University of Amsterdam

System & Network Engineering

Performance optimisation of webmail

July 4, 2013

Authors:
Periklis Stefopoulos perilkis.stefopoulos@os3.nl
Katerina Mparmpopoulou katerina.mparmpopoulou@os3.nl

Supervisor:
Michiel Leenaars

Abstract

E-mail is the dominant communication mean over Internet, which generates a significant amount of network traffic. Over the world hundreds of millions of end users depend on Webmail technologies. This paper presents a detailed analysis of the effectiveness of four different open source Webmail services: Squirrelmail, Roundcube, Horde and Afterlogic. All of these services are combined with three different IMAP backends in order for the most efficient solution among these to be found. We mainly focus on the most common actions that users perform, such as searching through their Webmails for a certain message or displaying the entire inbox folders. We make our evaluation in two different dimensions: from a user experience perspective and from a system performance perspective.

Contents

1 Introduction **1**
1.1 Related Work . . . 2
1.2 E-mail System Components . . . 2
1.3 E-mail System Protocols . . . 4

2 Approach **6**
2.1 Metrics . . . 6
2.2 Data gathering . . . 7

3 Experiments **9**
3.1 Experimental Environment . . . 9
3.2 Email Backends . . . 9
3.2.1 Postfix . . . 9
3.2.2 Security Modules . . . 10
3.2.3 Dovecot . . . 10
3.2.4 Courier . . . 12
3.2.5 Cyrus . . . 12
3.3 Webmail Frontends . . . 13
3.3.1 Squirrelmail . . . 13
3.3.2 Roundcube . . . 13
3.3.3 Afterlogic Webmail Lite . . . 13
3.3.4 Horde IMP . . . 15
3.4 Experiments . . . 15

4 Results **17**
4.1 Latency . . . 17
4.1.1 Fetching messages . . . 17
4.1.2 Searching from the "Subject" . . . 18
4.1.3 Searching from the entire message . . . 21
4.2 CPU time & Memory consumption . . . 25

5 Conclusions **30**

6 Future Work **31**

List of Figures **32**

Bibliography **33**

1 Introduction

The moment in 1971 when the first message was sent between two computers via the ARPANET network, marked an important step in the evolution of human communication. Over the years, developments in computer and Internet technology have rendered e-mail an essential tool which soon became the dominant communication mean over Internet. Although e-mail is still one of the most used Internet services, which generates a significant amount of network traffic, it has not evolved much and has stayed pretty constant.

At present the existing e-mail systems enhance the basic e-mail functionality, that of sending and receiving quick messages, providing more features. DomainKeys Identified Mail (DKIM), Pretty Good Privacy (PGP), Secure/Multipurpose Internet Mail Extensions (S/MIME), and various antispam and antivirus filters are some of them. Webmail is one of the most common services that e-mail providers provide to their customers. Webmail systems gather all the e-mail characteristics while large software companies focus on their continuous development in order to meet the new demands of the users. On the other hand, open source webmail system solutions are often a neglected area of development.

An open source webmail solution has essential advantages over a proprietary one, such as customizability, security, interoperability with multiple backends, and certainly the zero cost. Our primary motivation is to understand the key factors that limit the performance of these webmail systems and subsequently help their further development. Thus, the following research question is defined:

What are the bottlenecks, in terms of performance, of current Webmail implementations and which could be the most optimal solution?

A webmail system consists of a webmail application as the frontend and a classical e-mail service as a backend. In this research the focal point is to profile the performance of accessing the mailbox of different types of users from the web interface of an integrated webmail system. We characterize and analyze the results of all experiments that have been carried out, using three metrics: latency, Central Processor Unit (CPU) time and memory usage.

The structure of this paper is as follows: At first, the e-mail system components and the major e-mail protocols are described, followed by the Approach (Chapter 2) where the used metrics and the way the data was gathered are presented. In Chapter 3 the Experimental Environment is defined along with a short description of the webmail frontends and backends that were used, concluding with the definition of the conducted experiments. Afterwards in Chapter 4 the Results are presented. The Conclusions, in Chapter 5, summerize our findings and provide the answers to the research question while the Future Work (Chapter 6) gathers our ideas concerning all the further research work that

needs to be done.

1.1 Related Work

There is a limited number of research papers published on webmail servers/applications. In a recent work [1], the authors compare the performance between mail and webmail services over Cellular networks and also identify the different factors, related to the network configuration, which affect it. Throughput, which has defined as the amount of bytes transferred at the TCP layer divided by the total duration between the first and the last packet of the connection, is used as the main performance metric. They show that performance is significantly degraded in the case of the mail service compared to the webmail.

Wang and Hu [2] present a performance analysis on an Internet Service Provider (ISP) mail server. Modifying and using the SPECmail2001 as a benchmark, they evaluate the performance of the mail server based on realistic network connections, disk storage and client workloads. Several important results arrived through that research, such as the considerable impact of I/O latencies in the data transfer time of the e-mail requests.

The most relevant research to us has been done from Patrick Appiah-Kubi et al. [3]. They describe the architecture and the design of a Webmail server that runs on the bare hardware without an Operating System acting as the middleware. After conducting experiments, simulating common Webmail transactions, they conclude that their implementation gives better performance in terms of processing time.

1.2 E-mail System Components

An ordinary email system can been seen as a collection of distinct and possibly geographically separated components. Each of these components is organized to accomplish a specific function or a set of functions in the process of sending or receiving email. For most users, the operations that they perform are transparent. There are five basic types of functionality in an email system which make feasible the store-and-forward services of an electronic message.

When the user wants to compose or read an electronic message he or she contacts the Mail User Agent (MUA), more commonly referred to as email client. In its original form, it is a software program which runs on the user's local computer. Mutt, Pine, Mozilla Thunderbird and Microsoft Outlook are some of the well known MUAs. Web based email clients, like Squirrelmail, have the advantage of not requiring any software install or maintenance. The delivery process starts with the Mail Submission Agent (MSA) accepting the outgoing message from the MUA. An MSA acts as a submission server, which determines whether the messages should be accepted. According to RFC 4409

[4], an MSA can also deliver the message or relay it to a Mail Transfer Agent (MTA). Most of the times, the MSA is either an integrated piece of a MUA, such as in the case of Pine, or of an MTA.

The MTA is the component that in fact transfers the message towards its proper destination using client-server application architecture. Unix Postfix, Sendmail, Exim and Microsoft Exchange Server are examples of MTAs. After receiving a message, the MTA applies a "Received" trace header to the message's header. Cumulatively the "Received" headers that the different MTAs append aim to provide information on the message origin and on the route that the message has taken before reaching its destination. Once the message's recipient is a local user, the MTA passes it to the Mail Delivery Agent (MDA) for the final delivery, otherwise the message is relayed, that is, forwarded to another remote MTA. Although sometimes the MDA is a separate program, in most cases it will be a part of MTAs. Maildrop and Procmail, for instance, are such agents which offer message delivery to the user's mailbox.

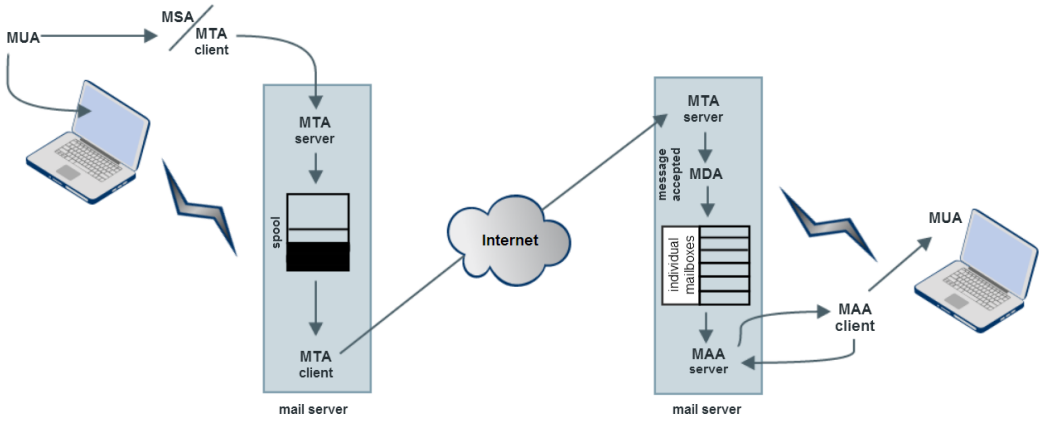


Figure 1: Major e-mail Components

The most common scenario today is that the users are connected through Internet to the mail server in order to receive their new emails. Usually MUAs do not have a direct access to the message store, the place where the MDA stores the incoming mails for the individual recipient. Here is where the Mail Access Agent (MAA) comes. Its responsibility is to interact with the message store on behalf of the MUA after the latter's explicit request [5]. Dovecot, Cyrus and Courier are included in this category of agents, among others.

Finally, in some situations users or sites use a program which acts as a Message Retrieval Agent (MRA) in order to send or retrieve messages. The MRA periodically

establishes a connection to the MAA to retrieve the messages from the server message store and stores them locally. Typical examples of MRAs are getmail and fetchmail.

1.3 E-mail System Protocols

Email, although it is simple to use, relies on several standards and protocols. These protocols work together and govern both the way the different email system components communicate to each other as well as the definition of message formats, in order for the message to be handled, transferred and delivered properly. The most important email system protocols are the Simple Mail Transfer Protocol (SMTP) for sending email messages and the Post Office Protocol (POP3) or the Internet Message Access Protocol (IMAPv4) for retrieving them.

SMTP, as defined in RFC document 2821 [6], is a standard transportation protocol which provides reliable and efficient email transfer. Basically, when an SMTP email client contacts an SMTP server, the latter through a set of commands authenticates the identity of the former (the credentials that the former obtains) and then accepts the message. Subsequently, the SMTP server looks up the recipient's mail server and directs the transfer of the mail accordingly. Obviously, a MUA can act as an SMTP client, while the role of the SMTP server can be assigned to an MSA or MTA. The other purpose of SMTP is to allow the relay of mail messages toward their final destinations.

