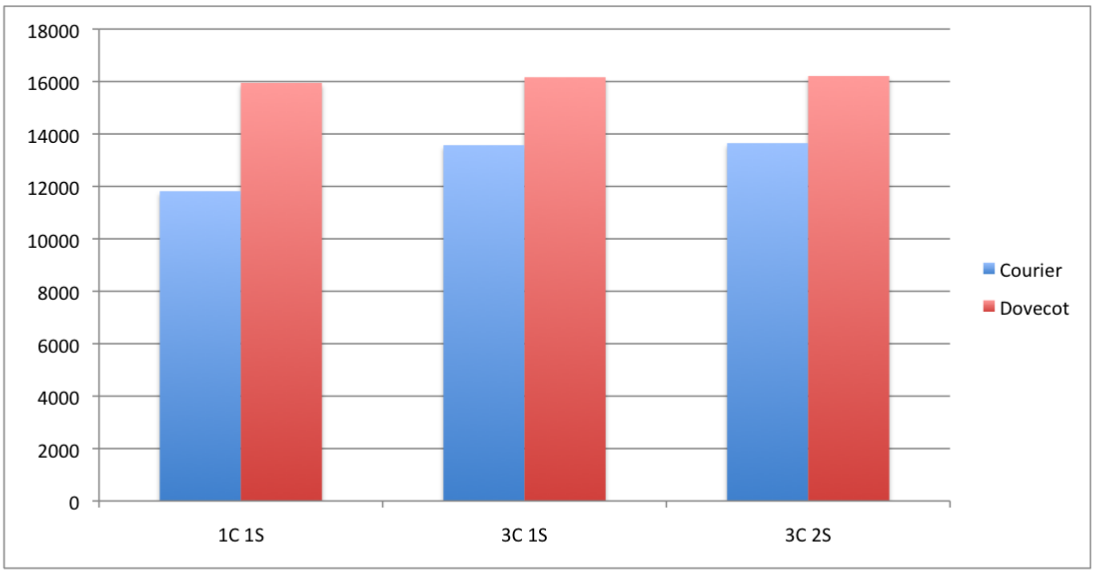

Figure 4.16: Courier vs. Dovecot - Single - Total IMAP sessions

4.2.5 Courier vs. Dovecot IMAP Results

In single environment (Figure 4.16), Dovecot achieved better results than Courier for all tests.

In cluster with two nodes environment (Figure 4.17), compared to the previous environment there is an increase in the number of sessions for all and Dovecot achieved better rresults than Courier for all tests again. Courier in test with 3C 2S had slightly worse results than int test with 3C 1S.

In cluster with three nodes environment (Figure 4.18), compared to the previous environment there is an increase in the number of sessions for all and Dovecot continues to get the best results for all tests. Courier is close to Dovecot in test with 3C 1S, but continues to have worse results in the test 3C 2S.

In cloud with two nodes environment (Figure 4.19), compared to the previous environments there is an decrease in the number of sessions for all. Courier even though the same number of connections per IP that Dovecot, had much lower results than Dovecot results.

