UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Annika Laumets

Migration from virtual environment to physical infrastructure of Raspberry Pis

Bachelor's Thesis (9 ECTS)

Supervisor: Artjom Lind, MSc

Tartu 2016

Üleminek virtuaalkeskkonnast Raspberry Pi arvutitest koosnevale füüsilisele infrastruktuurile

Lühikokkuvõte: Paljud firmad rendivad privaatseid virtuaalservereid, kus nende e-maili server ja veebiserver asuvad. Privaatserveri rentimine võib väikefirmale olla üpris kulukas. Kuna virtuaalserverid asetsevad tavaliselt püstikuserverites, mis kasutavad palju elektrit ja mis pool ajast ei kasuta kogu oma potentsiaali, siis oleks odavam kasutada Raspberry Pi arvutit. Selles töös kasutati ühte Raspberry Pi arvutit, et üles seada veebiserver ning meiliserver, ning seejärel mitut Raspberry Pi arvutit. Kasutusel oli kaks erinevates programmeerimiskeeltes kirjutatud rakendust, mis esialgses püstituses olid mõlemad samal Raspberry Pi arvutil. Seejärel vaadati võimalust, kuidas need samad rakendused üle viia erinevatele arvutitele ja et on olemas peaserver, mis on puhverserver (*proxy server*). Selles töös võrreldi nende kahe püstituse päringute lõpetamise kiirust Apache serveri jõudlustestide abil. Päringute lõpetamisel oli mõlema rakenduse (Roundcube'i meiliserver ja MediaDropi videoplatvorm) korral kiirem kolme Raspberry Pi arvutiga üles seatud lahendus. MediaDrop rakendusel esines tõrkeid päringutele vastamisel sellel samal püstitusel. Lisaks uuriti rahalist erinevust privaatse virtuaalserveri rentimise ja Raspberry Pi arvuti kasutamise vahel. Rahalises mõttes leiti, et Raspberry Pi kasutamine ühe aasta jooksul tuleks 20€ odavam kui virtuaalserveri rentimine.

Võtmesõnad: Raspberry Pi, Apache veebiserver, Roundcube meiliserver, MediaDrop videoplatvorm

CERCS: T120, Süsteemitehnoloogia, arvutitehnoloogia

Migration from virtual environment to physical infrastructure of Raspberry Pis

Abstract: Many companies use a virtual private server where they have a web server and a mail server. Renting a private server can become quite a big expenditure for a small company. Since many virtual servers are actually running in tower servers, which use a lot of electricity and they are still underutilised, it might be cheaper to use a Raspberry Pi computer. In this thesis, one Raspberry Pi computer was used to set up a web server and a mail server and then the same services were migrated to three Raspberry Pi computers. Two applications written in different programming languages where used and originally were located on a single Raspberry Pi computer. Later a possibility was tested were the same applications were on different Raspberry Pi computers and a main server was also in place acting as a proxy server. In this thesis, Apache server benchmarking tool was used to test the time to serve requests. The request serving time was faster with three Raspberry Pi computers than one Raspberry Pi computer, but one application had some failed requests with the three Raspberry Pi computer setup. Furthermore, the financial gain was calculated when using a Raspberry Pi computer instead of renting a virtual private server. It was found that the gain would be 20€ for a year.

Keywords: Raspberry Pi, Apache web server, Roundcube mail server, MediaDrop video platform

CERCS: T120, Systems engineering, computer technology

Contents

List of Figures

1 Introduction

Tower server setups have been used widely for many decades. Dedicated web servers are exclusively rented computers that have a Web server, related software and connection to the Internet. In today's world, dedicated web servers, are used quite a lot, especially by small-scale to medium-scale enterprises. To have a typical tower server or renting a virtual server for an enterprise of the aforementioned size would be quite a big expenditure since operational costs are quite big and the companies usually do not have that kind of money to spend every month to pay the electricity bill. Moreover, energy resources have been starting to be exhausted and people have had to start thinking "greener" to save Earth's resources for future generations. This also includes reducing power consumption to reduce the usage of oil, coal and so forth to produce electricity. This mindset has also expanded to computer science and people have started thinking about reducing power consumption of tower servers because they use a lot of energy and most of the time they are still underutilised [3]. To conclude, reducing power consumption of tower servers while still maintaining a good enough usage would help both the environment and small-to-medium enterprises.

Raspberry Pi is a low-power computer and it is widely available. Therefore, it would be the perfect solution for greening web servers. Since it uses less power it would be environmentally friendly and operational costs would also lower because of the reduced power consumption. This paper will explore the idea of using one Raspberry Pi as a web server hosting two applications - Roundcube web server and MediaDrop video platform. In addition, a setup of three Raspberry Pi computers to host the same applications is tested. Small and medium enterprises still use dedicated servers for their applications and websites, but even single-computer servers used approximately 6% of energy in the U.S in 2006 [4]. Therefore, using a Raspberry Pi computer as a dedicated server would reduce costs and even more when using one Raspberry Pi computer for different applications.

This thesis is organised as follows. Chapter 2 discusses related work. Chapter 3 describes the design of the solution and used technologies. The solution implementation is presented in Chapter 4. The analysis of the results is written in Chapter 5. Lastly, there will be a conclusion.

2 Related work

The power consumption of servers has been researched thoroughly, although not many of these papers suggest how to reduce energy consumption.
Olson et al. [5] described a prototype for small and medium enterprises (SME) to reduce energy consumption. One of the platforms was to be a low-power one which "is used during the periods of time that the request rate (demand) is low and thus allowing the high-power platform to sleep, hence energy is saved". The platforms in the prototype would support the same web server software and the content would be mirrored. This solution would reduce both operational costs and environmental damage. In the paper it was estimated that the electricity cost of servers and data centers was about $4.5 billion in 2006. Because of this Olson et al. suggested a system with two co-located platforms that appear as one website to the client. A mechanism for switching between platforms was also described and test results showed that the high-power platform and low-power platform had almost the same performance for delivering static pages. However, for active pages the high-power platform was faster.

Even fewer papers suggest some other hardware to replace the tower setup that is normally used at the moment. Migration of the services and applications is also not discussed. But Varghese et al. [6] studied the benefits of using low-power web servers. They used Raspberry Pi computers as low-power web servers. It was found that the performance advantage with the typical tower server was only 2.3x. Low-power systems "could sustain request rate of up to 200 requests per second" and with dynamic content up to 20 requests per second. Furthermore, the Raspberry Pi cluster could server 17x to 23x more requests per Watt than a typical tower server at high arrival rates. It was also found that when using multiple low-power devices the power reduction could be approximately 20x. Different dynamic policies were also considered to further reduce the power consumption and the always ON policy was much better than the delayed turn OFF policy "for reducing both the energy as well as the number of server transitions".
The Apache HTTP Server is an open-source HTTP (web) server and it is part of the Apache Software Foundation. It was launched in 1995 and after a year of the launch, it became the most popular web server on the Internet and has been on the top since then [7]. The name 'Apache' comes from the Native American Indian tribe of Apache. The tribe is "known for their superior skills in warfare strategy and their inexhaustible endurance" [8]. The Apache Software Foundation was formed "to provide organizational, legal, and financial support for the Apache HTTP Server" [9]. There are also other open-source software projects that are part of the Apache Software Foundation, e.g. Hadoop, Tomcat, and BookKeeper [10].
The Raspberry Pi computer is a small-sized computer that is low cost both in