According to RFC document 4409 [4] when email is submitted from MUA to a MSA it uses the Extended SMTP on TCP port 587 but many MTAs also allow submission on TCP port 25. The MTA-MTA communication is done using TCP port 25. In order to provide integrity, confidentiality and authenticity SMTPS can be used. SMTPS is not a new protocol or an extension to SMTP, it is only the way to secure SMTP on the transport layer. Client and Server use SMTP at the application layer and SSL/TLS at the transport layer.

The SMTP can be described as a connection-oriented, text-based protocol in which the SMTP client (MUA or MSA or MTA) initiates a session with the SMTP server (MSA, MTA) and within this session the SMTP transactions take place. This transaction starts with the `MAIL FROM: "<address>"` command which specifies to the receiver that is about to receive a new message and also the sender address. After a non erroneous response from the SMTP server the recipients of a message are defined using the command `RCPT TO:<address>`. Finally, the command DATA is used and if the server replies with `354 End data with <CR><LF>.<CR><LF>`, the actual message is sent as a succession of lines of text. The end of the message is denoted by a period on a line of its own.

Post Office Protocol (POP) is an e-mail retrieval protocol. It is also a client/server protocol as SMTP. The e-mail client establishes a TCP communication session with the e-mail server, on port 110, to retrieve the e-mail messages and stores them in the local

mailbox. After the control commands have been successfully transferred through the communication channel and the server has responded positively, the user can download the emails from the server to client, having the option to remove them from the server's message store. These messages are stored locally on the user's hard disk, saving storage space on the server. Since the user may have multiple systems on which he or she retrieves mail, this may lead to consistency problems in cases where the user wants to view the e-mail data from a different machine.

Version 3 of POP, which is the current standard, is simple and without many resource requirements. Thus, the configuration is transparent and not prone to errors. Unlike SMTP, POP3 uses authentication by default. Also Simple Authentication and Security Layer (SASL) extension is supported by POP3 since RFC 5034 [7]. POP3 can be deployed using TLS connection for providing encryption and integrity protection on the transport layer. There are two ways for this to be done. Either by establishing a TLS connection on port 995, in which POP3 transaction (POP3S) takes place, or by issuing the STARTTLS command in an existing unencrypted POP3 session to upgrade it to TLS. More details are discussed in RFC 2595 [8].

Internet Message Access Protocol (IMAP) is also used to retrieve e-mail messages from the server to the client. It is a connection-oriented, text-based protocol in which a MUA communicates with an MAA by issuing command strings and supplying necessary data over a reliable ordered data stream channel, typically a Transmission Control Protocol (TCP) connection over port 143. The standard defines that the messages are left to the server message store rather than downloaded locally. This implies a synchronization between the MUA's and the server's mail store. Webmail applications do not store the messages locally, this is the reason why they use IMAP instead of POP3.

RFC 3501 [9] defines the curent version, IMAP version 4 revision 1 (IMAP4rev1). Although IMAP4 is not as simple as POP3, it offers more features and flexibility to the user, such as providing mail subfolders on the server and concurrent access to the same mailbox from different clients. RFC 4959 [10] defines the IMAP extension for SASL authentication while RFC 2595 [8] defines the support for TLS encryption on the transport layer. Also two options are available for encryption, IMAPS on port 993 and the STARTTLS command for upgrading an unencrypted session to TLS.

2 Approach

2.1 Metrics

The purpose of our analysis is to evaluate the performance of webmail services. To this end, we characterize and analyze the results of all experiments that have been carried out, using three metrics: latency, Central Processor Unit (CPU) time and memory usage.

Latency is an important performance metric for a great deal of applications within the service provider industries and is closely related to the end-user experience. Identifying the sources of delay in webmail systems is of critical significance for their further optimization and fine tuning. Before presenting our results we need to specify the way we define latency. In this paper, the term latency deals with the delay between the client request and the first packet of the server response.

A webmail system consists of a backend and a frontend component. When the user makes a request, these two components exchange and process IMAP messages causing delays. More specifically, the following sources of delay exist:

- The backend processing time. It is the amount of time needed by the backend component to process the requests coming from the frontend.
- The frontend processing time. It is the amount of time needed by the frontend component to process the requests and the responses coming from the end-user and the backend component respectively.
- The latency incurred due to the communication between the backend and the frontend component.

Although the term latency[1] is not the most appropriate one, we make use of it in order to refer in the summation of these various types of delays.

CPU time is the total amount of time for which the CPU is dedicated to a process and does not include the time which has been spent waiting for I/O or running other programs [11]. According to [12], CPU time can be calculated by the equation:

$$CPU\ time = Number\ of\ CPU\ cycles\ to\ execute\ a\ program * Cycle\ time$$

In particular, in a system with more than one processor the CPU time of an application is the sum of the time that each processor spends for the execution of this specific application. We consider CPU time as a substantial metric in determining how CPU-intensive a program is. Moreover, it is directly associated with throughput; the less time a program uses the CPU, the longer the central processor is available, the more processes can be executed without waiting, thus maximizing the throughput.

[1] The term latency is used to describe the delay between a user request and the service response due to network delays because the processing of a user request during that time is latent (in a state of rest or inactivity).

Memory usage is another important figure of metric related to measuring performance. This is because when a software program needs to allocate memory, both for its code and for data, it takes time and binds resources. Hence, the overall performance of the system is affected.

In our research we measure the physical memory usage taking into account not only the amount of memory used exclusively by the process, but also the shared memory that the application has access to. The fact that large amount of memory pages are shared between multiple programs complicates the decent measurement of memory usage. Two metrics are taken which give us an overall view of the actual memory usage:

- The Unique Set Size (USS) which is the unshared memory that a process uses.
- The Proportional Set Size(PSS) which is the portion of the shared memory that the process has access plus the unshared memory (USS).

The Resident Set Size (RSS), although in many cases is used as a standard to describe the memory usage, is an overestimated metric because takes into account all the physical memory that is shared among several processes and not the specific portion. Consequently this metric is not used in this research paper.

For selecting the above measurement metrics, their characteristics, such as linearity, reliability, repeatability and consistency were considered. We believe that these metrics meet our requirements, allowing for accurate comparisons, thus leading to correct conclusions.

2.2 Data gathering

Data gathering is done through a selected set of tools:

sar [13] is a tool that is contained in the sysstat package. It provides the administrator with a set of reports about system's activity, such as CPU utilization, I/O transfer rates and memory and swap space utilization.

smem [14] reports physical memory usage, taking shared memory pages into account, offering a more accurate representation of the amount of memory used by each process.

tcpdump [15] is an application that prints out a description of the contents of packets on a network interface.

Whireshark [16] is a network protocol analyzer that lets you interactively browse packet data from a live network or from a previously saved capture file.

To measure the efficiency of different Webmail services and subsequently propose a complete integrated solution, the above tools were used to extract data for each of the predefined metrics. Concerning latency, the tcpdump command was used to capture

only the http traffic (port 80) and write the output to a file[2]. Afterwards, this captured file was analyzed with the help of Whireshark. The latency was calculated by subtracting the timestamp of the request from the timestamp of the response.

CPU time values were extracted using CPU utilization, reported from the sar in intervals of 1 second. This CPU utilization is expressed in percentage of time of the available CPU time. Given that in our experiments we use 4 processors, the available CPU time during 1 second of elapsed time is 4 seconds. This can be formulated:

$$CPU\ time = CPU\ utilization * available\ CPU\ time = CPU\ utilization * elapsed\ time * number\ of\ CPUs$$

Finally, for the memory utilization, the smem tool gave us all the required data[3]. More precisely, we limit its output setting the -P parameter followed by the target process. Table 1 summarizes our metrics and for each one of them the way in which their corresponding values were derived.

Metric	Extraction method
Latency	tcpdump/Wireshark
CPU time	systat/sar
Unique Set Size (USS)	smem
Proportional Set Size (PSS)	smem

Table 1: Benchmark metrics and their extraction method

[2]tcpdump port 80 -w file.pcap
[3]smem -P <process_name >

3 Experiments

The following chapter explains the experiments that were conducted. First the environment is described, in which these experiments took place. After that, each individual component is analyzed. Finally, in section 3.4 the experimental variables are explained along with the experimental procedure.

3.1 Experimental Environment

To identify and analyse the bottlenecks of different Webmail systems, a stable experimental environment was built. A Dell PowerEdge R210 II server is the actual physical machine which hosts the different IMAP backends and the Webmail applications. More specifically the installation of a hosted hypervisor make it feasible for a number of different Virtual Machines (VMs) to run on the server. The base Operating System (OS) running on this server is Ubuntu-12.04.2 with kernel version 3.5.0. The hypervisor is Xen version 4.2.1 [17]. In addition to Xen Hypervisor, the Bind DNS server was also installed on the base machine and configured with the proper MX records. In this way, the mail exchanger (MX) records for our email servers could be resolved to an IP address and subsequently receive e-mail messages.

On the top of Xen, three different email solution backends and four different Webmail servers were set up. In all our backends we chose Postfix version 2.10 [18] as the MTA service along with Amavisd 2.8.0 [19], Clam Antivirus 0.97.8 [20] and SpamAssasin [21] as antispam and antivirus security modules. Moreover, all the email systems make use of a MySQL database scheme (MySQL Community Server 5.6.12) for managing user accounts and e-mail forwarding due to the flexibility, security and efficiency it offers. What distinguishes these machines is the installed MAA server on it. Cyrus along with the cyrus-sasl authentication module, Courier and Dovecot are the different IMAP server that were selected for the three backend VMs.

Squirrelmail, Roundecube, Horde and Afterlogic are the Webmail applications that were set up in the remaining VMs. Their choice based on the fact that they are among the most well known open source web frontends to mail servers in the hosting industry.

In our experimental setup all the virtual machines that were laid on the Xen Hypervisor have the following hardware and OS configuration: 2 GB of memory, 20GB of disk using EXT3 file system and Ubuntu version 12.04 (Precise Pangolin).

3.2 Email Backends

3.2.1 Postfix

At present Postfix is the most widely used Unix mailer. Developed by Wietse Venema in 1998, Postfix as a complete MTA package was aimed to replace Sendmail. A great part of its success is due to its modularity. Modularity allows to disable some functions that

are not used and in this manner the execution of the functions that they are needed is quicker, making the overall performance better. However, its core functionality remains the same as any other's MTA. Postfix does not run exclusively in root mode, instead of a master program, referred to as the "master daemon" runs as the root and spawns off processes to handle the different functions. This implies a more advanced security level and even if a process is being compromised, the rest of the email functionality will remain intact.