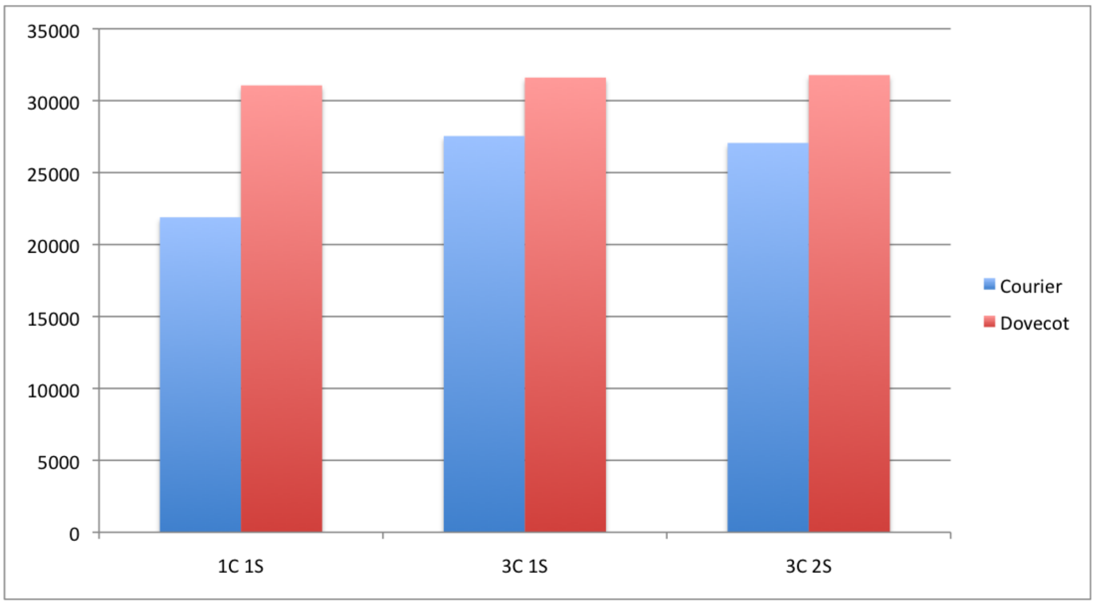

Figure 4.17: Courier vs. Dovecot - Cluster 2 Nodes - Total IMAP sessions

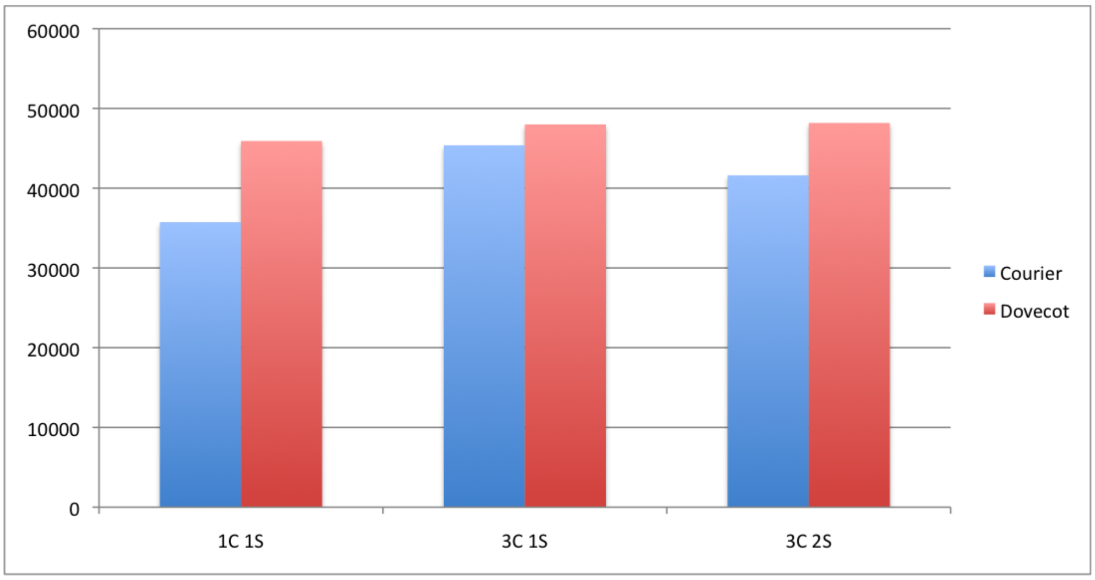

Figure 4.18: Courier vs. Dovecot - Cluster 3 Nodes - Total IMAP sessions

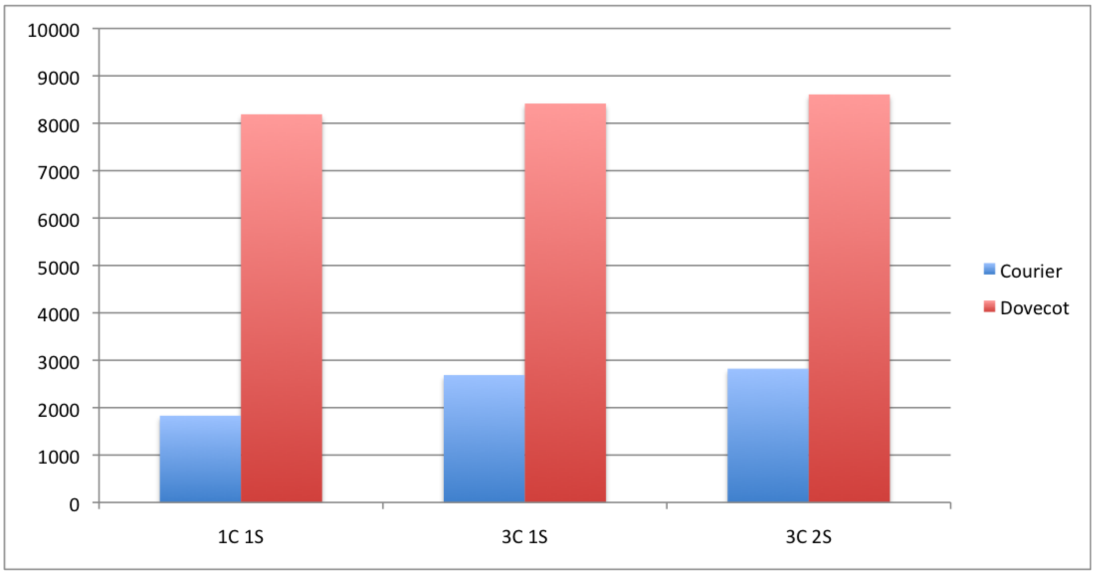

Figure 4.19: Courier vs. Dovecot - Cloud 2 Nodes - Total IMAP sessions

Chapter 5

Conclusions

The dependency on communication services, especially, the ones based on Internet technology, has been changing the way people and organizations work. E-mail is one of the most successful and well known, for it's flexibility, availaility and ease of use. Moreover, the prevalence of e-mail service in almost every kind of device, either for personal use or company use, makes it ubiquitous, in terms of time, place and opportunity. It is common, for example, for persons to have, simultaneously, several devices with e-mail access at the same time, for example, the mobile phone, laptop, iPod or others. From the organization point of view, it is strategic to maintain an e-mail contact address, providing a communication end-point operational 24/7.

This dissertation allowed a detailed study of the cluster, grid and cloud environments with focus on the concepts, architecture, infrastructure for hosting e-mail services. In addition to the study of computing environments, this dissertation also enabled the study of the characteristics of several open-source e-mail servers focusing on architectural details, platforms and infrastructure.

The main motivation was to gather some knowledge about which of these computing environments, applications services and email servers, as well as which combinations, are most appropriate in order to achieve better performance in server-to-server (SMTP) and in server-to-client (POP and IMAP) communication.

Based on the results we conclude that the best computing environment for implementation of e-mail services is the cluster environment with a load balancer with direct routing in almost all the scenarios. However, it is not clear the importance and the limits related to the number of nodes, pointing to a balance between the performance and associated cost.

On the server application side, Postfix was the application that achieved best overall results for the SMTP service. Moreover, Postfix also demostrated other advantages, when compared to Qmail, such as being activelly maintained and showing frequent releases, IPv6 support, multiple scan engines support and protection against spam.

For IMAP, Dovecot was the application that obtained the overall best results in all computing environments tested. For POP3 we can conclude that both the Courier and Dovecot have good results, not imposing a clear distinction. The choice should be based on features of each one.

5.1 Future Work

For future work, we would like to further study the cluster environment, in terms of the limitations and the behaviour of the servers in relation to the increase in the number of nodes. It is also not clear how the parallelization of hardware could be of help when dealing with the replication of e-mail messages in the same node (for example, for 5D or 10D, in the above scenarios).

Due to poor results obtained in the cloud environment, it is also interesting to think about testing other approaches, such as SaaS, and other server applications, such Zimbra Collaboration Suite, Gmail, Scalix and Atmail. These suites build on e-mail, extending it with other services such as collaboration suite, calendar services, and so on.

Bibliography

[lam, 2006] (2006). Lam faq: Heterogenenity and lam/mpi. http://www.lam-mpi.org/faq/category11.php3#question1.

[clu, 2009] (2009). Cluster suite overview - red hat cluster suite for red hat enterprise 5.

[mbo, 2009] (2009). Mbox vs maildir: Mail storage formats - linux mail server setup and howto guide. http://www.linuxmail.info/mbox-maildir-mail-storage-formats/. [Online; accessed 10-August-2010].

[top, 2009] (2009). Top500 supercomputers sites. http://www.top500.org/. [Online; accessed 23-January-2010].

[aws, 2010] (2010). Amazon web services. http://aws.amazon.com/. [Online; accessed 05-August-2010].

[dov, 2010] (2010). Dovecot - secure imap server. http://www.dovecot.org/. [Online; accessed 18-August-2010].

[euc, 2010] (2010). Eucalyptus systems. http://www.eucalyptus.com/. [Online; accessed 06-August-2010].

[exi, 2010] (2010). The exim home page. http://www.exim.org/. [Online; accessed 17-August-2010].

[gog, 2010] (2010). Gogrid cloud hosting. http://www.gogrid.com/cloud-hosting/. [Online; accessed 05-August-2010].