price and in energy consumption. It is low power becuase it is ARM-based. It can be used with standard keyboard and mouse and it plugs into a monitor. The user is able to browse the Internet, create files, modify files and play games on the Raspberry Pi computer. It was developed to help children understand how computers work and how to program on them [11]. The Raspberry Pi Foundation was created to "advance the education of adults and children, particularly in the field of computers, computer science and related subjects" [12].
For different applications different modules and programming languages were used. Python and PHP are languages that are being tested in this thesis.
Firstly, PHP is a "widely-used Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML" [13]. PHP stands for Hypertext Preprocessor and the syntax of PHP is influenced by C, Java, and Perl [13]. PHP was created by Rasmus Lerdorf in 1994 and soon Andi Gutmans and Zeev Suraski joined him in developing PHP [14]. Today dozens of developers work on developing PHP. In April 2016 PHP was in 6th rank according to the TIOBE Index and in 3rd rank according to the PYPL Index [15] [16]. The module in Apache for PHP is called `php5`. It can be downloaded by using the `apt-get` command and the it has to be enabled for it to work. Both of these command are present in figure 1. After restarting Apache server, the module should work and pages written in PHP can be viewed.

```
root@raspberrypi:~# apt-get install libapache2-mod-php5
root@raspberrypi:~# a2enmod php5
```

Figure 1: Command to install PHP module for Apache server

The second programming language selected is Python. It is defined as "an interpreted, interactive, object-oriented programming language" [17]. Guido van Rossum created Python in the 1990s [18]. In April 2016 Python was in 5th rank according to the TIOBE Index and according to the PYPL Index it was in 2nd rank [15] [16]. For Python to work in Apache, a interface has to be installed and enabled. There are 4 ways for Python to work in web [19]:

1. Common Gateway Interface (CGI) - the oldest interface and it is supported by many web servers; since every request starts a new Python interpreter, the interface makes servers slow;

2. Apache module `mod_python` - faster than CGI because the interpreter is embedded into the Apache process; when changing files the web server has to be restarted because the interpreter uses caching of files; difficult to switch between versions of Python without recompiling `mod_python`;

3. FastCGI and SCGI - long-running background processes are created instead of embedding the interpreter into the web server; SCGI is basically like FastCGI, but it has little web server support; FastCGI is not used directly nowadays and it is only used for deployment of WSGI applications;

4. Apache module `mod_wsgi` - specifically designed to host WSGI applications and it directly embeds WSGI applications into the web server, therefore it doesn't need glue code; it is only available for Apache web server.

3 Solution design

In the practical part of this thesis Raspberry Pi 2 Model B computers were used. In the first section of the implementation only one Raspberry Pi is used. The Raspberry Pi computer is connected to a router. On the Raspberry Pi computer, there is an Apache server with a main page and two applications. The main page has links to the applications. Before accessing the applications HTTP authentication takes place to ensure that whoever is trying to access the applications is who they claim they are. The authentication takes place under the URL `/apps` and if somebody is trying to access a sub-URL of the URL `/apps`, authentication takes place. The applications are located under the URLs `/apps/app1` and `/apps/app2` respectively. The first application will be Roundcube mail application and the second one is MediaDrop video platform. The setup is also displayed in figure 2 with URLs mapped to the correct application.

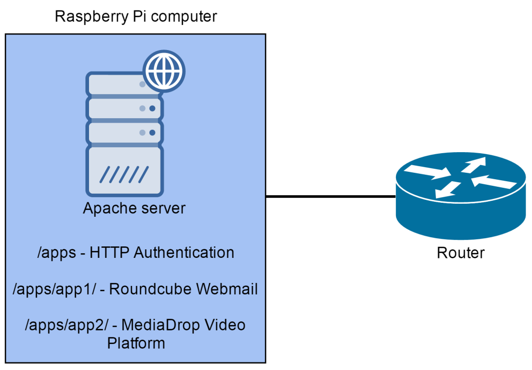

Raspberry Pi computer
Apache server
/apps - HTTP Authentication
/apps/app1/ - Roundcube Webmail
/apps/app2/ - MediaDrop Video Platform
Router

Figure 2: Setup with one Raspberry Pi computer

After that the applications are migrated to different nodes - the first application on one node and the second one on another node. Roundcube application will be on host `raspberrypi2` and MediaDrop video platform on host `raspberrypi3`. The authentication and the main page will stay on the original node whose hostname is `raspberrypi`. Requests to the applications will be redirected to other nodes. The original node acts as a proxy server and maps the URLs to the remote systems. The Raspberry Pi computers are all connected to the same router, so they can easily communicate with each other without any special networking. The design of the three node setup can be seen in figure 3.

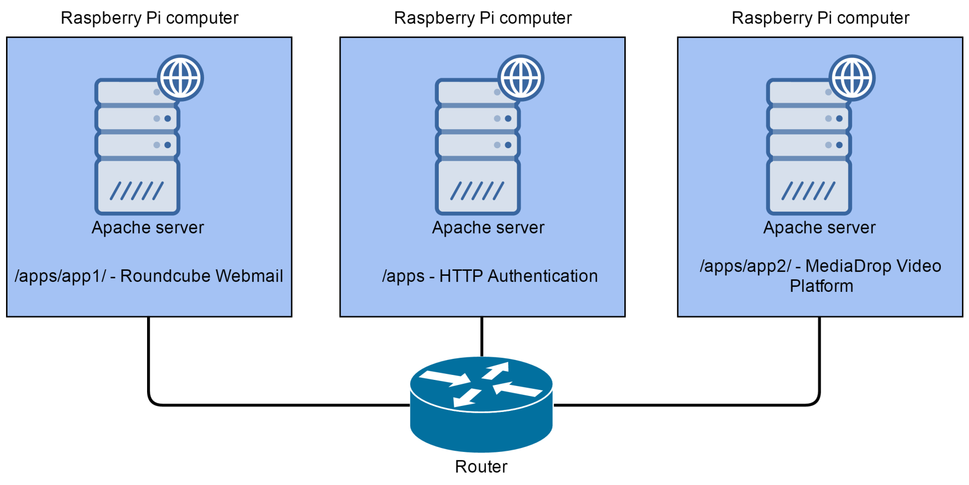


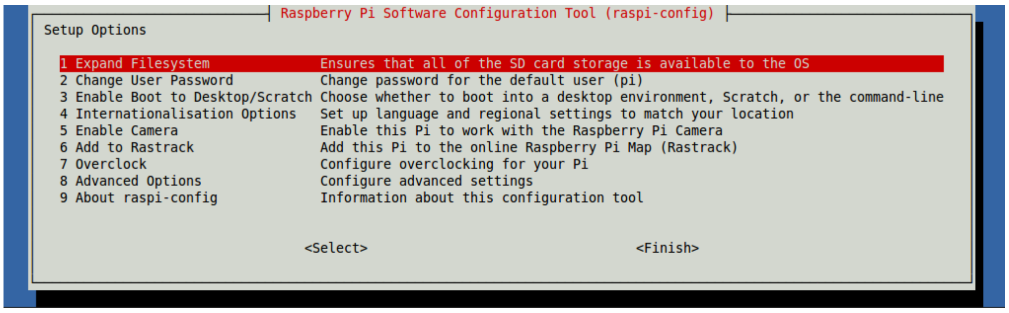
Figure 3: Setup with three Raspberry Pi computers

4 Solution implementation

The implementation consists of two parts. The first part will be about setting up one Raspberry Pi computer to have an Apache server and two applications written in different programming languages. The second part will show how to migrate two applications from one Raspberry Pi computer to two Raspberry Pi computers and have one Raspberry Pi computer as a proxy server and performer of HTTP authentication.