Another implementation advantage is that the configuration of Postfix is pretty straightforward as it basically depends on two configuration files; the "main.cf" and the "master.cf". The first one contains parameters that are used by the Postfix programs while the "master.cf" contains the parameters that Postfix master program uses when runs the other core processes. In its key characteristics stability is also included, in terms that the Postfix server can continue to be up and running independently of a component failure.

Postfix is a feature-rich MTA software. It supports many protocols such as TLS encryption and authentication, IPv6, Multipurpose Internet Mail Extensions (MIME), multiple SASL authentication implementations (Cyrus, Dovecot), DomainKeys Identified Mail (DKIM), DomainKeys and SenderID authentication (via Milter plug-in) and PKI-less TLS server certificate verification based on DANE (DNS-Based Authentication of Named Entities). Additionally, it supports a large number of databases such as Memcache database, SQLite database, CDB database, PostgreSQL database, MySQL database, Berkeley DB database and DBM database.

3.2.2 Security Modules

There are many spam filtering tools and antivirus solutions available. Amavis, SpamAssassin and ClamAv are the most popular open-source programs of this category. Before delivering electronic messages on the local mailbox, Postfix hands them out to the Amavis content filter. Then Amavis processes and checks the content of the messages with the help of other antivirus and antispam modules, such as ClamAv and SpamAssassin. Both of them are capable of updating their antivirus definitions and spam rules without manual intervention. Figure 2 depicts the interaction of Postfix with Amavis, Clamav and Spamassasin.

3.2.3 Dovecot

Dovecot[22] is an open source IMAP and POP3 email server. Its most remarkable characteristics are the support of IPv6, SSL/TLS, mbox and maildir format, many different methods for obtaining credentials for authentication (e.g. LDAP, databases like PostgreSQL, MySQL, SQLite, Dict key-value databases, Passwd-file) and mechanisms (CRAM-MD5, DIGEST-MD5, GSSAPI). Its installation and configuration procedure is not laborious and it has a well-maintained and useful online documentation.

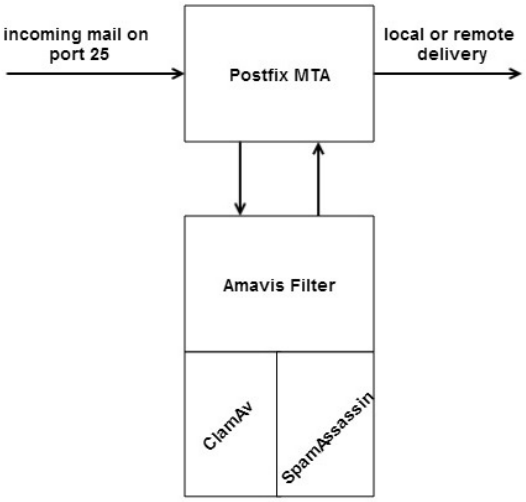


Figure 2: Postfix and security modules

Each individual mailbox contains two index files which are created automatically by Dovecot:

dovecot.index is the main index file and contains information for each stored message, like the UID, pointer to cache file, summary information on how many messages exist, how many of them are unseen and how many are marked with the "deleted" flag.

dovecot.index.cache is the cached mailbox data and contains information about message headers, sent & received date, message's parsed MIME structure and IMAP's BODYSTRUCTURE fields

Dovecot's functionality is based on the following four processes.

Master process (Dovecot) is the core process which main task is to keep all the other processes running and passes them the required settings from the configuration files. Also it is responsible for keeping logs of the IMAP/POP3 service.

Login processes (imap-login, pop3-login) run with the lowest privileges and provide the required IMAP and POP3 protocol function before the user logs in.

Authentication process (dovecot-auth) carries out the authentication process.

Mail processes (imap, pop3) handle the other IMAP/POP3 functions after the user successful log on.

3.2.4 Courier

Courier IMAP server is part of the Courier Mail server which can also be used with other MTAs such as Postfix or Exim, provided that maildir message format is used. It aggregates IMAP and POP3 services. The use of shared folders, the ability of IMAP over SSL and IMAP/POP3 proxying are some of its features. Courier offers the administrator the option to set a variety of restrictions, such as the maximum number of IMAP logins or the maximum number of logins from the same IP address.

Configuration files are in plain text format, specifically `imapd` and `imapd-ssl` files are used to configure the Courier IMAP daemon and `pop3d` & `pop3d-ssl` are used to configure Courier POP3 daemon. Finally the authentication is done by the Courier Authentication Library which has modules for different authentication options, like authcram (CRAM-MD5), authuserdb (DB database files), authpam (PAM), authpgsql (PostgreSQL), authldap (LDAP) and authmysql (MySQL).

3.2.5 Cyrus

The Cyrus IMAP server was developed at Carnegie Mellon University as part of the Cyrus [4] project at early of 1990. Cyrus uses an hierarchical structured mailbox which is created automatically when the user logins for the first time or manually using the `cyradm` administration tool. In addition to the basic folders, such as Inbox and Drafts, each Cyrus mailbox contains four binaries:

cyrus.index is the core of the mailbox format. It contains a header with information about the entire mailbox and one record per message. The header provides information regarding the number of messages stored in the mailbox, the unique identifier (UID) and the time of the message which added last in the mailbox. Each of the message records holds data like the UID of the message and its size and header.

cyrus.header metadata file contains the name of the root directory from which the quota is applied and also a copy of the access lists (ACLs) applied in the mailbox.

cyrus.cache is a binary file used only for performance efficiency. It mainly stores all the headers come from previous user's queries, creating a cache.

cyrus.seen contains information about the messages that have been seen already by the user.

Cyrus IMAP offers a great variety of features. It supports quotas, SIEVE for filtering messages, ACLs for enabling multiple users to access a certain mailbox, bulletin boards for allowing users to share folders without the administration intervention. Mail store partitioning is also provided, permitting users to distribute their mailbox in more

[4] It was named after Cyrus the Great of Persia (576-530 BC) who initiated one of the first well known postal systems.

than one different physical disks, enhancing scalability. Finally, it uses the Simple Authentication and Security Layer library (SASL) for authentication allowing CRAM-MD5, DIGEST-Message-Digest Algorithm 5 (MD5), OTP and PLAIN as authentication mechanisms.

Cyrus can be configured through `cyrus.conf` and `imap.conf`. The first controls the services that Cyrus supplies while the second one is the main configuration file for the IMAP daemon and controls the login mechanisms. Since our main goal is to evaluate the performance of different open source webmail solutions rather than the in-depth analysis of the IMAP technologies, we suggest to those who want to acquire more information regarding Cyrus and Courier to refer to [23].

3.3 Webmail Frontends

3.3.1 Squirrelmail

Squirrelmail is one of the best-known free webmail solutions, first released in 1999. It is a lightweight application, written in PHP, which allows the users to view and compose their mails through a web browser. At the time of writing the most recent stable version of Squirrelmail is version 1.4.22. It provides users with shared calendars and address books while there are also plugins for S/MIME, GPG/OpenPGP and automatic forwarding of incoming mail to a different address support. A more complete list with its features can be found in `http://squirrelmail.org/wiki/SquirrelMailFeatures`.

There are three ways to change the SquirrelMail configuration file: using the configuration tool `config/conf.pl`, using the Administrator plugin, or editing it manually (`config/config.php`).

3.3.2 Roundcube

Roundcube is a relatively new, open-source, web-based MUA which has been written in PHP. It offers full functionality, like shared/global IMAP folders, drag-&-drop message management, IMAP folder management, IMAP quotas and ACLs. It uses Asynchronous JavaScript and XML (AJAX) to provide a more fluid user interface. It can be easy installed and the configuration is administrator friendly. The only requirements are a Webserver with PHP and a database (MySQL, PostgreSQL, SQLite or MSSQL).

3.3.3 Afterlogic Webmail Lite

Aterlogic Webmail Lite is an open source webmail frontend. It supports multiple domains, address book, IMAP mail search, IMAP quotas and Sieve for mail filtering, auto-respond and forwarding. It gives the advantage of safely opening HTML messages due to blocking javascript and external images. It comes in two flavors, for PHP and ASP.NET platforms, and MySQL or MSSQL is required. It has a straightforward installation and configuration procedure through a web-based interface. For fine tuning

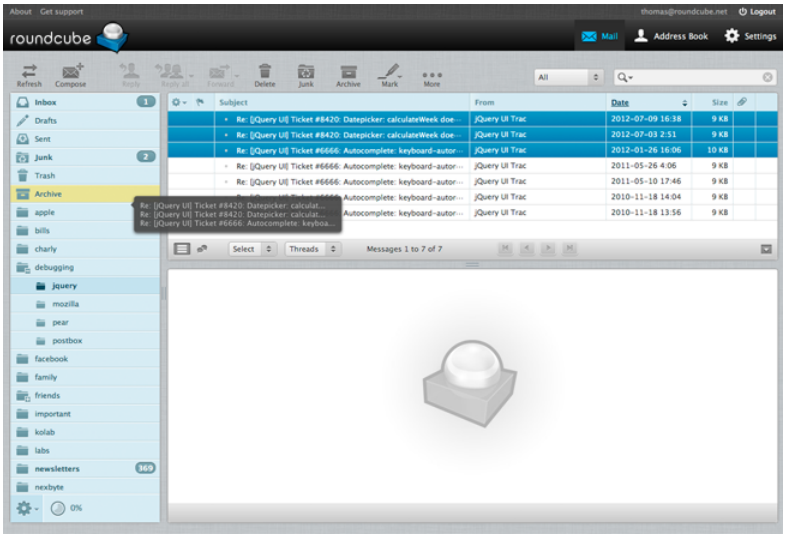


Figure 3: Roundcube user interface

the settings.xml file can be modified accordingly.