[gae, 2010] (2010). Google app engine. http://code.google.com/appengine/. [Online; accessed 05-August-2010].

[azu, 2010] (2010). Microsoft azure. http://www.microsoft.com/windowsazure/. [Online; accessed 06-August-2010].

[mil, 2010] (2010). milter.org - an interactive catalog of sendmail filters. http://www.milter.org/. [Online; accessed 16-August-2010].

[pos, 2010] (2010). The postfix home page. http://www.postfix.org/. [Online; accessed 16-August-2010].

[cyr, 2010] (2010). Project cyrus. http://cyrusimap.web.cmu.edu/. [Online; accessed 20-August-2010].

[qma, 2010] (2010). Qmail: the internet's MTA of choice. http://cr.yp.to/qmail.html. [Online; accessed 16-August-2010].

[rac, 2010] (2010). Rackspace cloud. http://www.rackspacecloud.com/. [Online; accessed 05-August-2010].

[sca, 2010] (2010). scalix. http://www.scalix.com/. [Online; accessed 16-August-2010].

[sen, 2010] (2010). Sendmail.org. http://www.sendmail.org/. [Online; accessed 16-August-2010].

[sla, 2010] (2010). Slamd distributed load generation engine. http://www.slamd.com/. [Online; accessed 06-October-2010].

[zim, 2010] (2010). Zimbra. http://www.zimbra.com/. [Online; accessed 16-August-2010].

[Al-Khannak et al., 2007] Al-Khannak, R., Bifzer, B., and Hezron (2007). Grid computing by using multi agent system technology in distributed power generator. In *Universities Power Engineering Conference, 2007*, pages 62 – 67, University of Brighton, Brighton, UK.

[Armbrust et al., 2009] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing.

[Bader and Pennington, 2001] Bader, D. and Pennington, R. (2001). Cluster computing: Applications. *The International Journal of High Performance Computing Applications*, 15:181–185.

[Baker, 2000] Baker, M. (2000). Cluster computing white paper.

[Betti et al., 2009] Betti, E., Cesati, M., Gioiosa, R., and Piermaria, F. (2009). A global operating system for HPC clusters. In *Cluster Computing and Workshops, 2009*, pages 1–10, New Orleans, Louisiana. IEEE Xplore.

[Bourke, 2001] Bourke, T. (2001). *Server Load Balancing*. O'Reilly Media.

[Chee-Wei and Chen-Khong, 2007] Chee-Wei, A. and Chen-Khong, T. (2007). Analysis and optimization of service availability in a HA cluster with load-dependent machine availability. *Parallel and Distributed Systems, IEEE Transactions*, 18(9):1307 – 1319.

[Crispin, 2003] Crispin, M. (2003). RFC 3501 - INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. http://tools.ietf.org/html/rfc3501.

[Esposito and Tarricone, 2003] Esposito, A. and Tarricone, L. (2003). Grid computing for electromagnetics: a beginner's guide with applications. *Antennas and Propagation Magazine, IEEE*, 45(2):91 – 100.

[Foster et al., 2001] Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *Intl J. Supercomputer Applications*, 15(3).

[Gens, 2008] Gens, F. (2008). Defining cloud services and cloud computing. http://blogs.idc.com/ie/?p=190. [Online; accessed 1-February-2010].

[Golanski, 2000] Golanski, Y. (2000). The exim mail transfer agent in a large scale deployment.

[Keith, 2009] Keith, P. (2009). Understanding public clouds: Iaas, paas, & saas. [Online; accessed 25-May-2010].

[Klensin, 2008] Klensin, J. (2008). RFC 5321 - simple mail transfer protocol. http://tools.ietf.org/html/rfc5321.

[Klensin and Gellens, 1998] Klensin, J. and Gellens, R. (1998). RFC 2476 - message submission. http://tools.ietf.org/html/rfc2476.

[Kouvatsos and Mkwawa, 2003] Kouvatsos, D. and Mkwawa, I. (2003). Multicast communication in grid computing networks with background traffic. *Software, IEEE Proceedings*, 150(4):257 – 264.

[Laine and Midorikawa, 2007] Laine, J. and Midorikawa, E. (2007). Using analytical models to load balancing in a heterogeneous network of computers. In *Parallel Computing Technologies*, volume 4671 of *Lecture Notes in Computer Science*, pages 559 – 568. Springer Berlin / Heidelberg.

[Li-Gu et al., 2006] Li-Gu, Z., De-Zhi, H., Shi-Zheng, Z., and Chang-Sheng, X. (2006). High availability cluster with combining NAS and ISCSI. In *Fifth Internacional Conference on Machine Learning and Cybernetics*, pages 4455 – 4460, Dalian. IEEE.

[Maxey, 2008] Maxey, M. (2008). Cloud computing public or private? how to choose cloud storage. [http://cloudcomputing.sys-con.com/node/707840?page=0,0 - Online; accessed 24-January-2010].

[Naiouf et al., 2006] Naiouf, M., Giusti, L., Chichizola, F., and Giusti, A. (2006). Dynamic load balancing on non-homogeneous clusters. In *Frontiers of High Performance Computing and Networking - ISPA 2006 Workshops*, volume 4331 of *Lecture Notes in Computer Science*, pages 65 – 73. Springer Berlin / Heidelberg.

[Newman, 1999] Newman, C. (1999). RFC 2595 - using TLS with IMAP, POP3 and ACAP. http://tools.ietf.org/html/rfc2595.

[Razak, 2009] Razak, S. (2009). Cloud computing in malaysia universities. In *Innovative Technologies in Intelligent Systems and Industrial Applications, 2009*, pages 101 – 106, Malaysia.

[Robertson, 2000] Robertson, A. (2000). The evolution of the Linux-HA project.

[Tarreau, 2006] Tarreau, W. (2006). Making applications scalable with load balancing.

[Wikipedia, 2010] Wikipedia (2010). Exim — wikipedia, the free encyclopedia. [Online; accessed 23-August-2010].

[Youseff et al., 2008] Youseff, L., Butrico, M., and Silva, D. (2008). Toward a unified ontology of cloud computing.

[Yves and Patrice, 2008] Yves, L. and Patrice, B. (2008). A technology and networking guide for high availability clusters extended across multiple data centers.

Appendix A

Appendix - Installations and Configurations

In this appendix will describe the procedures for installation and configuration of all software used in three scenarios set out in chapter three.