4.1 Setup with one Raspberry Pi computer

Computers need to have an operating system to work. Since the Raspberry Pi computers used for this thesis all have a micro SD slot [20] [21], the operating system should be on a micro SD card that has been formatted and made bootable. The operating system used is Raspbian Jessie Lite which is the latest Raspbian version. Because the latest version of Raspbian is based on Debian Jessie, a lot of the packages that are available for Debian Jessie are available for Raspbian. Raspbian is the official supported operating system for Raspberry Pi computers [22].
When the micro SD card is formatted and made bootable with the Raspbian Jessie Lite image, the Raspberry Pi computer is ready to be started. After the computer has started, it is necessary to change some settings, i.e. hostname, locale, timezone, keyboard layout, the password for the user "pi", enabling SSH server and expanding the file system. All of the aforementioned settings and some more can be changed with the command `raspi-config` in the command line which opens Raspberry Pi Software Configuration Tool. Figure 4 shows the list of options that are displayed with the command `raspi-config` in Raspberry Pi Software Configuration Tool.



```
Raspberry Pi Software Configuration Tool (raspi-config)
Setup Options

 1 Expand Filesystem              Ensures that all of the SD card storage is available to the OS
 2 Change User Password           Change password for the default user (pi)
 3 Enable Boot to Desktop/Scratch Choose whether to boot into a desktop environment, Scratch, or the command-line
 4 Internationalisation Options   Set up language and regional settings to match your location
 5 Enable Camera                  Enable this Pi to work with the Raspberry Pi Camera
 6 Add to Rastrack                Add this Pi to the online Raspberry Pi Map (Rastrack)
 7 Overclock                      Configure overclocking for your Pi
 8 Advanced Options               Configure advanced settings
 9 About raspi-config             Information about this configuration tool

                    <Select>                                  <Finish>
```

Figure 4: Raspberry Pi Software Configuration Tool [1]

```
root@raspberrypi:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -j DROP
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
```

Figure 5: Command to list all current firewall rules

Furthermore, the firewall has to be configured to allow requests made to port 80 to allow requests to Apache server and port 22 to allow SSH connections. This can be achieved using the command `iptables`. The `iptables -S` command in figure 5 shows how to list all current firewall rules. It can be seen that in the Raspberry Pi computer port 80 and port 22 are open and accepting packets. In figure 6 the command adds a firewall rule that accepts packets on port 80. The same command can be used with SSH for port 22 but the option `--dport` has to be specified as 22. It should be noted that when restarting or powering off the server, the firewall rules are deleted, so they should be written in a file that is read when the server boots up.

```
root@raspberrypi:~# iptables -A INPUT -m state --state NEW -p tcp
 ↪  -m tcp --dport 80 -j ACCEPT
```

Figure 6: `iptables` command to accept packets on port 80

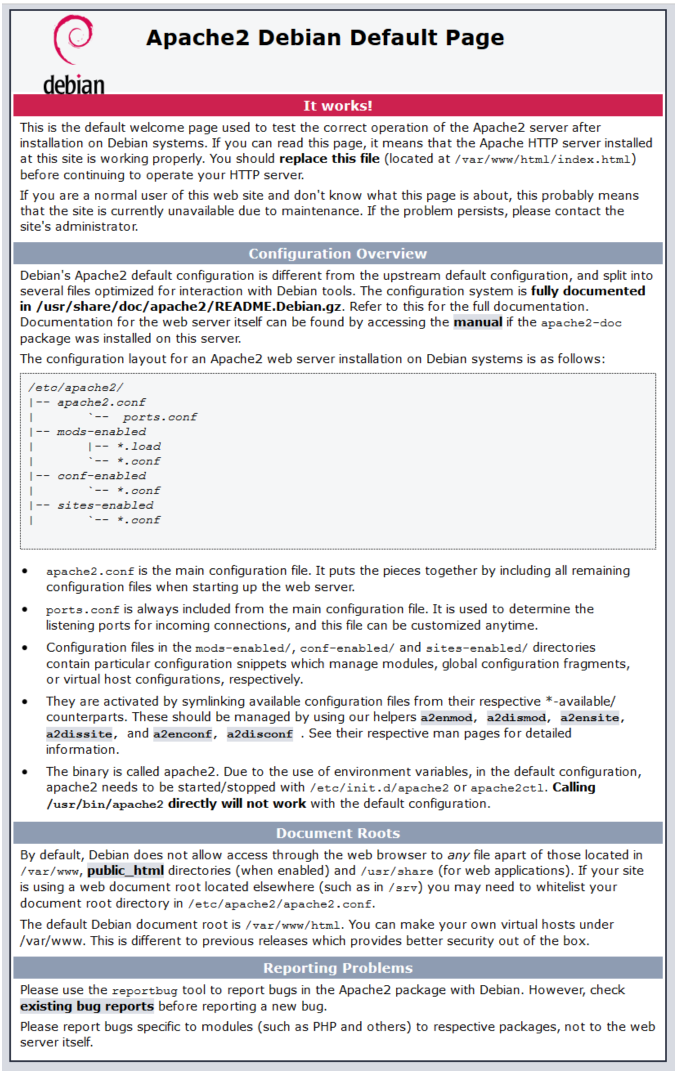
4.1.1 Apache server

Apache can be installed using `apt-get` as seen in figure 7. The `apache2` package consists of full installation, configuration files, init scripts and support scripts [23].

```
root@raspberrypi:~# apt-get install apache2
```

Figure 7: Apache installation

Web pages are controlled by configuration files in Apache server. The configuration files are located in `/etc/apache2/sites-available` directory. To enable a site, it is advised to use a built-in command `a2ensite`, and to disable a site, the command `a2dissite`. The commands create or remove symbolic links to site configuration files accordingly. After installation has finished, it is advised to check if the user can access the default page of Apache. It should look like figure 8. The default configuration is enabled by default and an `index.html` page is also created in `/var/www/html` directory, the default directory for web site files.