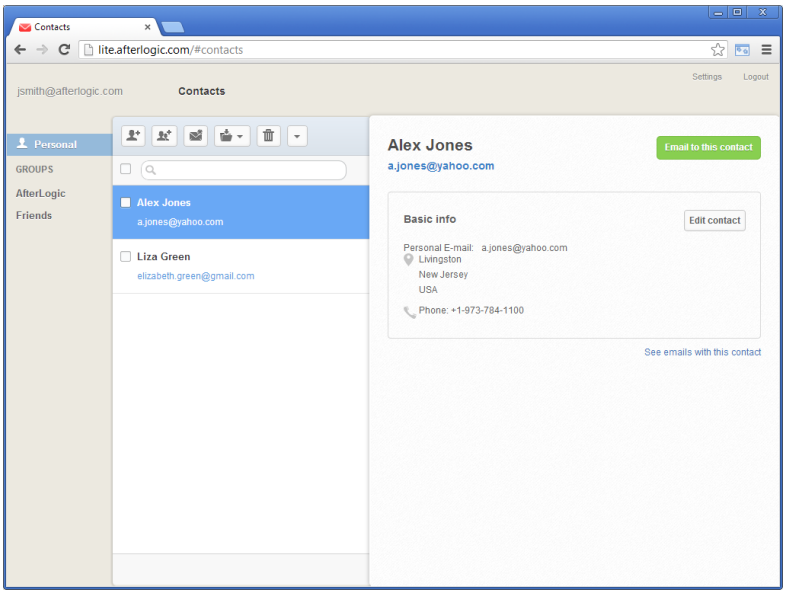


Figure 4: Afterlogic Webmail Lite user interface

Its strongest point is the use of Asynchronous JavaScript and XML (AJAX). AJAX scripts run on the client-side, enabling asynchronous communication with the server. Afterlogic uses AJAX to pre-fetch (without a user interaction) emails, headers and bodies, to the client cache in order to be available instantly.

3.3.4 Horde IMP

Horde IMP is a webmail, part of the Horde Project, available for free. Similarly to the other selected Webmails of this paper, it offers encryption and signing support of messages (S/MIME and PGP), IMAP quota, shared IMAP folders, message filtering and searching.

The configuration is pretty simple using only two files, the `backends.php` and the `conf.php`. The first file is to configure the IMAP backend properties and the conf.php for different webmail options like attachment size limit and reply number limit.

3.4 Experiments

To evaluate the performance of a Webmail system we conduct experiments, based on four different scenarios. Data about our metrics (analyzed in Section 2.1), which concern latency, CPU consumption and memory utilization were collected using the methods that have been described in Section 2.2.

The first of our scenarios involves the fetching of the first 15 headers of the mailbox's messages after the user logs in. The second type of experiments investigates the procedure in which the user requests the fetching of all the message headers through the webmail interface. Subsequently, two user actions define the last two experiments: searching the entire mailbox for mails with a keyword within a mail's header field, more specifically the "Subject" and searching the entire mailbox for mails with a keyword within the entire message content.

Users	# messages	Contents of messages	Size of the mail dataset
user 1	1500	text only	49.5 MB
user 2	3000	text only	99 MB
user 3	4500	text only	148.5 MB
user 4	6000	text only	188 MB
user 5	1500	text with attachments	2.3 GB

Table 2: The different types of users and the number of messages

All these experiments are performed in a consistent experimental environment, using all the IMAP servers - Webmail combinations, leading to 12 different Webmail solutions. For the above scenarios, 5 different types of users act as an additional experimental variable. Four of them have a mailbox with only text messages, each differentiated by the number of messages. The fifth one has a mailbox in which the mails, along with text, contain also attachments. Moreover, the number of the messages in the fifth user's mailbox is equal to the number of messages of one of the previous four users. In this way a directly comparison between a mailbox with and without attachments is feasible,

highlighting the impact of attachments in a mailbox. Table 2 demonstrates all the types of the users, the number of the messages in their mailbox, as well the size of each mail dataset.

It is worth mentioning that in order to achieve accurate results, all the experiments were repeated five times. This provided us with enough data, capable of returning an average, yet reliable result, and also preventing us from erroneous conclusions.

4 Results

In this chapter, the results of the previously described experiments are presented and further analyzed.

4.1 Latency

4.1.1 Fetching messages

The bar chart in Figure 5 shows the average latency for different combinations of IMAP backends and webmail frontends while the user logs in. We present the average latency because there was almost no difference in latencies concerning the different mailbox sizes. All the Webmails are configured to display the first 15 messages by default, as a result only the first 15 message headers are fetched from the IMAP server. We observe that although the conversations between the IMAP backends and the Webmail frontends are almost identical, Squirrelmail and Afterlogic performs slightly better, achieving better login times compared to Roundcube and Horde.

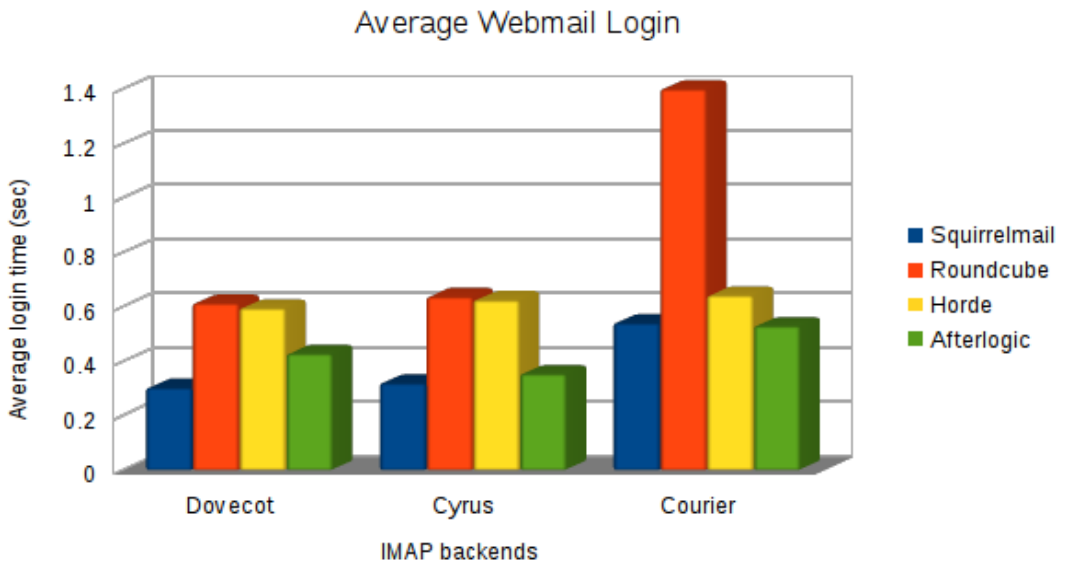


Figure 5: Average fetching time during login

Courier is the backend with the longest latencies, especially in combination with Roundcube. After analysing deeper the exchanged messages between these two, we noticed that for every case (Dovecot-Cyrus), Roundcube opens two tcp sockets simultaneously, one for fetching the first 15 headers from the inbox folder and another one for getting the available folders. This does not happen to the Courier solution and also explains the reason for this difference in performance.

Squirrelmail is the only webmail, among the ones selected for our experiments, that gives the choice to users to display their entire mailbox with only the click of one button. In Figure 6, we investigate how much the fetching time is affected, for this user action, by the different mailbox sizes. We did this by coupling Squirrelmail with all the IMAP backend options.

As expected, as the size of mailbox is increasing, the time where the headers are being fetched is getting longer. Again Dovecot and Cyrus perform better than Courier with Dovecot preceding a little. In the case where our experimental user has a mailbox with 1500 mails with attachments, Courier's behavior diverts from the other IMAP backends. This is due to the fact that Courier parses all the mail files to get the headers, while the others do not.

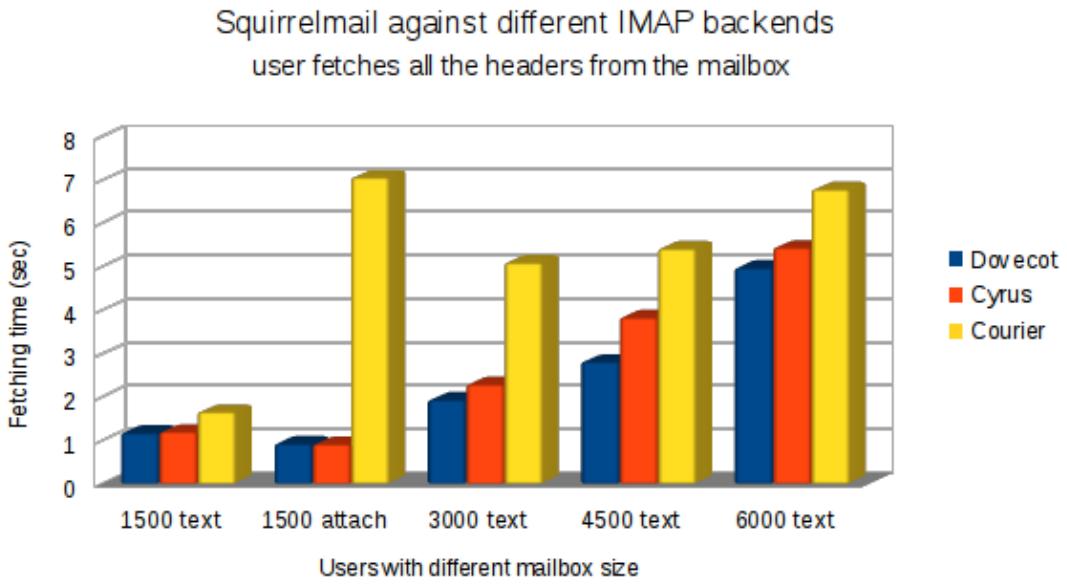
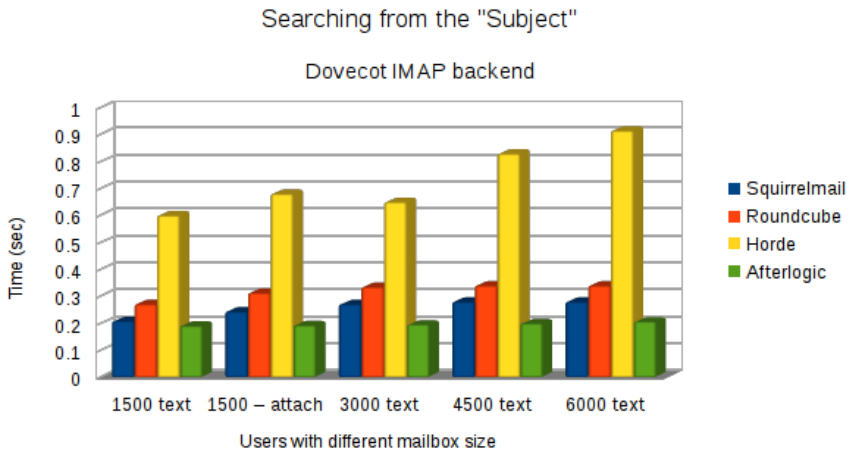