A.1 Operating System (OS)

The OS chosen for all machines was Linux and the distribution chosen was the Ubuntu Server 10.04 64bits. During installation of the OS was chosen to install only the base system. We created two partitions, swap and the system (/), and the system partition is formatted to ext4 filesystem. The range of IPs used belong to the network 193.137.106.0/25, the gateway of the network is 193.137.106.126 and the used DNS servers address are 193.136.195.220 and 193.136.195.219. After completing the base installation was installed a few packages that would be needed, as well as their dependencies. The following commands describe these procedures:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install ssh gcc
```

It was also configured the Network Time Protocol (NTP) service editing the file */etc/default/ntpdate* and change the following lines:

```
NTPDATE_USE_NTP_CONF=no
NTPSERVERS="ntp.ipb.pt"
```

Then it was added the following line in crontab so that the NTP service to synchronize the clock every hour:

```
@hourly /usr/sbin/ntpdate-debian
```

.

A.2 LDAP

The LDAP server was installed using the *Openldap* from sources with version 2.4.21. To install the openldap first had to also install *DB* from sources with version 4.7.25. The steps for installing the *DB-4.7.25* were as follows:

```
$ wget http://download.oracle.com/berkeley-db/db-4.7.25.tar.gz
$ tar -xzvf db-4.7.25.tar.gz
$ cd db-4.7.25/build_unix/
$ ../dist/configure
$ make
$ sudo make install
$ sudo echo '/usr/local/BerkeleyDB.4.7/lib/' > /etc/ld.so.conf.d/db.conf
$ sudo ldconfig
```

The steps for installing *openldap-2.4.21* were as follows:

```
$ wget
ftp://ftp.linux.pt/pub/mirrors/OpenLDAP/openldap-release/openldap-2.4.21.tgz
$ tar -xzvf openldap-2.4.21.tgz
$ cd openldap-2.4.21
$ ./configure --enable-overlays --enable-crypt --with-tls \
 CPPFLAGS=-I/usr/local/BerkeleyDB.4.7/include \
LDFLAGS=-L/usr/local/BerkeleyDB.4.7/lib
$ make depend
$ make
$ sudo make test
$ sudo make install
```

To configure the LDAP server for the scenarios it was necessary to perform the following steps:

- Copy from Qmail source the file *qmail.schema* to */usr/local/etc/openldap/schema/*.
- Define a *root* password for Openldap:

  ```
  $ slappasswd -h {MD5}
  ```

- Edit the file */usr/local/etc/openldap/slapd.conf* and change the change/add the next lines:

  ```
  include         /usr/local/etc/openldap/schema/core.schema
  include         /usr/local/etc/openldap/schema/cosine.schema
  include         /usr/local/etc/openldap/schema/nis.schema
  include         /usr/local/etc/openldap/schema/inetorgperson.schema
  include         /usr/local/etc/openldap/schema/qmail.schema

  database        bdb
  suffix          "dc=ipb,dc=pt"
  rootdn          "cn=root,dc=ipb,dc=pt"
  rootpw {MD5}FKjYPUvcocJlUdVQvcHNcA==

  directory       /usr/local/var/openldap-data
  index objectClass,member       eq
  index uid   eq
  index mailAlternateaddress   eq
  index mail                   eq
  index cn                   eq
  ```

- Start the openldap service:

```
   $ /usr/local/libexec/slapd &
```

- Create a file *base.ldif* with the following content:

```
dn: dc=ipb,dc=pt
objectClass: dcObject
objectClass: organization
o: Instituto Politecnico de Braganca
dc: ipb

dn: ou=users,dc=ipb,dc=pt
ou: users
objectClass: top
objectClass: organizationalUnit
description: Utilizadores
```

- Load the file *base.ldif* into openldap server:

```
   $ ldapadd -x -D "cn=root,dc=ipb,dc=pt" -W -f base.ldif
```

- Create a user creating a file *user.ldif* with the following content:

```
dn: uid=teste1,ou=users,dc=ipb,dc=pt
cn: Teste1
gidnumber: 100
homedirectory: /home/teste10
loginshell: /bin/false
mail: teste10@wifi.ipb.pt
mailmessagestore: /home/vmail/teste10/Maildir/
mailquotasize: 0
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: qmailUser
objectclass: posixAccount
objectclass: top
objectclass: shadowAccount
shadowexpire: -1
shadowlastchange: 12839
sn: Teste1
uid: teste1
uidnumber: 2001
userpassword: {MD5}qhv0ZG3mf9kIbPbHkAcCbA==
```

- Load the file *user.ldif* into openldap server:

```
   $ ldapadd -x -D "cn=root,dc=ipb,dc=pt" -W -f user.ldif
```

- Repeat last two steps for other nine users.

A.3 Qmail

To install Qmail and first need to install two applications that will give him some support, these applications are daemon-tools and ucspi-tcp and were developed by the same author of Qmail. To install the daemon-tools is necessary to perform the following steps:

```
$ wget http://cr.yp.to/daemontools/daemontools-0.76.tar.gz
$ sudo mkdir -p /package
$ sudo chmod 1755 /package
$ cd /package
$ sudo tar -zxvf /path_to_daemon_tools/daemontools-0.76.tar.gz
$ cd admin/daemontools-0.76
$ sudo package/install
```

To install ucspi-tcp is necessary to perform the following steps:

```
$ wget http://cr.yp.to/ucspi-tcp/ucspi-tcp-0.88.tar.gz
$ tar -zxvf ucspi-tcp-0.88.tar.gz
$ cd ucspi-tcp-0.88
$ make
$ sudo make setup check
```

Before installing Qmail is necessary create the directory */var/qmail*, create some local users editing the file */etc/passwd* with the command *vipw*, create two new groups by editing the file */etc/group*, adding the following lines in each file:

- */etc/passwd*

  ```
  alias:x:503:502::/var/qmail/alias:/bin/bash
  qmaild:x:504:502::/var/qmail:/bin/bash
  qmaill:x:505:502::/var/qmail:/bin/bash
  qmailp:x:506:502::/var/qmail:/bin/bash
  qmailq:x:507:501::/var/qmail:/bin/bash
  qmailr:x:508:501::/var/qmail:/bin/bash
  qmails:x:509:501::/var/qmail:/bin/bash
  ```

- */etc/group*

  ```
  qmail:x:501:root:
  nofiles:x:502:root:
  ```

Before compiling Qmail is necessary download the source of Qmail and Qmail-Ldap patch, and apply the patch:

```
$ wget ftp://ftp.ntnu.no/pub/unix/mail/qmail/qmail-1.03.tar.gz
$ wget http://www.nrg4u.com/qmail/qmail-ldap-1.03-20060201.patch.gz
$ tar -zxvf qmail-1.03.tar.gz
$ gunzip qmail-ldap-1.03-20060201.patch.gz
$ cd qmail-1.03
$ patch -p1 < ../qmail-ldap-1.03-20060201.patch
```

After the patch is applied is necessary edit the file *Makefile* and modify the following lines to the values presented below:

```
LDAPFLAGS=-DQLDAP_CLUSTER -DEXTERNAL_TODO -DDASH_EXT \
-DSMTPEXECCHECK -DALTQUEUE -DIGNOREVERISIGN -DQUOTATRASH

LDAPLIBS=-L/usr/local/lib -lldap -llber

LDAPINCLUDES=-I/usr/local/include

MDIRMAKE=-DAUTOMAILDIRMAKE

HDIRMAKE=-DAUTOHOMEDIRMAKE

SHADOWLIBS=-lcrypt
```

To compile and install Qmail, just run the following command:

```
$ sudo make setup check
```

To configure Qmail is necessary to perform the following steps:

- Add to the control files with the following values:

  ```
  $ sudo echo mail.wifi.ipb.pt > /var/qmail/control/me
  $ sudo echo -e "mail.wifi.ipb.pt\n wifi.ipb.pt\n localhost" >
  /var/qmail/control/locals
  $ sudo echo -e "mail.wifi.ipb.pt\n wifi.ipb.pt\n localhost" >
  /var/qmail/control/rcphosts
  $ sudo echo ou=users,dc=ipb,dc=pt > /var/qmail/control/ldapbasedn
  $ sudo echo cn=root,dc=ipb,dc=pt > /var/qmail/control/ldaplogin
  $ sudo echo password > /var/qmail/control/ldappassword
  $ sudo chmod 600 /var/qmail/control/ldappassword
  $ sudo echo localhost > /var/qmail/control/ldapserver
  $ sudo echo 1001 > /var/qmail/control/ldapuid
  $ sudo echo 1001 > /var/qmail/control/ldapgid
  $ sudo echo pt > /var/qmail/control/plusdomain
  ```

- Create the *supervise* and *log* directories and change its permissions:

  ```
  $ sudo cd /var/qmail
  $ sudo mkdir -p supervise/qmail-pop3d/log
  $ sudo mkdir -p supervise/qmail-send/log
  $ sudo mkdir -p supervise/qmail-smtpd/log
  $ sudo cd /var/log
  $ sudo mkdir -p qmail/pop3d
  $ sudo mkdir qmail/send
  $ sudo mkdir qmail/smtpd
  $ sudo chown qmaill.qmail -R qmail/
  ```

- Create the file */var/qmail/supervise/qmail-pop3d/run* with the following content:

  ```
  #!/bin/sh
  QMAILQUEUE="/var/qmail/bin/qmail-queue" export QMAILQUEUE
  ```

```
    exec /usr/local/bin/tcpserver -v -R -H -l 0 0 110 \
     /var/qmail/bin/qmail-popup mail.wifi.ipb.pt \
     /var/qmail/bin/auth_pop /var/qmail/bin/qmail-pop3d ./ 2>&1
```

- Create the file */var/qmail/supervise/qmail-pop3d/log/run* with the following content:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog t \
            s1500000 n100 /var/log/qmail/pop3d
```

- Create the file */var/qmail/supervise/qmail-send/run* with the following content:

```
#!/bin/sh
QMAILQUEUE="/var/qmail/bin/qmail-queue" export QMAILQUEUE
exec /var/qmail/rc
```

- Create the file */var/qmail/supervise/qmail-send/log/run* with the following content:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill \
/usr/local/bin/multilog t s1500000 n400 /var/log/qmail/send
```

- Create the file */var/qmail/supervise/qmail-smtpd/run* with the following content:

```
#!/bin/sh
QMAILDUID=`id -u qmaild`
NOFILESGID=`id -g qmaild`
LOCAL=`head -1 /var/qmail/control/me`
QMAILQUEUE="/var/qmail/bin/qmail-queue" export QMAILQUEUE
if [ -z "$QMAILDUID" -o -z "$NOFILESGID" -o -z "$LOCAL" ]; then
     echo QMAILDUID, NOFILESGID, or LOCAL is unset in
     echo /var/qmail/supervise/qmail-smtpd/run
    exit 1
fi
if [ ! -f /var/qmail/control/rcpthosts ]; then
     echo "No /var/qmail/control/rcpthosts!"
     echo "Refusing to start SMTP listener because it'll create an open relay"
    exit 1
fi
exec /usr/local/bin/tcpserver -v -R -H -P -l "$LOCAL" \
-x /etc/tcp.smtplocal.cdb -c 100  -u "$QMAILDUID" -g "$NOFILESGID" 0 smtp \
/var/qmail/bin/qmail-smtpd mail.wifi.ipb.pt /var/qmail/bin/auth_smtp /bin/true 2>&1
```

- Create the file */var/qmail/supervise/qmail-smtpd/log/run* with the following content:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog \
t s1500000 n100 /var/log/qmail/smtpd
```

- Create the file */var/qmail/rc* with the following content:

```
#!/bin/sh
# Using splogger to send the log through syslog.
# Using qmail-local to deliver messages to ~/Mailbox by default.
exec env - PATH="/var/qmail/bin:$PATH" \
qmail-start ./
```