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers **`a2enmod`**, **`a2dismod`**, **`a2ensite`**, **`a2dissite`**, and **`a2enconf`**, **`a2disconf`** . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Debian does not allow access through the web browser to *any* file apart of those located in `/var/www`, **public_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Debian document root is `/var/www/html`. You can make your own virtual hosts under /var/www. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the `reportbug` tool to report bugs in the Apache2 package with Debian. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

Figure 8: Apache default page

The command `a2dissite 000-default.conf` in figure 9 disables the configuration file of the default page. Since web site configuration files are processed in alphabetical order, it is necessary to disable the default configuration file in Apache because it is always processed first because the name of the file starts with zeros. Therefore, it is ensured that the new configuration file will be processed first when the default configuration file is disabled.

```
root@raspberrypi:~# a2dissite 000-default.conf
```

Figure 9: Command to disable the default site of Apache

Now the user can make his/hers own configuration file for a web site. The default configuration file is a good place to look for usage of different directives in Apache configuration files. Since in this thesis different domain names are not important, the user can leave out the `<VirtualHost>` directive in the configuration file. Figure 10 shows an example configuration file without the `<VirtualHost>` directive. The `DocumentRoot` directive specifies "the main document tree visible from the web" and the `LogLevel` directive "controls the verbosity of the ErrorLog" [24]. The level in `LogLevel` by default is `warn` which means that Apache logs warning conditions. The `ErrorLog` and `CustomLog` directive both specify where the server logs errors and access respectively. It is also advisable to name the error log files and access log files with a way to distinguish them from other configuration's error and access logs. In figure 10 the errors are written in a file named `main-error.log` and access requests into file named `main-access.log`. The `combined` keyword in `CustomLog` directive means it uses the common log format with two more fields. One of the fields is the "referer" HTTP request header which writes to the log where the client was referred from. The User-Agent HTTP request header is the other field added with `combined`. It adds information about the browser the client is using. The new configuration file must be enabled for it to work. By using the command `a2ensite` the new configuration file for the site will be enabled.

```
DocumentRoot /var/www/html
LogLevel warn
ErrorLog \${APACHE_LOG_DIR}/main-error.log
CustomLog \${APACHE_LOG_DIR}/main-access.log combined
```

Figure 10: Apache configuration file without `<VirtualHost>` directive

Now it is time to change the page that is displayed when accessing the IP address of the server. By default the `index.html` page from `/var/www/html` directory is displayed. Pages that are displayed by web servers are written in HTML code. Figure 11 shows an example of HTML code that has a title (title defines a title in the browser toolbar and it displays a title for the page in search-engine results), a header and some links to other pages. When adding applications later, the URLs for the applications will be `/apps/app1/` and `apps/app2/` accordingly.

```html
<html>
  <head>
    <title>Main page</title>
  </head>
  <body>
    <h1>This is the main page.</h1>
    <p>Link to Roundcube: <a href="/apps/app1/">click</a></p>
    <p>Link to MediaDrop: <a href="/apps/app2/">click</a></p>
  </body>
</html>
```

Figure 11: Example code of HTML to access different URLs

Preparing for the applications in next sections, it is necessary to make a new directory under `/var/www` directory. It can be named anything, but in this thesis it is named `vhosts`. Under the newly created directory two new directories have to be made. There the files for the applications - Roundcube and MediaDrop - respectively will be stored.

```
root@raspberrypi:~# service apache2 restart
root@raspberrypi:~# apachectl restart
```

Figure 12: Commands to restart Apache server

After changing the configuration files, enabling and disabling sites, it is mandatory to restart Apache server. In figure 12 different commands are shown on how to restart Apache server.

4.1.2 Roundcube Webmail

Roundcube is a "a browser-based multilingual IMAP client with an application-like user interface" that is written in PHP [25]. For it to work, a mail server is needed.

In the practical part of this thesis, Postfix was used, but the implementation is out of the scope of this thesis. Postfix is a mail server that uses Simple Mail Transfer Protocol (SMTP) [26]. The moving of emails from one mail server to another is done by SMTP. Since Roundcube uses Internet Message Access Protocol (IMAP) to work, an IMAP email server is also needed. For this reason, Dovecot was used. Dovecot is an "IMAP and POP3 email server for Linux/UNIX-like system, written with security primarily in mind" [27]. IMAP is responsible for getting emails from servers over a TCP/IP connection. Because Roundcube is written in PHP, the `php5` module has to be enabled in Apache server. MySQL server and MySQL client packages are also needed. In addition, the package named `php5-mysql` is also mandatory because it ensures that database connections can be done directly from PHP scripts.
Inside the `/var/www/vhosts/app1` directory the package for Roundcube can be downloaded. All of the files should have ownership of the user who is running Apache server (usually `www-data`). Roundcube also needs a MySQL database with a user who has all privileges regarding the database. After creating the database, initial data has to be inserted into the database. The command for doing this can be seen in figure 13.

```
root@raspberrypi:~# mysql roundcube <
 ↪  /var/www/vhosts/app1/SQL/mysql.initial.sql
```

Figure 13: Command to load initial data into database `roundcube`

The database, the user who can access the database and his/hers password have to be specified in the configuration file for Roundcube. The sample configuration file is under the `config` folder inside `/var/www/vhosts/app1` folder. The setting to be changed is called `$config['db_dsnw']`. Furthermore, the host, mail domain and SMTP server have to be specified in the same file (figure 14).

```
$config['default_host'] = 'mail.raspberrypi.test';
$config['smtp_server'] = 'mail.raspberrypi.test';
$config['mail_domain'] = 'raspberrypi.test';
```

Figure 14: Settings inside Roundcube configuration file to be changed

In Apache server it is also necessary to specify how to process requests made for URL `/apps/app1`. This can be done using the directive `Alias`. Moreover, it is necessary to specify how Apache should process files with the extension `.php` inside the directory for Roundcube and that everybody has access into the aforementioned directory. The example of these specifications is in figure 15. Basic directories - configuration directory, temporary directory and logs directory - should be protected however. For the changes to take effect, Apache server should be restarted.

```
Alias /apps/app1 /var/www/vhosts/app1
<Directory /var/www/vhosts/app1>
        Options +FollowSymLinks
        DirectoryIndex index.php
        <IfModule mod_php5.c>
                AddType application/x-httpd-php .php
                php_flag magic_quotes_gpc Off
                php_flag track_vars On
                php_flag register_globals Off
        </IfModule>
        AllowOverride All
        Require all granted
</Directory>
```

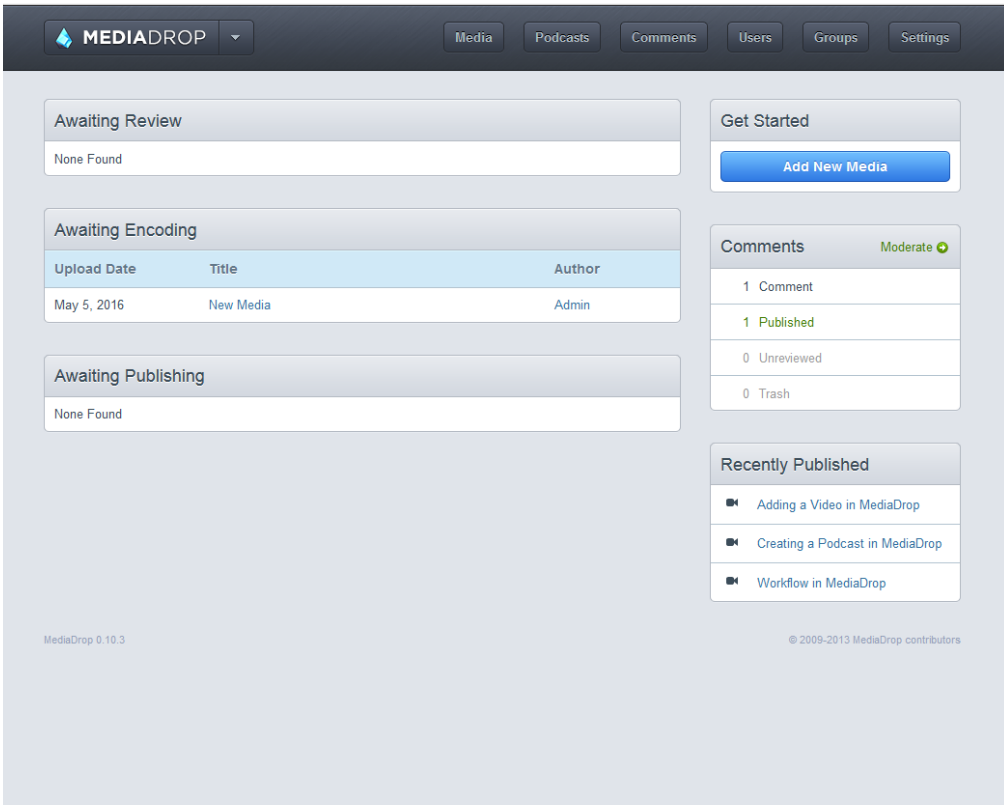
Figure 15: Example configuration for Roundcube in Apache configuration file