Figure 6: Users request to display all the mails from their inbox

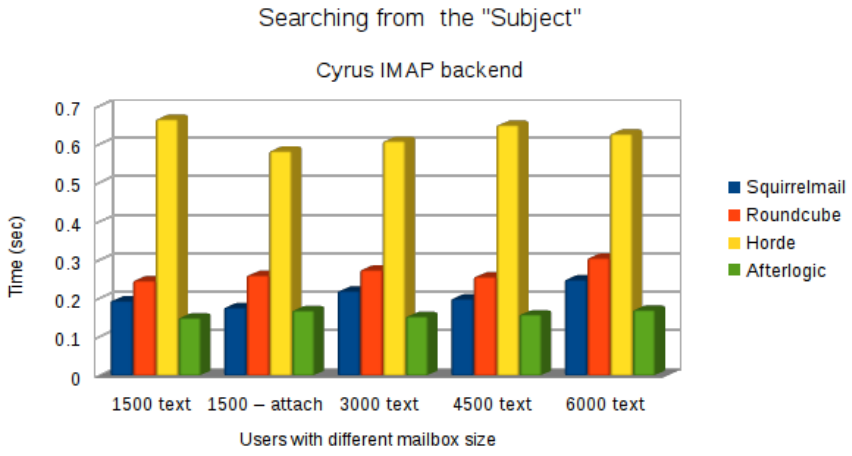
4.1.2 Searching from the "Subject"

In this section we study the required time from the point where the user searches a keyword from the mail's "Subject" field until the time when the response is produced. To give an overall picture, again we conduct the experiments for all the IMAP server-Webmail combinations. Figures 7 (a) and (b) demonstrate the change in behavior in the increment of the user's mailbox size when Dovecot and Cyrus are implemented. Clearly, Afterlogic and Squirrelmail handle the request of the user faster than the others, in time less than 0.3 seconds. Roundcube is slightly slower, while Horde completes the search

within a time interval from around 0.5 to 0.9 sec. To display the search results, Horde uses a new page, that needs to send, unlike the others which send only the part of the page that has been changed.



(a)



(b)

Figure 7: Users search a keyword from the "Subject" field: Dovecot-Cyrus comparison

As for the mailbox size, in the case of Cyrus it seems that it does not affect the searching time. Namely, each of the Webmails performs the search request in the same time, apart from the mailbox size. In contrast, when Dovecot is the IMAP backend the searching time is being increased along with the number of emails.

Figure 8 presents respectively the changes in behavior when Courier is used. Undoubt-

edly, Courier cannot carry out the search requests the same well as Dovecot and Cyrus. The times exceed the ranges of 1 second, reaching even the 6.5 second when the mailbox of the user stores 1500 messages with attachments. This poor performance is because Courier parses the mails in order to find the requested keyword in contradiction with the others which use indexing. Moreover it can be observed the overhead in latency that Horde webmail adds, compared to the other webmail frontends.

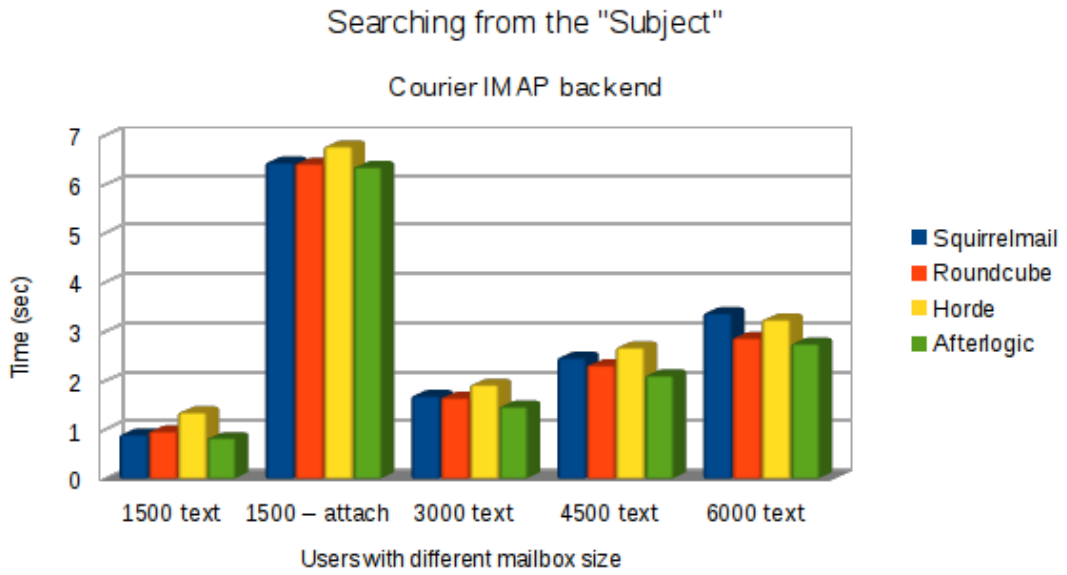


Figure 8: Users search a keyword from the "Subject" field: Courier as a backend

Finally, based on the previous results, the most efficient webmail system solutions were selected for a further comparison. In this comparison, we consider as efficiency their speed in searching within the mailbox for a keyword from the "Subject" field. For each of them, Figure 9 depicts their behavior, regarding the size and the type of user's inbox.

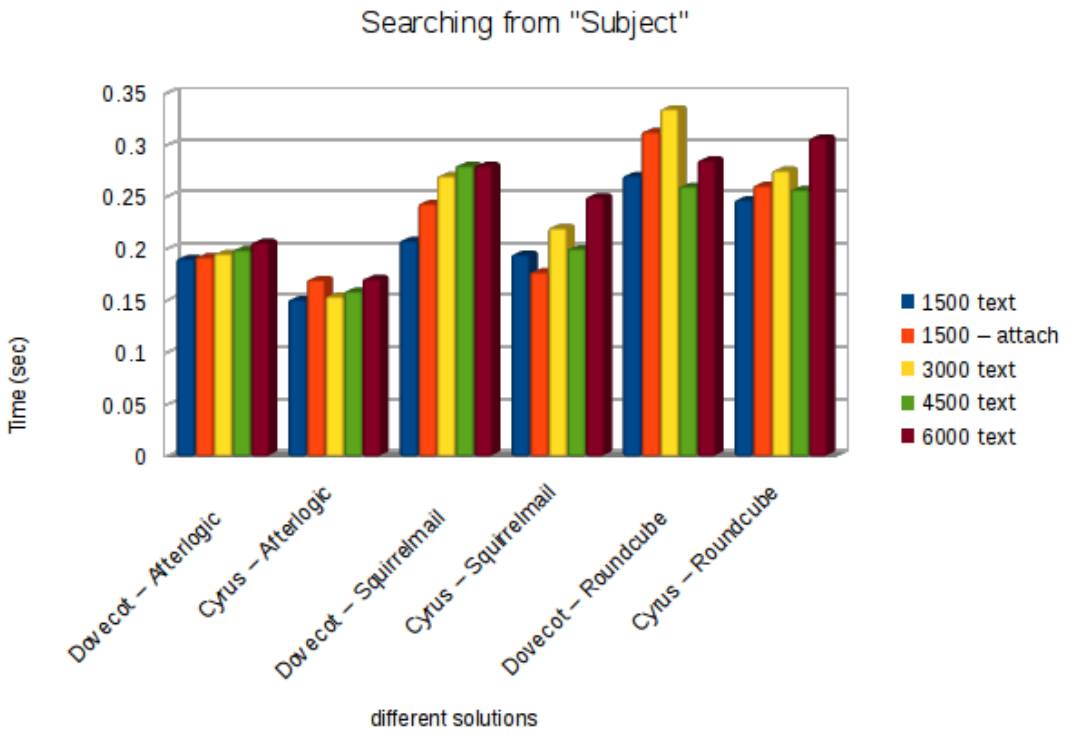


Figure 9: Comparison of the most effective Solutions regarding searching from "Subject"

4.1.3 Searching from the entire message

Within this section the results of our experiments, regarding searching the mailbox for a keyword from the entire message are demonstrated.

Figure 10 shows the change in behavior in the increment of the user's mailbox size when Dovecot is implemented. In this bar chart we can see that as the mailbox size, in terms of number of messages, grows the searching needs more time to be completed. The time needed to search through 1500 messages with attachments is equivalent with time to search through 1500 messages without any attachments. This happens because Dovecot searches for the keyword only on the first part of the body of the stored message and not on the second part of the body which includes the attachment.

As illustrated in Figure 11, Courier behaves the same as Dovecot when the messages in the inbox are increased in number. However, in the case of an inbox full of messages with attachments, Courier searches for the keyword also in the part of the message that contains the attachment. Justifiably, this adds an extra overhead in the searching time. Apart from that it is more than obvious that the overall performance is poor in comparison with Dovecot.