- Create the file */var/qmail/bin/qmailctl* with the following content:

```
#!/bin/sh
PATH=/var/qmail/bin:/bin:/usr/bin:/usr/local/bin:/usr/local/sbin
export PATH

QMAILDUID=`id -u qmaild`
NOFILESGID=`id -g qmaild`

case "$1" in
  start)
    echo "Starting qmail"
    if svok /service/qmail-send ; then
      svc -u /service/qmail-send /service/qmail-send/log
    else
      echo "qmail-send supervise not running"
    fi
    if svok /service/qmail-smtpd ; then
      svc -u /service/qmail-smtpd /service/qmail-smtpd/log
    else
      echo "qmail-smtpd supervise not running"
    fi
    if svok /service/qmail-pop3d ; then
       svc -u /service/qmail-pop3d /service/qmail-pop3d/log
    else
      echo qmail-pop3sd supervise not running
    fi
    if [ -d /var/lock/subsys ]; then
      touch /var/lock/subsys/qmail
    fi
    ;;
  stop)
    echo "Stopping qmail..."
    echo "  qmail-smtpd"
    svc -d /service/qmail-smtpd /service/qmail-smtpd/log
    echo "  qmail-send"
    svc -d /service/qmail-send /service/qmail-send/log
    echo "  qmail-pop3d"
    svc -d /service/qmail-pop3d /service/qmail-pop3d/log
    if [ -f /var/lock/subsys/qmail ]; then
      rm /var/lock/subsys/qmail
    fi
    ;;
```

```
stat)
  svstat /service/qmail-send
  svstat /service/qmail-send/log
  svstat /service/qmail-smtpd
  svstat /service/qmail-smtpd/log
  svstat /service/qmail-pop3d
  svstat /service/qmail-pop3d/log
  qmail-qstat
  ;;
doqueue|alrm|flush)
  echo "Flushing timeout table and sending ALRM signal to qmail-send."
  /var/qmail/bin/qmail-tcpok
  svc -a /service/qmail-send
  ;;
queue)
  qmail-qstat
  qmail-qread
  ;;
reload|hup)
  echo "Sending HUP signal to qmail-send."
  svc -h /service/qmail-send
  ;;
pause)
  echo "Pausing qmail-send"
  svc -p /service/qmail-send
  echo "Pausing qmail-smtpd"
  svc -p /service/qmail-smtpd
  echo "Pausing qmail-pop3d"
  svc -p /service/qmail-pop3d
  ;;
cont)
  echo "Continuing qmail-send"
  svc -c /service/qmail-send
  echo "Continuing qmail-smtpd"
  svc -c /service/qmail-smtpd
  echo "Continuing qmail-pop3d"
  svc -c /service/qmail-pop3d
  ;;
restart)
  echo "Restarting qmail:"
  echo "* Stopping qmail-smtpd."
  svc -d /service/qmail-smtpd /service/qmail-smtpd/log
  echo "* Sending qmail-send SIGTERM and restarting."
  svc -t /service/qmail-send /service/qmail-send/log
  echo "* Restarting qmail-pop3d."
  svc -t /service/qmail-pop3d /service/qmail-pop3d/log
  ;;
cdb)
  /usr/local/bin/tcprules /etc/tcp.smtplocal.cdb \
  /etc/tcp.smtplocal.tmp < /etc/tcp.smtplocal \
  chmod 644 /etc/tcp.smtplocal.cdb echo "Reloaded /etc/tcp.smtplocal."
  ;;
  help)
    cat <<HELP
   stop -- stops mail service (smtp connections refused, nothing goes out)
  start -- starts mail service (smtp connection accepted, mail can go out)
  pause -- temporarily stops mail service (connections accepted, nothing leaves)
   cont -- continues paused mail service
   stat -- displays status of mail service
    cdb -- rebuild the tcpserver cdb file for smtp
restart -- stops and restarts smtp, sends qmail-send a TERM & restarts it
doqueue -- schedules queued messages for immediate delivery
 reload -- sends qmail-send HUP, rereading locals and virtualdomains
  queue -- shows status of queue
   alrm -- same as doqueue
  flush -- same as doqueue
    hup -- same as reload
HELP
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|doqueue|flush|reload|stat|pause
     |cont|cdb|queue|help}"
    exit 1
    ;;
esac

exit 0
```

- Create the following symlinks:

```
    $ sudo cd /usr/local/sbin
    $ sudo ln -s /var/qmail/bin/qmailctl
    $ sudo cd /service
    $ sudo ln -s /var/qmail/supervise/qmail-pop3d
    $ sudo ln -s /var/qmail/supervise/qmail-send
    $ sudo ln -s /var/qmail/supervise/qmail-smtp
```

A.4 Postfix

To install Postfix with LDAP support just run the following command:

```
$ sudo apt-get install postfix postfix-ldap
```

To configure Postfix is necessary to perform the following steps:

- Create the file */etc/postfix/accountsmap.cf* with the following content:

```
server_host = localhost
search_base = ou=users,dc=ipb,dc=pt
query_filter = (&(objectClass=qmailUser)(|(mail=%s)(mailAlternateAddress=%s)))
result_attribute = mailMessageStore
bind = yes
bind_dn = cn=root,dc=ipb,dc=pt
bind_pw = teste
```

- Create the file */etc/postfix/ldap-aliases.cf* with the following content:

```
server_host = 127.0.0.1
search_base = ou=users,dc=ipb,dc=pt
query_filter = (&(objectClass=*)(uid=%u))
result_attribute = mail
bind = yes
bind_dn = cn=root,dc=ipb,dc=pt
bind_pw = teste
```

- Edit the file */etc/postfix/main.cf* and modify the following lines:

```
myhostname = mail.wifi.ipb.pt
mydestination = wifi.ipb.pt, mail.wifi.ipb.pt, localhost.wifi.ipb.pt, localhost
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 193.137.106.0/25
local_transport = virtual
local_recipient_maps = $virtual_mailbox_maps
virtual_mailbox_base = /
virtual_mailbox_maps = ldap:/etc/postfix/accountsmap.cf
virtual_minimum_uid = 100
virtual_uid_maps = static:1001
virtual_gid_maps = static:1001
```

A.5 Courier

To install Courier with LDAP authentication support, just run the following command:

```
$ sudo apt-get install courier-authdaemon courier-authlib-ldap \
courier-imap courier-pop
```

To configure Courier is necessary to perform the following steps:

- Edit the file */etc/courier/authdaemonrc* and modify the following line:

```
authmodulelist="authldap"
```

- Edit the file */etc/courier/authldaprc* and modify the following lines:

```
LDAP_URI                ldap://localhost
LDAP_PROTOCOL_VERSION   3
LDAP_BASEDN             ou=users, dc=ipb, dc=pt
LDAP_AUTHBIND           1
LDAP_MAIL               uid
LDAP_GLOB_UID           vmail
LDAP_GLOB_GID           vmail
LDAP_HOMEDIR            mailMessageStore
LDAP_MAILDIR            mailMessageStore
LDAP_CRYPTPW            userPassword
```

- Edit the file */etc/courier/imapd* and modify the following line:

  ```
  MAXPERIP=50
  ```

- Edit the file */etc/courier/pop3d* and modify the following line:

  ```
  MAXPERIP=50
  ```

A.6 Dovecot

To compile and install Dovecot with LDAP authentication support, just run the following commands:

```
$ wget http://www.dovecot.org/releases/2.0/dovecot-2.0.1.tar.gz
$ tar -zxvf dovecot-2.0.1.tar.gz
$ cd dovecot-2.0.1
$ ./configure --with-ldap=yes --without-vpopmail --without-mysql \
--without-pgsql --with-storages=maildir --without-nss --without-shadow \
--without-pam --without-sqlite
$ make
$ sudo make install
$ cp -vrf doc/example-config/conf.d/ /usr/local/etc/dovecot
$ cp -vrf doc/example-config/dovecot.conf /usr/local/etc/dovecot
$ cp -vrf doc/example-config/dovecot-ldap.conf.ext /usr/local/etc/dovecot
```

To configure Courier is necessary to perform the following steps:

- Edit the file */usr/local/etc/dovecot/dovecot.conf* and modify the following line:

  ```
  protocols = imap pop3
  ```