4.1.3 MediaDrop Video Platform

MediaDrop is "a media-oriented content manager" [28] where users can upload videos and podcasts, comment on them, like or dislike them. The site also shows views for every video. Furthermore, there is an administrator panel as seen in figure 16 where the administrator can review new uploaded videos before they are made public for users to view, make sure they are encoded right, review users' comments, make groups, add users to new groups, etc.

To install MediaDrop, the system has to meet the requirements which are as follows [28]:

- Python 2.x (2.4.x or newer),
- MySQL server (packages mysql-server and mysql-client) with newly created database for MediaDrop and a user who has all privileges regarding the database,
- GCC,



Awaiting Review

None Found

Awaiting Encoding

Upload Date	Title	Author
May 5, 2016	New Media	Admin

Awaiting Publishing

None Found

Get Started

Comments

1 Comment
1 Published
0 Unreviewed
0 Trash

Recently Published

Adding a Video in MediaDrop
Creating a Podcast in MediaDrop
Workflow in MediaDrop

MediaDrop 0.10.3

© 2009-2013 MediaDrop contributors

Figure 16: MediaDrop Administrator Panel

- development headers for Python (package named python-dev),
- development headers for MySQL client (package named libmysqlclient-dev),
- development headers for packages named libjpeg, zlib and libfreetype6,
- setuptools (package named python-setuptools),
- virtual environment creator (package named python-virtualenv).

Following the installation steps from MediaDrop documentation [28], a virtual environment has to be created and activated. The command on the first line in figure 17 creates a virtual environment named `venv` in a working directory and the command on the second line in the same figure activates the virtual environment.

After activating the virtual environment, the official release of MediaDrop can be downloaded from MediaDrop site. Since it is a file with an extension .tar.gz, it has to be unpacked with the command `tar`, which creates a new directory inside

```
root@raspberrypi:/var/www/vhosts/app2# virtualenv --distribute
 ↪  --no-site-packages venv
root@raspberrypi:/var/www/vhosts/app2# source venv/bin/activate
```

Figure 17: Creating and activating Python virtual environment

/var/www/vhosts/app2 directory named MediaCore-0.10.3. When inside this directory, the command in figure 18 installs all the necessary dependencies inside the virtual environment.

```
(venv) root@raspberrypi:/var/www/vhosts/app2/MediaCore-0.10.3#
 ↪  python setup.py develop
```

Figure 18: Command to install dependancies for MediaDrop

It might take a while because there are quite many dependencies to be installed. When the installation is completed, it is time to create a configuration file which contains basic MediaDrop settings. It is possible to create this file by running the command in figure 19 in the main directory for application two which is /var/www/vhosts/app2. Inside the newly created file called deployment.ini some settings have to be changed according to the system. The name of the database, username and a password for the user that has all privileges regarding the database for MediaDrop have to be specified by changing the setting sqlalchemy.url.

```
(venv) root@raspberrypi:/var/www/vhosts/app2# paster make-config
 ↪  MediaCore deployment.ini
```

Figure 19: Command to create a basic configuration file for MediaDrop

After finishing changing the settings in deployment.ini file, the folder containing the file content has to be copied next to it. The file content is located in a directory called data inside the directory MediaCore-0.10.3. Now everything is ready and the initial data is ready to be loaded into newly created tables in the database for MediaDrop by running the command in figure 20. It has to be noted as well that the files inside /var/www/vhosts/app2 folder should have the ownership of the user who is running Apache server. Usually the user is called www-data.

```
(venv) root@raspberrypi:/var/www/vhosts/app2# paster setup-app
↪  deployment.ini
```

Figure 20: Command to load initial data for MediaDrop

```
deployment_config = '/var/www/vhosts/app2/deployment.ini'
temp_dir = '/var/www/vhosts/app2/data/temp'

import os
os.environ['TMPDIR'] = temp_dir

if __name__.startswith('_mod_wsgi_'):
  from paste.script.util.logging_config import fileConfig
  fileConfig(deployment_config)
  from paste.deploy import loadapp
  application = loadapp('config:'+deployment_config)
```

Figure 21: `mediadrop.wsgi` file

The file system for the web site is now configured, so it is time to configure the application in Apache server as well. For this the module `wsgi` is needed. It can be installed with the command `apt-get` and the package name is `libapache2-mod-wsgi`. After that the module `wsgi` has to be enabled. Modules in Apache can be enabled and disabled the same way as site configurations. The commands are `a2enmod` and `a2dismod` respectively. Moreover, to run an application as a WSGI application a file with the extension `.wsgi`, where the deployment configuration is set, is needed. The example file of this is located in the GitHub page of MediaDrop. It must be downloaded and modified according to the system in use and it should be located in the directory `/var/www/vhosts/app2` next to `deployment.ini`. Figure 21 shows the file for the system used for this thesis.

In the configuration file for Apache that the user made earlier the code in figure 22 is also needed so that Apache understands that all requests made to the URL `/apps/app2/` have to be passed to the `mediadrop.wsgi` file so the requests will be processed through a WSGI application. Other directives in figure 22 define from where different data should be fetched when accessing a specific URL (`Alias` directive) and that some directories can be accessed by all (`Directory` directive and `Allow` directive). After changing the configuration file, a restart is required so

```
WSGIDaemonProcess mcore \
    processes=2 \
    threads=1 \
    display-name=%{GROUP} \ python-path=/var/www/vhosts/app2/venv/lib/python2.6/site-packages \ python-eggs=/var/www/vhosts/app2/data/python-egg-cache

WSGIProcessGroup mcore
# Intercept all requests to /apps/app2/* and pass them to mediadrop.wsgi
WSGIScriptAlias /apps/app2 /var/www/vhosts/app2/mediadrop.wsgi
# Create an exception for media and podcast image from your data directory
AliasMatch /apps/app2/images/(media|podcasts)(.*) /var/www/vhosts/app2/data/images/$1$2
# Create an exception for all static mediadrop content
AliasMatch /apps/app2/(admin/)?(images|scripts|styles)(.*) /var/www/vhosts/app2/MediaCore-0.10.3/mediacore/public/$1$2$3
# Create an exception for your custom appearance css and images
Alias /apps/app2/appearance /var/www/vhosts/app2/data/appearance
# Make all the static content accessible
<Directory /var/www/vhosts/app2/MediaCore-0.10.3/mediacore/public/*>
    Order allow,deny
    Allow from all
    Options -Indexes
</Directory>
```

Figure 22: WSGI directives for configuration file in Apache

the changes would take effect. The different commands to restart Apache server are listed in figure 12.