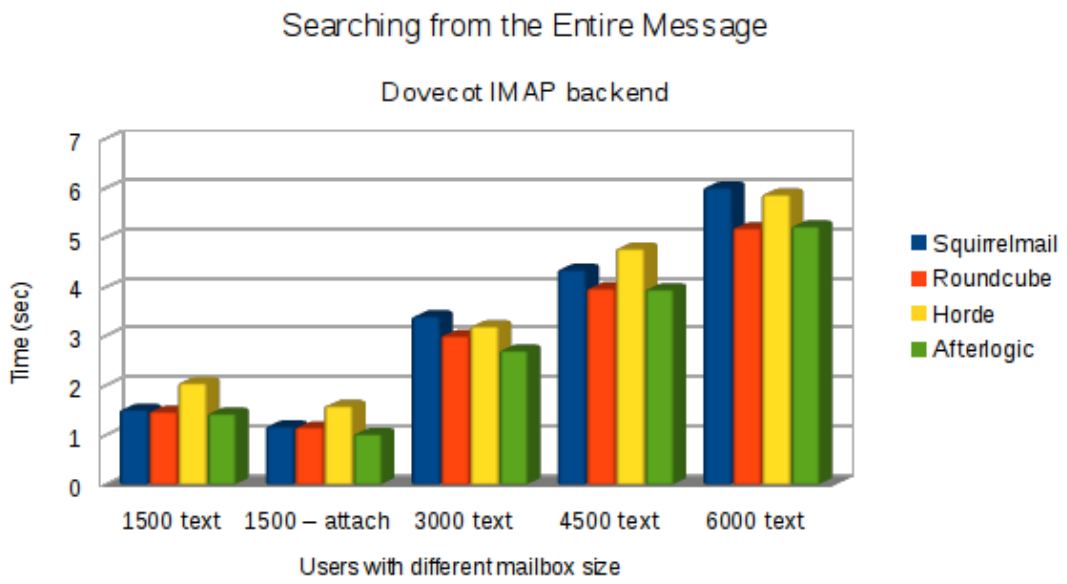


Figure 10: Users search a keyword from the Entire Message Content: Dovecot

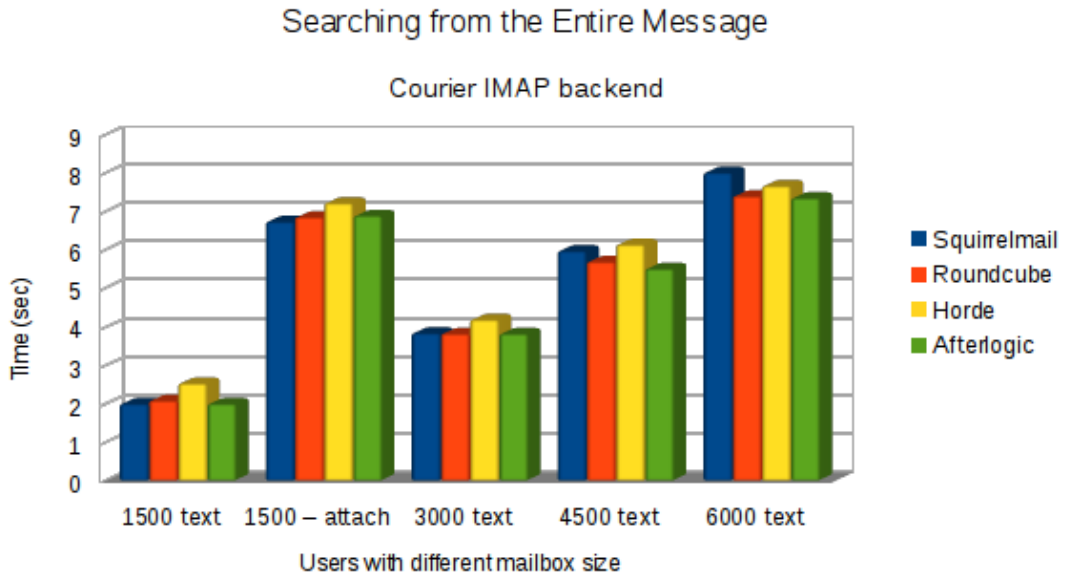


Figure 11: Users search a keyword from the Entire Message Content: Courier

The shortest response times are found in implementations with Cyrus as the IMAP backend. It is clear that the number of messages does not significantly affect the response

times of Cyrus. The most noticeable increment in latency is in the case of the 1500 attached messages but is negligible comparing to Dovecot and Courier. This verifies that the indexing service that Cyrus uses, considerably speeds up the search within the mailbox.

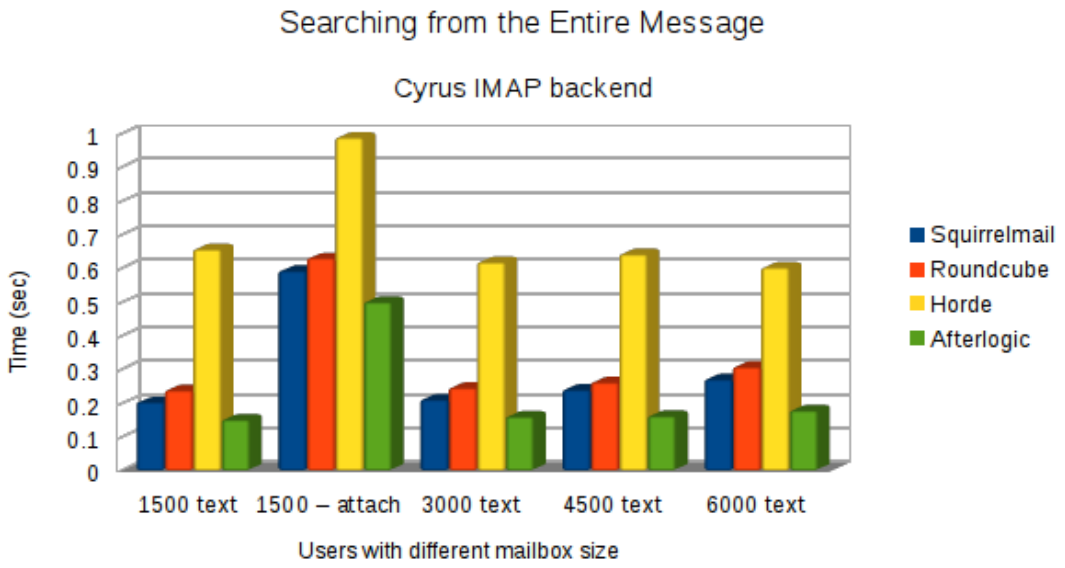


Figure 12: Users search a keyword from the Entire Message Content: Cyrus

As before, the most efficient webmail system solutions were selected for a further comparison. Here, in this comparison, we consider as efficiency their speed in searching within the mailbox for a keyword from the entire message content. For each of them, Figure 13 depicts their behavior, regarding the size and the type of user's inbox.

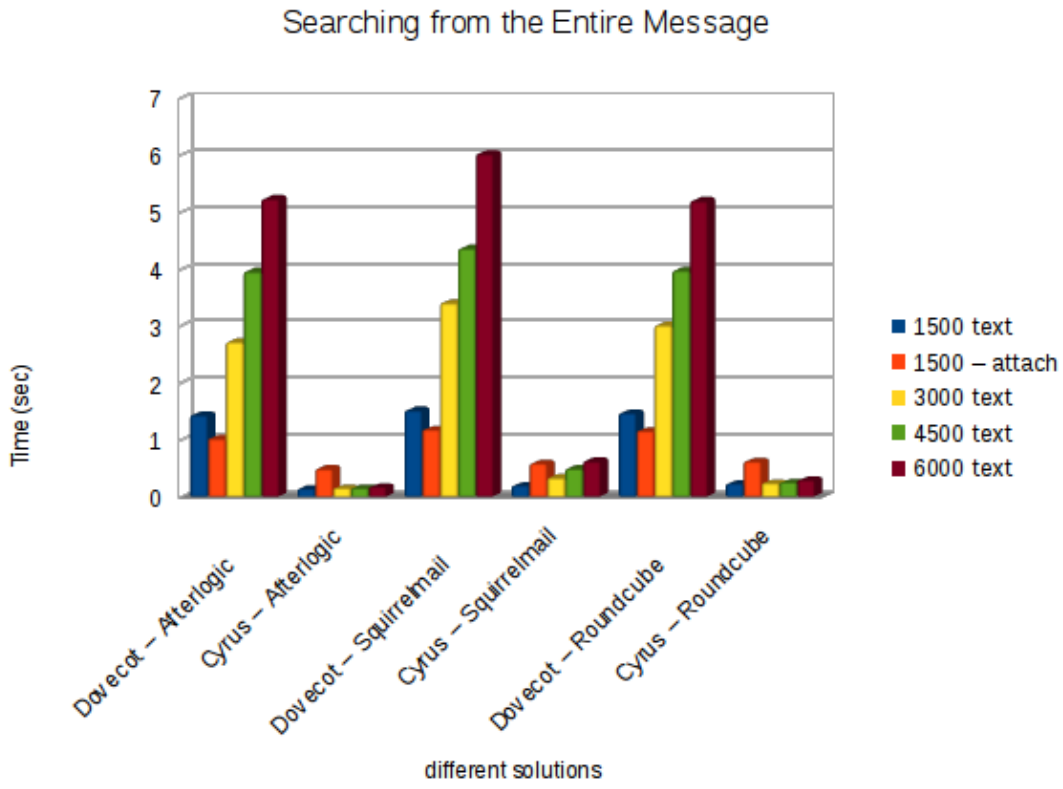
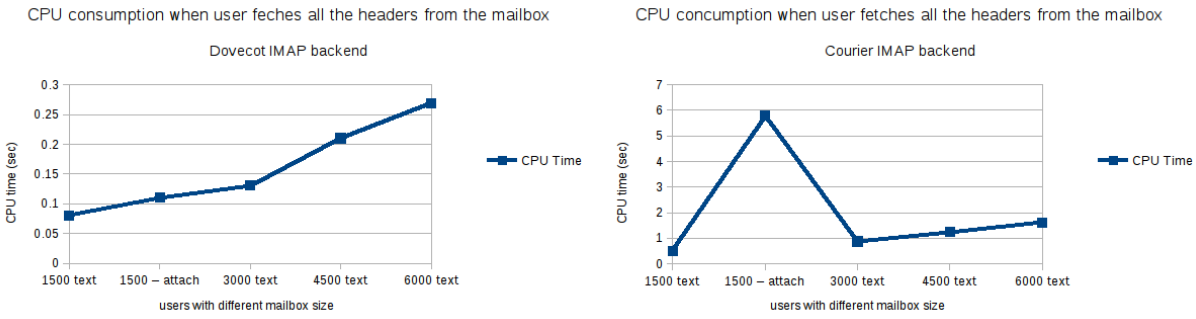


Figure 13: Comparison of the most effective Solutions regarding searching from the entire message content

4.2 CPU time & Memory consumption

A proper evaluation of a webmail system is not consisted solely of the best possible performance that a user perceives. Besides, our results that concern the user experience, our aim is to analyze and evaluate also the impact of these solutions in the performance from the system perspective. In this section, the CPU utilization and the memory consumption of the different IMAP backends are examined. Although, relevant measurements for webmail frontends were taken, the results were insignificant, compared to the utilized resources from the IMAP backends, thus, they are neglected.