- Edit the file */usr/local/etc/dovecot/dovecot-ldap.conf.ext* and modify the following lines:

  ```
  hosts = localhost
  auth_bind = yes
  ldap_version = 3
  base = ou=users,dc=ipb,dc=pt
  scope = subtree
  user_attrs = mailMessageStore=home,qmailUID=vmail,qmailGID=vmail
  user_filter = (&(objectClass=qmailUser)(uid=%u))
  pass_attrs = uid=user,userPassword=password
  ```

- Edit the file */usr/local/etc/dovecot/conf.d/10-auth.conf* and modify the following line:

  ```
  !include auth-ldap.conf.ext
  ```

- Edit the file */usr/local/etc/dovecot/conf.d/10-mail.conf* and modify the following line:

```
mail_location = maildir:./:LAYOUT=fs
mail_uid = vmail
mail_gid = vmail
```

- Edit the file */usr/local/etc/dovecot/conf.d/10-master.conf* and comment the following lines:

```
  ##inet_listener imaps {
    #port = 993
    #ssl = yes
  ##}
  ##inet_listener pop3s {
    #port = 995
    #ssl = yes
  ##}
```

- Edit the file */usr/local/etc/dovecot/conf.d/20-imap.conf* and modify the following line:

```
mail_max_userip_connections = 50
```

- Edit the file */usr/local/etc/dovecot/conf.d/20-pop3.conf* and modify the following line:

```
mail_max_userip_connections = 50
```

- Edit the file */usr/local/etc/dovecot/conf.d/auth-ldap.conf.ext* and modify the following lines:

```
userdb {
  driver = ldap
  args = /usr/local/etc/dovecot/dovecot-ldap.conf.ext
}
```

A.7 LVS-DR

To install the LVS-DR is necessary install in load balancer the following packages:

```
$ sudo apt-get install ipvsadm ldirectord nagios-plugins
```

To configure the LVS-DR load balancer is required to perform the following steps:

- Create the file */etc/ha.d/ldirectord.cf* with the following content:

```
autoreload=no
quiescent=no

virtual=193.137.106.100:25
        real=193.137.106.8:25 gate
        real=193.137.106.5:25 gate
        real=193.137.106.6:25 gate
        service=smtp
        scheduler=rr
        protocol=tcp
        checktype=external
        checkcommand="/usr/local/sbin/check-smtpd"

virtual=193.137.106.100:110
        real=193.137.106.5:110 gate
        real=193.137.106.6:110 gate
        real=193.137.106.8:110 gate
        scheduler=rr
        protocol=tcp
        checktype=external
        checkcommand="/usr/local/sbin/check-pop3"

virtual=193.137.106.100:143
        real=193.137.106.5:143 gate
        real=193.137.106.6:143 gate
        real=193.137.106.8:143 gate
        scheduler=rr
        protocol=tcp
        checktype=external
        checkcommand="/usr/local/sbin/check-imap4"
```

- Create the file */usr/local/sbin/check-smtpd* with the following content:

```
#!/bin/bash

args=("$@")
/usr/lib/nagios/plugins/check_smtp -H ${args[2]} -p ${args[3]}
```

- Create the file */usr/local/sbin/check-pop3* with the following content:

```
#!/bin/bash

args=("$@")
/usr/lib/nagios/plugins/check_pop -H ${args[2]} -p ${args[3]}
```

- Create the file */usr/local/sbin/check-imap4* with the following content:

```
#!/bin/bash

args=("$@")
/usr/lib/nagios/plugins/check_imap ${args[2]}
```

- Add the VIP address to network configuration in file */etc/network/interfaces*:

```
auto eth0:0
iface eth0:0 inet static
```

```
address 193.137.106.100
netmask 255.255.255.128
broadcast 193.137.106.127
```

- Disable the IP forwarding address:

  ```
  $ echo 0 > /proc/sys/net/ipv4/conf/eth0/forwarding
  ```

In real servers is necessary to configure the VIP address and disable ARP, adding the following lines to the file */etc.rc.local*:

```
echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/eth0/arp_announce
echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
ifconfig lo:0 193.137.106.100 netmask 255.255.255.255 broadcast 193.137.106.100 up
```

A.8 Eucalyptus

To install the frontend components of eucalyptus (CLC,CC,SC, Walrus) is necessary to run the following commands:

```
$ sudo apt-get install eucalyptus-cloud eucalyptus-cc eucalyptus-walrus \
eucalyptus-sc
```

During installation is necessary to answer the debconf questions:

- Configure postfix for internet delivery
- Name the cluster: cluster1
- Add a list of available IP addresses on test network: 193.137.106.60 - 193.137.106.70

To install the node components of eucalyptus (NC), is necessary to run the following command in all nodes:

```
$ sudo apt-get install eucalyptus-nc
```

After instalation is necessary to change the network configuration, the NIC will be in bridge mode. To do this simply run the following script:

```
interface=eth0
bridge=br0
sudo sed -i "s/^iface $interface inet \(.*\)$/iface $interface inetmanual\n\n
  auto br0\niface $bridge inet \1/" /etc/network/interfaces
sudo tee -a /etc/network/interfaces <<EOF
bridge_ports $interface bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
EOF
sudo /etc/init.d/networking restart
```

After is necessary to configure the file */etc/eucalyptus/eucalyptus.conf* with the name of the bridge, and restart the NC:

```
$ sudo sed -i "s/^VNET_BRIDGE=.*$/VNET_BRIDGE=$bridge/"
 /etc/eucalyptus/eucalyptus.conf
$ sudo /etc/init.d/eucalyptus-nc restart
```

After, is necessary to install the CLC's eucalyptus user's public ssh key into the NC's *eucalyptus* user's *authorized_keys* file. The easiest way to do this is:

- On the NCs, temporarily set a password for the *eucalyptus* user:

  ```
  $ sudo passwd eucalyptus
  ```

- Then, on the CLC:

  ```
  $ sudo -u eucalyptus ssh-copy-id -i ~eucalyptus/.ssh/id_rsa.pub
  eucalyptus@<IP_OF_NODE>
  ```

- Remove the password of the *eucalyptus* account on the Nodes:

  ```
  $ sudo passwd -d eucalyptus
  ```

Finaly is necessary to register the Walrus, CC, SC, and the NCs. To do this just starting the publication services:

```
$ sudo start eucalyptus-walrus-publication
$ sudo start eucalyptus-cc-publication
$ sudo start eucalyptus-sc-publication
$ sudo start eucalyptus-nc-publication
$ sudo start uec-component-listener
```

For users interact with the cloud is required to obtain credentials. To obtain the credentials from the command line and for that is necessary to perform the following steps:

```
$ mkdir -p ~/.euca
$ chmod 700 ~/.euca
$ cd ~/.euca
$ sudo euca_conf --get-credentials mycreds.zip
$ unzip mycreds.zip
$ ln -s ~/.euca/eucarc ~/.eucarc
$ cd -
```

To validate that everything is working correctly, is just necessary to get the local cluster availability details:

```
. ~/.euca/eucarc
euca-describe-availability-zones verbose
```

A.9 Management of VM images on Eucalyptus

After install eucalyptus environment is necessary to create the VM images, integrate them with Eucalyptus, registering with eucalyptus and finally running a VM image on eucalyptus.