4.1.4 HTTP Authentication

Authentication is any process by which the user verifies that he/she is who they claim they are. HTTP authentication is possible with Apache server in many ways. There is, for example, a utility called `htpasswd` that comes with Apache server [29]. It is used to create files that store usernames and passwords for authentication.

The downside for this is that every user in the system has to be added to the file to be granted access. In this thesis, however, Pluggable Authentication Module (PAM) is used because it reads users and their password from system files. This means that every user that has an account on the system can access a location or directory if they have been granted permission. Firstly, the authentication modules have to be downloaded as packages with the command `apt-get` as seen in figure 23. After the download has finished, the module `authnz_external` has to be enabled in Apache server.

```
root@raspberrypi:~# apt-get install libapache2-mod-authnz-external
↪  pwauth
```

Figure 23: Command to download packages for authentication with PAM

Moreover, since the authentication will take place when a user accesses a certain URL, the modification of the configuration file in Apache is also necessary. The `<Location>` directive in Apache "applies the enclosed directives only to matching URLs" [24]. Therefore, if a user wants to view information under a certain URL, the `<Location>` directive is exactly needed. The code in figure 24 makes sure authentication is in place to access the URL `/apps` and everything under it. Because PAM authentication in Apache is an external method of authentication, it needs to be specified with `AddExternalAuth` and `SetExternalAuthMethod` directives. The `<IfModule>` directive checks if the module is enabled or not. If it is, it processes the enclosed directives. Since the configuration file for a site was changed, Apache server needs to be restarted with one of the commands from figure 12.

When accessing the protected URL, a box appears asking for a username and a password as seen in figure 25. After inserting the right username and password, the user is granted access to the web page under the specific URL.

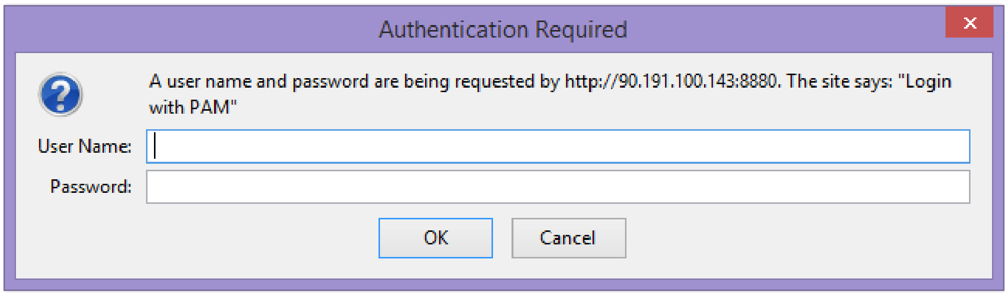
4.2 Setup with three Raspberry Pi computers

The setup for moving to three Raspberry Pi computers begins the same was as for one Raspberry Pi computer. For the new computers micro SD cards have to be formatted and made bootable. After booting the computers, they have to be configured with Raspberry Pi Software Software Configuration Tool as in figure 4. When the initial configuration is completed, the firewall has to be also configured with `iptables` command like in figure 6. With `apt-get` command Apache server can be installed (figure 7) and now the new nodes are both ready to host an application.

```
<IfModule mod_authnz_external.c>
  AddExternalAuth pwauth /usr/sbin/pwauth
  SetExternalAuthMethod pwauth pipe
</IfModule>

<Location /apps>
  AuthType Basic
  AuthName "Login with PAM"
  AuthBasicProvider external
  AuthExternal pwauth
  Require valid-user
</Location>
```

Figure 24: Apache configuration to enable authentication for a specific URL


Authentication Required

A user name and password are being requested by http://90.191.100.143:8880. The site says: "Login with PAM"

User Name:

Password:

Figure 25: HTTP Authentication with PAM

4.2.1 Main node configuration

The main node will be the Raspberry Pi that was used to host all of the applications in section 4.1. Some changes are required to be made in order to host the applications under new nodes. In this thesis the main node is made a proxy server to pass requests for applications to other servers. The HTTP Authentication will be left in the main node.
To make Apache server a proxy server, two modules are needed - `proxy` and `proxy_http`. After enabling them, the configuration file for the web site has to be changed as well. Figure 26 shows the changes made in the configuration file used for this thesis. All of the specifications for the applications from section 4.1 have to be removed. The HTTP authentication in the configuration file should not be changed.

```
ProxyRequests Off
ProxyVia On

# Roundcube
ProxyPass /apps/app1/ http://10.11.12.17/
ProxyPassReverse /apps/app1/ http://10.11.12.17/

# MediaDrop
ProxyPass /apps/app2/ http://10.11.12.18/
ProxyPassReverse /apps/app2/ http://10.11.12.18/
```

Figure 26: Directives added into Apache configuration to make the server a proxy server

The directive `ProxyRequests` determines if forward proxy requests are enabled or not [2]. The directive `ProxyVia` decides if information is provided in HTTP response header about the proxied requests. If this is set to `On` a line is added to the HTTP response header about the current host [2]. The directive `ProxyPass` "maps remote servers into local server URL-space" which means that it is "a mirror of the remote server" [2]. Another way to write `ProxyPass /apps/app2/ http://10.11.12.18/` is seen in figure 27.

```
<Location /apps/app2/>
        ProxyPass http://10.11.12.18/
</Location>
```

Figure 27: Another way to use directive `ProxyPass` [2]

The directive `ProxyPassReverse` makes sure redirects from the remote server which is behind a proxy are adjusted "before forwarding the HTTP redirect response to the client" [2].

4.2.2 Roundcube Webmail

The installation of Roundcube is the same as for one node in section 4.1.2. Some settings have to be changed, like the connection to the database. The same packages have to be downloaded as well. They are `php5`, `php5-mysql`, `mysql-server` and `mysql-client`. After enabling the module `php5` in Apache and adding the

Roundcube configuration into Apache configuration file, the application should be working correctly after restarting Apache server.

4.2.3 MediaDrop Video Platform

The steps to make MediaDrop work on one node are similar to getting it work on the main node next to a PHP application. Secure copy can be used to copy all the files for MediaDrop from the main node into another node. All of the files for MediaDrop are located in `/var/www/vhosts/app2` directory and they are being copied into the directory `/var/www/html` in figure 28. Also it should be checked that the permissions of the files and directories are correct and the user who is running Apache (usually `www-data`) can read, write and execute the files.

```
root@raspberrypi:~# scp -r /var/www/vhosts/app2
↪  root@raspberrypi3:/var/www/html
```

Figure 28: Secure copy command to copy files from one server to another

Since the node is brand new, all of the requirements for MediaDrop have to be installed as well. The requirements were specified in section 4.1.3. A new database has to be created in MySQL and a user who has all privileges regarding the database. Moving of the data from the old database into the new one is quite easy. In the main node `mysqldump` command has to be used to make a copy of the data from the database called "mediadrop" where all of the files are stored. Using secure copy the file `mediadrop.sql`, which holds the database data, has to be moved into the new node. Inserting the data inside the database in the new node is also easy. The command is seen in figure 30 - the data from the file `mediadrop.sql` is inserted into a database named "mediadrop" in the new node.

```
root@raspberrypi:~# mysqldump -p -u root mediadrop > mediadrop.sql
```

Figure 29: Command to create a copy of the data from a database into a file

```
root@raspberrypi3:~# mysql -p -u root mediadrop < mediadrop.sql
```

Figure 30: Command to insert data from a file into a database

Now that the data is ready, it is time to make sure all of the packages in the virtual environment are present. After activating the virtual environment `venv`, the command in figure 18 installs all the packages that might be missing from the virtual environment. After that inside the `deployment.ini` file the `sqlalchemy.url` has to be checked but it should be correct when the file was copied from the main node. Since proxy is being used, inside the same file the setting `proxy_prefix` has to be uncommented and set to `/apps/app2/`. Otherwise, the application will not work. Then the file system is ready, but it is advised to run the command in figure 20.