We start from analyzing the system resources that are utilized when a user attempts to display all the messages, that is when all the headers from the inbox are fetched. The line graphs in Figure 14 reflect the CPU consumption in terms of CPU time,which has been calculated as specified in Section 2.2.



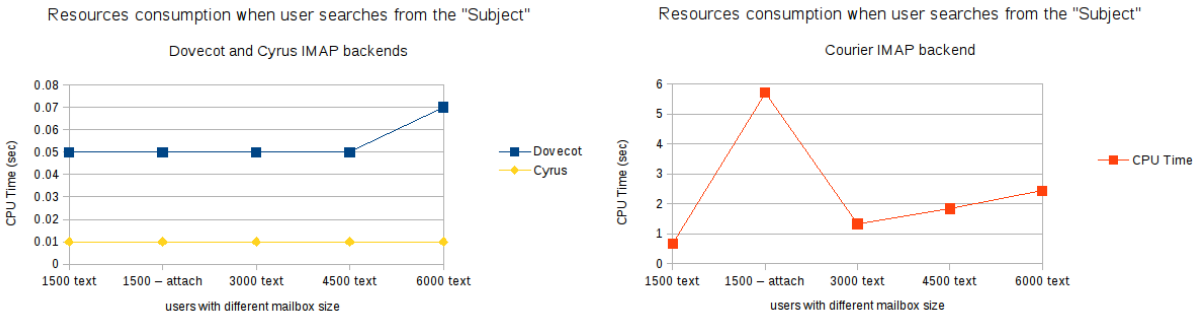
(a) (b)

Figure 14: Users request to display all the mails from their inbox: CPU time consumption for Dovecot and Courier IMAP backends

Compared to Courier, Dovecot needs less CPU operations for the same function, leading to less CPU time, that does not exceed the 0.3 second, even in the case of an inbox of 6000 text messages. On the other hand, Courier utilizes the CPU at least for 0.5 second when the inbox is of 1500 text messages, rising up to 1.6 seconds for 6000 messages. One noticeably detail is that the CPU time jumps to 5.7 seconds when the mailbox is consisted of 1500 messages with attachments. Surprisingly, with Cyrus the levels of CPU consumption were less than with Dovecot and almost equal to zero for all the types of users.

Afterwards, CPU consumption when the user searches from the "Subject" field was measured, providing us the following graphs of Figure 15. We observe that when Dovecot is implemented the values of CPU times are roughly stable while in case of Courier there is an increasing trend, especially for the type of user with the 1500 messages with

attachments. We can also support that searching the entire mailbox for a keyword from the "Subject" field is cheaper, in terms of resources consumption, than displaying all the messages. Again, the levels of CPU consumption with Cyrus were less than Dovecot and almost equal to zero for all the types of users.



(a) (b)

Figure 15: Users search a keyword from the "Subject" field: CPU time consumption for Dovecot and Courier IMAP backends

Finally, the graph on Figure 16 presents the CPU consumption when the user searches for a keyword from the entire message content. Both with Dovecot and Courier the CPU time is being increased as the number of messages in the inbox grows. However, in solutions with Dovecot IMAP backend the CPU time is quite shorter. Especially, for the second type of user, the one with the 1500 attachments in the inbox, Courier needs almost 6 seconds of CPU time while Dovecot does not need more than 1 second. One more time, the results with Cyrus tend to zero.

To avoid erroneous assumptions while studying the above graphs, the line does not indicate continuous CPU increment. The measurements consider only five different sizes of mailboxes. The lines which connect the values of CPU time for the corresponding sizes of mailboxes are placed only for the ease of the reader.

The following three graphs depict the average memory consumption when the different IMAPs process user requests through webmail. As for the first graph in Figure 17, we need to point out that in the idle state, this is the state where no communication takes place between the frontend and Dovecot IMAP, only three of these processes are running; dovecot, dovecot/log and dovecot/anvil. The other four are spawned when a user logs in and uses the IMAP server and for each user a different dovecot/imap process is started. The dovecot/imap process differentiates approximately from 2500 Kbytes to 6000 Kbytes.

When Courier is in idle state only three different processes are running; courier-authdaemon,

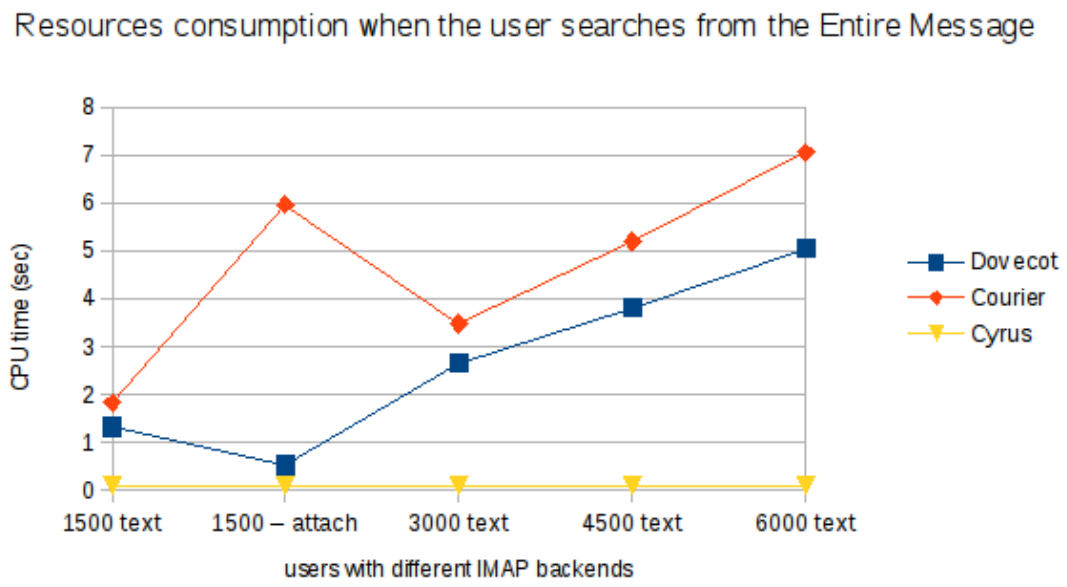


Figure 16: Users search a keyword from the Entire Message Content: CPU time consumption for Dovecot and Courier IMAP backends

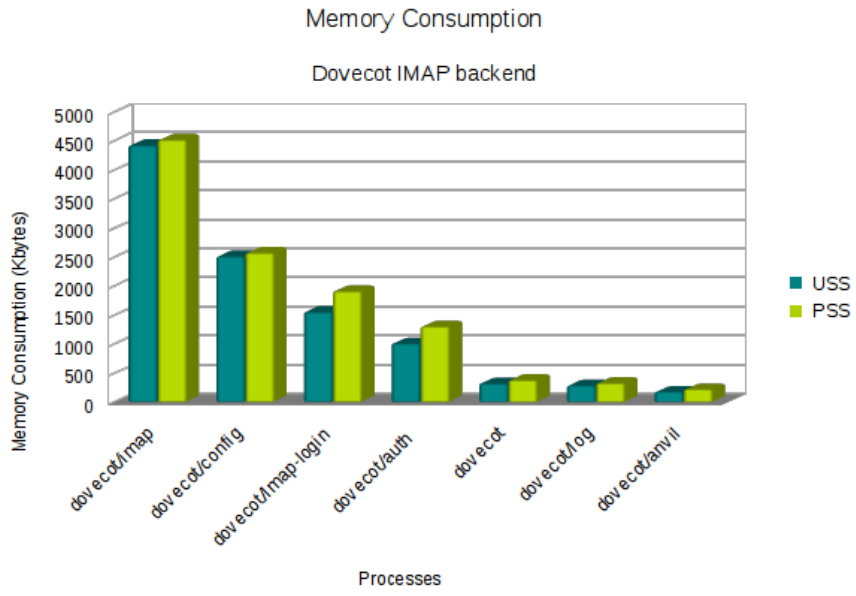


Figure 17: Average memory consumption of different processes of Dovecot

couriertcpd and courierlogger, while the other two are spawned whenever it is needed. The bar chart in Figure 18 presents the average memory consumption when a user interacts with the Courier IMAP server.

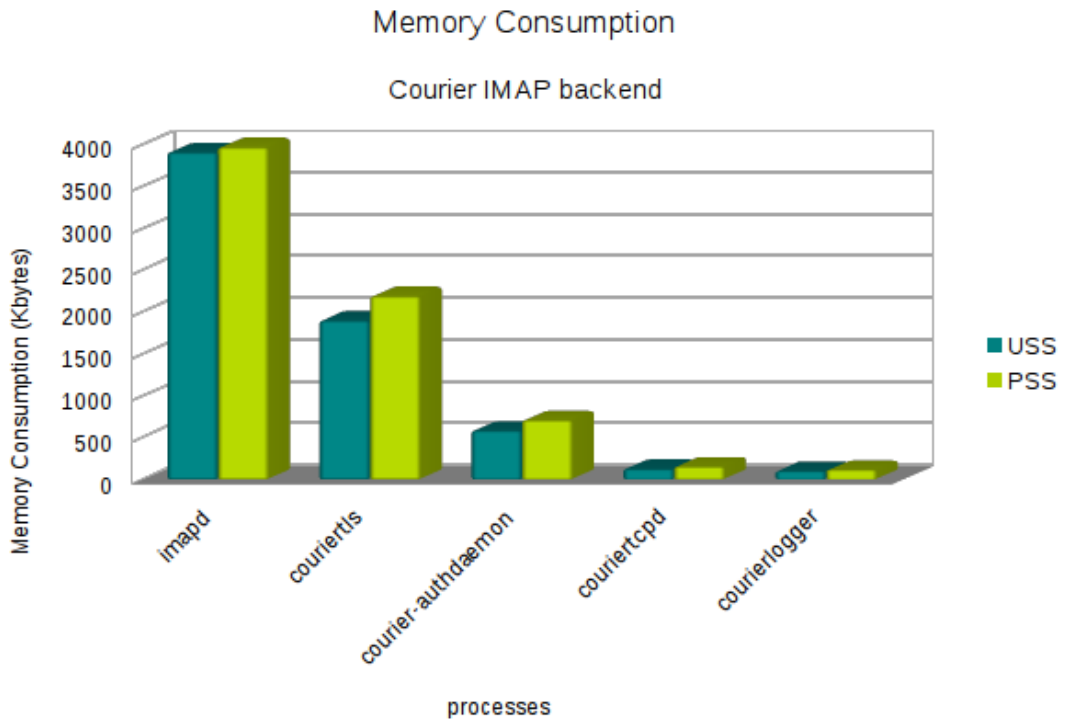


Figure 18: Average memory consumption of different processes of Courier

Cyrus IMAP in idle state is running cyrus-imapd process and about 14 imapd daemons occupying 932 Kbyte and 1014 Kbyte USS and PSS respectively each of them. When a user logs in and uses the IMAP server only three of these processes differentiate in terms of memory usage (imapd #1, #2 and #3). The average memory consumption for Cyrus IMAP is depicted in Figure 19.