A.9.1 Create Image

To create a VM image is necessary to perform the following steps:

- Create a new disk Image:

  ```
  $ kvm-img create -f qcow2 image.img 20G
  ```

- Install OS:

  ```
  $ sudo kvm -m 256 -cdrom ubuntu-10.04-server-amd64.iso \
   drive file=image.img,if=scsi,index=0 -boot d -net nic -net user \
  -nographic -vnc :0
  ```

- Access by vnc client to follow the installation:

  ```
  $ vncviewer A.B.C.D:0
  ```

  A.B.C.D is the IP address of the computer where is being create the VM image.

- After finishing the installation, is necessary to relaunch the VM by executing the following command:

  ```
  $ kvm -m 256 -drive file=image.img,if=scsi,index=0,boot=on -boot c -net nic \
   -net user -nographic -vnc :0
  ```

  At this point is necessary add all the packages required for VM.

A.9.2 Integrate image with eucalyptus

An instance running under Eucalyptus needs to know what IP it has and also, it needs to have the public key of the user allowed to do a passwordless access through SSH. This is accomplished by using a restful interface provided by the cloud. The interface is available under http://169.254.169.254/latest/meta-data and is accessible from within the instance. Eucalyptus node controller is set up to prevent automatic key injection if the system is in *MANAGED-NOVLAN* mode. Instead, it is assumed that the instance will use the above metadata service to retrieve the public keys when running in this mode. Is necessary to facilitate this by installing *curl* package and adding a script that will run on each boot. To install the *curl*:

```
$ sudo apt-get install curl
```

To add the script that will run on each boot, edit the file */etc/rc.local* and add the following lines before the *exit 0*:

```
depmod -a
modprobe acpiphp
mkdir -p /home/user/.ssh
echo >> /home/user/.ssh/authorized_keys
curl -m 10 -s
http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key |
```

```
grep 'ssh-rsa' >> \
                          /home/user/.ssh/authorized_keys
echo "AUTHORIZED_KEYS:"
echo "************************"
cat /home/user/.ssh/authorized_keys
echo "************************"
```

Before finish the VM installation is necessary to remove the network persistent rules from */etc/udev/rules.d*, so that the instance always comes up with *eth0* as the interface name as expected by eucalyptus:

```
$ sudo rm -rf /etc/udev/rules.d/70-persistent-net.rules
```

A.9.3 Register image with eucalyptus

For upload to eucalyptus in addition to the image file, other two files are required: vmlinuz-2.6.32-22-server, initrd.img-2.6.32-22-server. To upload this files and register the images is necessary to perform the following steps:

- Register the kernel image

  ```
  $ euca-bundle-image -i vmlinuz-2.6.32-22-server --kernel true
   $ euca-upload-bundle -b mybucket -m \
   /tmp/vmlinuz-2.6.32-22-server.manifest.xml
   $ euca-register mybucket/vmlinuz-2.6.32-22-server.manifest.xml
  ```

  Is necessary to ave the output produced by the last command above (eki-XXXXXXXX), which will be needed while registering the disk image.

- Register the ramdisk image

  ```
  $ euca-bundle-image -i initrd.img-2.6.32-22-server
  $ euca-upload-bundle -b mybucket -m \
  /tmp/initrd.img-2.6.32-22-server.manifest.xml
  $ euca-register mybucket/initrd.img-2.6.32-22-server.manifest.xml
  ```

  Is necessary to ave the output produced by the last command above (eri-XXXXXXXX), which will be needed while registering the disk image.

- Register the disk image

  ```
     $ euca-bundle-image -i image.img --kernel eki-XXXXXXXX --ramdisk eri-XXXXXXXX
     $ euca-upload-bundle -b mybucket -m \
  /tmp/image.img.manifest.xml
     $ euca-register mybucket/image.img.manifest.xml
  ```

A.9.4 Running a VM image on eucalyptus

To running a VM image on eucalyptus is necessary to perform the following steps:

- Create a keypair (Secure Shell (SSH) key) necessary to use to log into instance as root, once it boots.

```
if [ ! -e ~/.euca/mykey.priv ]; then
    mkdir -p -m 700 ~/.euca
    touch ~/.euca/mykey.priv
    chmod 0600 ~/.euca/mykey.priv
    euca-add-keypair mykey > ~/.euca/mykey.priv
fi
```

- Allow access to port 22 for instances:

  ```
  $ euca-authorize default -P tcp -p 22 -s 0.0.0.0/0
  ```

- Create a instance of image created:

  ```
  $ euca-run-instances $EMI -k mykey -t c1.xlarge
  ```

- Monitor the state of instance:

  ```
  $ watch -n5 euca-describe-instances
  ```

- When the instance is fully started, the above state will become 'running'. To connect to an instance is necessary, using the IP address assigned to instance in the output, with the following command:

  ```
  $ IPADDR=$(euca-describe-instances | grep $EMI | grep running | tail -n1 |
  awk '{print $4}')
  $ ssh -i ~/.euca/mykey.priv ubuntu@$IPADDR
  ```

- To terminate a intance is necessary run the following command:

  ```
  $ INSTANCEID=$(euca-describe-instances | grep $EMI | grep running | tail -n1
  | awk '{print $2}')
  $ euca-terminate-instances $INSTANCEID
  ```

A.10 HAproxy

To install and configure de HAproxy is necessary to perform the following steps:

- Install HAproxy from the available packages:

  ```
  $ sudp apt-get install haproxy
  ```

- Edit and modify the file */etc/haproxy*:

```
global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        maxconn 32000
        ulimit-n 65536
        user haproxy
        group haproxy
        daemon
        nbproc  4
        noepoll
        nopoll

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        retries 3
        option redispatch
        maxconn 2000
        contimeout      5000
        clitimeout      50000
        srvtimeout      50000

listen smtp 172.19.1.4:25
        mode tcp
        option tcplog
        balance roundrobin
        server smtp 193.137.106.60:25 check inter 50000
        server smtp1 193.137.106.61:25 check inter 50000
```