The file system is now ready and it is time to configure Apache server. The code from figure 22 has to be moved from the main node Apache configuration file into the new node's configuration file. Furthermore, the module `wsgi` has to be downloaded and enabled like in section 4.1.3 but on the new node. The `mediadrop.wsgi` has to be changed to match the new locations of deployment and temporary directories. After changing the configuration file, a restart for Apache server is required. The different commands are listed in figure 12.

5 Results

Raspberry Pi computers are known for their small size and their low power consumption. In this section, a comparison between the operational costs for a Raspberry Pi computer and renting a virtual server are explained. Furthermore, Apache Benchmarking tool is used to determine if Raspberry Pi computers are actually capable of serving small-to-medium-scale web pages.

5.1 Operational Costs

The typical bare-board active current consumption for Raspberry Pi 2 Model B is 330 mA [30]. Since the model is powered by 5V micro-USB, Rasbperry Pi 2 Model B uses 0,00165 kW. Servers typically are powered on 24 hours per day and 365 days per year, which means that Raspberry Pi 2 Model B uses 14,454 kWh per year. When comparing electricity prices in Estonia, it was noted that for a year the electricity bill for Raspberry Pi 2 Model B would be approximately 20€ [31]. Furthermore, the cost of one Raspberry Pi 3 in Estonia is around 45€ [32] [33]. The Raspberry Pi 2 computers used in this thesis are an older version than the Raspberry Pi 3 and therefore, they might be even cheaper. When also adding the price of the micro SD card and the power adapter, the price for the Raspberry Pi would be around 60€. Moreover, the size of Raspberry Pi computers is the same as for a credit card and thus, it would take very little room.

On the other hand, virtual machines or cloud machines would be a great alternative for Raspberry Pi computers. Raspberry Pi 2 has 1 GB of RAM and a quad-core CPU [20], therefore, a selection of different server providers are considered with similar indicators. For example, general purpose virtual machines in Azure have 1 core, 0.75 GB of RAM AND 20 GB of memory. In North Europe, this service would cost approximately 13€ per month [34]. DigitalOcean's plan with 1 core, 1 GB of RAM and 30 GB of memory costs 8,8€ per month (10$ per month) [35]. In Estonia a hosting service Zone.ee asks 18,29€ per month for 1 core, 1 GiB of RAM and 50 GiB virtual private server [36]. All of the prices here are without tax and a comparison table can be seen in table 1. For a yearly period Azure's virtual server would cost 156€, DigitalOcean's virtual server would cost 105,6€ and Zone.ee's virtual server would cost 219,48€.

In conclusion, a year of using Raspberry Pi as a dedicated web server would be much cheaper than paying for a virtual server in Azure, DigitalOcean or Zone.ee. Even when including the price of buying a Raspberry Pi 2/3, the total cost would only be 85€ per year which makes a 20€ difference in the worst way.

Table 1: Different hosting services and their prices per month [34] [35] [36]

	Azure	DigitalOcean	Zone.ee
Memory	20 GB	30GB	50 GiB
RAM	0,75 GB	1 GB	1 GiB
Cores	1	1	1
Price per month	13€	8,8€	18,29€

5.2 Apache HTTP Server Benchmarking Tool

For benchmarking a tool called Apache HTTP Server Benchmarking Tool was used. It shows "how many requests per second your Apache installation is capable of serving" [37]. The benchmarks were run each for three times - all with arguments 1000 requests and 100 concurrency - and then an average was calculated.

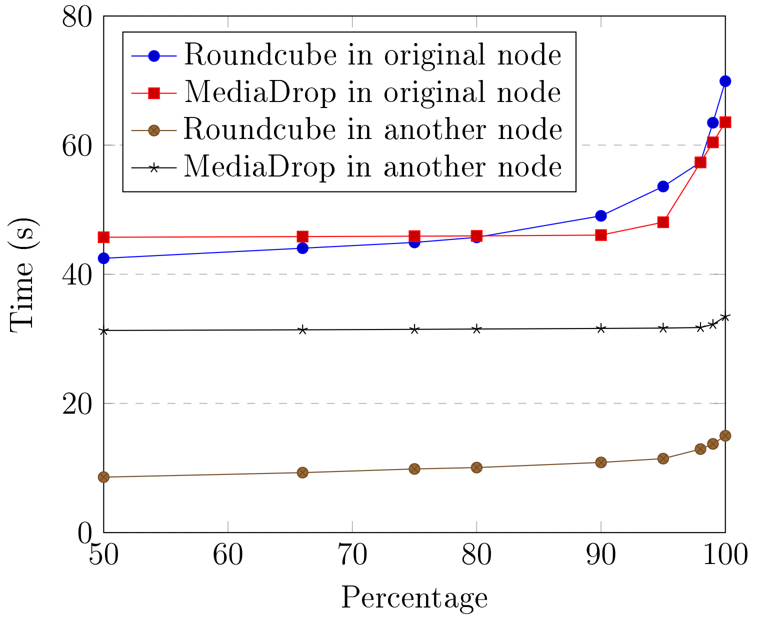


Figure 31: Percentage of the requests served within a certain time (s)

First, benchmarks were run on one Raspberry Pi computer which hosted two applications, the main page and HTTP authentication. Then the same tool was used with the same parameters when the applications were on different nodes. Figure 31 shows the results when requesting both applications on main node and when requesting both applications on different nodes. It can be seen that when applications were located on a different node than the main node, the request time was smaller. When testing Roundcube on both times, no requests failed. However, none of the requests failed for MediaDrop application when it was located on the main node. When it was on a different node, approximately 26 requests out of a 1000 failed. It can be said that some applications are good to use on a Raspberry Pi as a server for a small-to-medium enterprise. Others might take a while to load, but they can still be used if speed is not a priority. `ProxyPass` definitely made things faster than when everything - both applications and HTTP authentication - were on the same node.

6 Conclusion

Different aspects of using Raspberry Pi computers as web servers were looked at in this thesis. It was noted that using one Raspberry Pi as a web server for a year would be cheaper than renting a virtual private server for the same amount of time. If the Raspberry Pi computers were to be used for years, the difference in money saving would be even greater. Moreover, different setups of Raspberry Pi computers were used and it was found that when using three Raspberry Pi computers with two of them serving different applications, the speed increased compared to one Raspberry Pi hosting the same two applications.
In the future, a bigger setup of Raspberry Pi computers can be tested. Applications written in other programming languages can also be considered. Using more than one application per node in a Raspberry Pi cluster is also an opportunity for future work and a real-life testing setup for a small company should be done with a Raspberry Pi cluster.