Concluding, Cyrus has the highest memory consumption, followed by Dovecot and finally by Courier. Based also on the previous results, which concern the latency that a user perceives when sends requests such as searching a keyword or displaying all the messages, it seems that the more efficient the backend the more the memory it consumes.

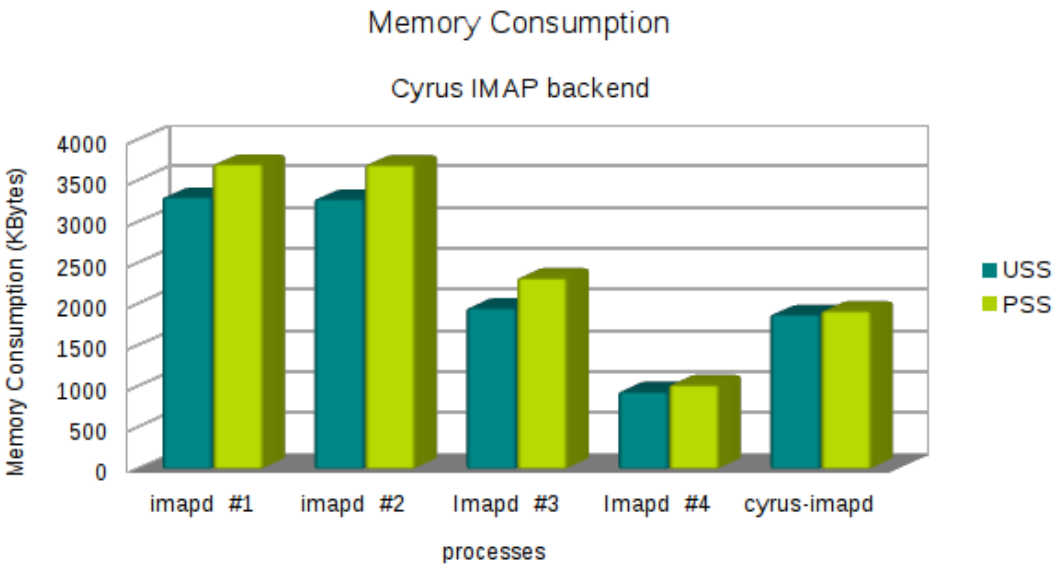


Figure 19: Average memory consumption of different processes of Cyrus

5 Conclusions

This paper has presented a detailed analysis of the effectiveness of four different open source Webmail services, Squirrelmail, Roundcube, Horde and Afterlogic. All of these services were combined with three different IMAP backends in order for the most efficient solution to be found. We mainly focused on the most common actions that users perform, such as searching through their Webmails for a certain message or displaying the entire inbox folders. We made our evaluation in two different dimensions: from user experience perspective and from system performance perspective. To this end three metrics were used: latency, CPU time and memory consumption.

After our experiments, it seems that Squirrelmail is the fastest, among the selected webmail applications, during the user login. This is due to its lightweight page. Afterlogic achieves shortest latencies for both searching from the "Subject" field and from the entire message content. In addition, because of the AJAX use (pre-caching on client side), Afterlogic provides responsiveness similar to desktop email clients. On the other hand, Horde has the longest response time, especially during searching. This happens because Horde sends a new page to display the search results, unlike the others which send only the part of the page that has been changed.

It is observed that the major bottleneck in an integrated webmail system is the IMAP backend. Experiments were conducted with Dovecot, Courier and Cyrus implementations. All of them perform almost the same, in term of latency, when a user logs in through the webmail frontends. Dovecot and Cyrus can carry out the search from "Subject" request, with relatively the same latency, and much better than Courier. However, when the user searches a keyword from the entire message content, the latency is far shorter if Cyrus is the IMAP backend, followed by Dovecot. The indexing feature that Cyrus and Dovecot provide is capable of more efficient searching.

Besides, our results that concern the user experience, our aim was to analyze and evaluate the impact of these webmail system solutions in the performance from the system perspective. Cyrus has the highest average memory consumption for all the IMAP functions, followed by Dovecot and finally by Courier. On the contrary, Cyrus has the lowest CPU utilization followed by Dovecot and Courier.

Summarizing the solution of using Cyrus IMAP combined with Afterlogic Webmail Lite performs better in terms of both user experience and system overall performance.

6 Future Work

Over the world hundreds of millions of end users depend on e-mail technologies. However, open source web frontends to mail servers remain a neglected area of development. Hopefully, the work that has been done here will be a contribution to better understanding how the performance of web mail applications is determined. We consider that this is a prerequisite for better tuning and further development of webmail applications.

Through our remarks, which have been derived from experimental results, our expectation is to trigger the further development of the products to improve their performance. We think that there is enough room for improvement both in Webmail and IMAP backend projects.

In our research, the data set used was made from e-mail messages from the mailbox of an actual user. While this provided reliable real-world data, a larger synthetic test suite (from e.g. Wikipedia content) based on the general characteristics of an email box would have produced experiments reproducible by others, including the developers of the various applications. We therefore recommend the creation of such a test suite as a next step of future research in this area.

Our future plans include the evaluation of different functions that a webmail system provides, like deleting or copying messages. Moreover, we are interested in an evaluation of an integrated webmail system, while an IMAP proxy is implemented between the webmail server and the different IMAP servers. A relevant and also interesting research would be to investigate how the amount of load of these systems impacts their responsiveness.

List of Figures

1 Major e-mail Components 3
2 Postfix and security modules 11
3 Roundcube user interface 14
4 Afterlogic Webmail Lite user interface 14
5 Average fetching time during login 17
6 Users request to display all the mails from their inbox 18
7 Users search a keyword from the "Subject" field: Dovecot-Cyrus comparison 19
8 Users search a keyword from the "Subject" field: Courier as a backend . . 20
9 Comparison of the most effective Solutions regarding searching from "Subject" 21
10 Users search a keyword from the Entire Message Content: Dovecot 22
11 Users search a keyword from the Entire Message Content: Courier 22
12 Users search a keyword from the Entire Message Content: Cyrus 23
13 Comparison of the most effective Solutions regarding searching from the entire message content 24
14 Users request to display all the mails from their inbox: CPU time consumption for Dovecot and Courier IMAP backends 25
15 Users search a keyword from the "Subject" field: CPU time consumption for Dovecot and Courier IMAP backends 26
16 Users search a keyword from the Entire Message Content: CPU time consumption for Dovecot and Courier IMAP backends 27
17 Average memory consumption of different processes of Dovecot 27
18 Average memory consumption of different processes of Courier 28
19 Average memory consumption of different processes of Cyrus 29

Bibliography

[1] Hafsaoui Aymen et al. *"A study of email usage and performance over Cellular technology"*. Communications and Networking (ComNet), 2010 Second International Conference on. IEEE, 2010.

[2] Jun Wang and Yiming Hu. *"A performance study on Internet Server Provider mail servers"*. Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on. Vol. 1. IEEE, 2004.

[3] Ramesh Karne Appiah-Kubi, Patrick and Alexander Wijesinha. *"Design and Performance of a Webmail Server on Bare PC"*. High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on. IEEE, 2010.

[4] Randall Gellens and John C. Klensin. *"Message Submission for Mail"*. RFC 4409, 2006.

[5] C. P. J. Koymans and J. Scheerder. *""Email." Handbook of Network and System Administration"*. Elsevier Science, 2007.

[6] John C. Klensin. *"Simple Mail Transfer Protocol"*. RFC 2821, 2001.

[7] A. Menon-Sen R. Siemborski. *"The Post Office Protocol (POP3) Simple Authentication and Security Layer (SASL) Authentication Mechanism"*. RFC 5034, 2007.

[8] C. Newman. *"Using TLS with IMAP, POP3 and ACAP"*. RFC 2595, 1999.

[9] M. Crispin. *"Internet Message Access Protocol - Version 4rev1"*. RFC 3501, 2003.

[10] A. Gulbrandsen R. Siemborski. *"IMAP Extension for Simple Authentication and Security Layer (SASL)"*. RFC 4959, 2007.

[11] David A. Patterson and John L. Hennessy. *"Computer organization and design: the hardware/software interface"*. Morgan Kaufmann, 2008.

[12] Chandra Thimmannagari. *"CPU design: answers to frequently asked questions"*. Springer, 2004.

[13] [Online]. Available: `http://packages.ubuntu.com/precise/sysstat`.

[14] [Online]. Available: `http://packages.debian.org/sid/utils/smem`.

[15] [Online]. Available: `http://www.tcpdump.org/`.

[16] [Online]. Available: `http://www.wireshark.org/download.html`.

[17] *Xen.org, Xen hypervisor 4.2.1 Download.* [Online]. Available: `http://www.xen.org/download/index4.2.1.html/`.

[18] [Online]. Available: `ftp://ftp.nl.uu.net/pub/unix/mail/postfix/`.

[19] [Online]. Available: `http://www.ijs.si/software/amavisd/`.

[20] [Online]. Available: `http://www.clamav.net/lang/en/download/sources/`.

[21] [Online]. Available: `http://spamassassin.apache.org/`.

[22] *"Dovecot Wiki"*. [Online] Available: `http://wiki2.dovecot.org/`.

[23] Peer Heinlein and Peer Hartleben. *"The Book of IMAP: Building a Mail Server with Courier and Cyrus"*. No Starch Press.