7 Üleminek virtuaalkeskkonnast Raspberry Pi arvutitest koosnevale füüsilisele infrastruktuurile

Bakalaureusetöö(9 EAP)
Annika Laumets
Resümee

Püstikuservereid on kasutatud juba mitu aastakümmet. Lisaks kasutatakse ka privaatservereid, mis on eksklusiivselt renditud arvutid veebiserveri, vajaliku tarkvaraga ja Interneti-ühendusega. Neid kasutavad just väikefirmad. Privaatserveri rent või püstikuserveri omamine võib aga väikefirmale üle jõu käia. Energiaressursid hakkavad samuti vaikselt otsakorrale saama ning kuna neist saab elektrit, siis oleks mõistlikum elektrit kokku hoida. Kuna püstikuserverid kasutavad palju elektrit ning pool ajast ei kasutata nende täit potentsiaali, siis oleks nende välja vahetamine suurepärane võimalus hoida raha kokku tegevuskulude ja elektriarve osas. Raspberry Pi arvutid on väikse elektrikuluga laialt levinud arvutid, mistõttu oleksid need ideaalsed asendama püstikuservereid. Selles töös proovitakse üles seada veebiserver ühel Raspberry Pi arvutil, millel jookseb kaks erinevat rakendust, ning seejärel see lahendus viia üle kolmele Raspberry Pi arvutile nii, et rakendused asuksid erinevatel arvutitel.
Siin töös leiti, et Raspberry Pi arvuti kasutamine ühe aasta jooksul tuleks odavam kui samalaadse privaatserveri rentimine sama ajaperioodi jooksul. Erinevates programmeerimiskeeltes kirjutatud rakendused töötasid ühel Raspberry Pi arvutil hästi, ent päringud muutusid kiiremaks, kui kasutati kolme Raspberry Pi arvutit, millest üks oli puhverarvuti. Mõndadel päringutel esines tõrge, kui tehti jõudlusteste MediaDropi rakendusel kasutades puhverserverit. Ilma puhverserverita (ühel Raspberry Pi arvutil töötav lahendus) ei esinenud kummagi rakenduse jõudlustestide ajal tõrkeid. Kokkuvõttes võib öelda, et väikefirmad hoiaksid oma tegevuskulud madalamal ning säästaksid loodust kasutades vähem energiat, kui nad kasutaksid Raspberry Pi arvutitele ehitatud servereid.

References

[1] “Raspberry Pi - Configuration.” https://www.raspberrypi.org/documentation/configuration/raspi-config.md. Accessed: 2016-05-07.

[2] “Apache Module proxy.” https://httpd.apache.org/docs/current/mod/mod_proxy.html. Accessed: 2016-05-11.

[3] L. A. Barroso and U. Hölzle, “The case for energy-proportional computing,” *Computer*, vol. 40, pp. 33–37, Dec 2007.

[4] “Report to congress on server and data center energy efficiency.” Public Law 109-431, August 2 2007. U.S. Environmental Protection Agency ENERGY STAR Program.

[5] M. Olson, K. Christensen, S. Lee, and J. Yun, “Hybrid web server: Traffic analysis and prototype,” in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pp. 131–134, Oct 2011.

[6] B. Varghese, N. Carlsson, G. Jourjon, A. Mahanti, and P. Shenoy, “Greening web servers: A case for ultra low-power web servers,” in *Green Computing Conference (IGCC), 2014 International*, pp. 1–8, Nov 2014.

[7] “Apache HTTP Server Project.” https://httpd.apache.org/. Accessed: 2016-04-02.

[8] “Httpd Wiki.” http://wiki.apache.org/httpd/FAQ#Why_the_name_.22Apache.22.3F. Accessed: 2016-04-02.

[9] “Apache HTTP Server Project - About.” https://httpd.apache.org/ABOUT_APACHE.html. Accessed: 2016-04-02.

[10] “Apache Software Foundation - Projects.” http://www.apache.org/index.html#projects-list. Accessed: 2016-04-02.

[11] “Raspberry Pi - What is Raspberry Pi.” https://www.raspberrypi.org/help/what-is-a-raspberry-pi/. Accessed: 2016-04-04.

[12] “Raspberry Pi - About.” https://www.raspberrypi.org/about/. Accessed: 2016-04-04.

[13] “PHP.” https://secure.php.net/manual/en/preface.php. Accessed: 2016-05-06.

[14] "PHP History." http://php.net/manual/en/history.php.php. Accessed: 2016-05-06.

[15] "Tiobe Index." http://www.tiobe.com/tiobe_index?page=index. Accessed: 2016-04-18.

[16] "Pypl Index." http://pypl.github.io/PYPL.html. Accessed: 2016-04-18.

[17] "Python." https://docs.python.org/3/faq/general.html#what-is-python. Accessed: 2016-04-18.

[18] "History of Python." https://docs.python.org/3/license.html#history-of-the-software. Accessed: 2016-04-18.

[19] "Python in the web." https://docs.python.org/2.7/howto/webservers.html. Accessed: 2016-05-02.

[20] "Raspberry Pi 2 Model B." https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. Accessed: 2016-05-07.

[21] "Raspberry Pi 3 Model B." https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed: 2016-05-07.

[22] "Raspbian Operating System." https://www.raspberrypi.org/downloads/raspbian/. Accessed: 2016-05-07.

[23] "Debian Jessie package Apache2." https://packages.debian.org/jessie/apache2. Accessed: 2016-05-08.

[24] "Apache Core Features." httpd.apache.org/docs/2.4/mod/core.html. Accessed: 2016-05-09.

[25] "Roundcube - Webmail Software." https://roundcube.net. Accessed: 2016-05-11.

[26] "Postfix Mail Server." http://www.postfix.org. Accessed: 2016-05-11.

[27] "Dovecot Secure IMAP Server." http://www.dovecot.org. Accessed: 2016-05-11.

[28] "MediaDrop documentation." http://mediadrop.net/docs/. Accessed: 2016-05-09.

[29] "htpasswd utility in apache." http://httpd.apache.org/docs/current/programs/htpasswd.html. Accessed: 2016-05-09.

[30] “Raspberry Pi - Frequently Asked Questions.” www.raspberrypi.org/help/faqs/. Accessed: 2016-05-10.

[31] “Eelektrihind - Comparison of different electricity packets in Estonia.” http://elektrihind.ee/paketid/. Accessed: 2016-05-10.

[32] “Raspberry pi 3 in ITTGroup.” http://www.ittgroup.ee/et/raspberry-pi-jt-miniarvutid/721-raspberry-pi-3-mudel-b.html. Accessed: 2016-05-10.

[33] “Raspberry pi 3 in Arvutitark.” http://www.arvutitark.ee/est/tootekataloog/Arvutikomponendid-Barebone422/Raspberry-Pi-3-mudel-B-208670. Accessed: 2016-05-10.

[34] “Virtual Hosting in Azure.” https://azure.microsoft.com/en-us/pricing/details/virtual-machines/. Accessed: 2016-05-10.

[35] “Virtual Hosting in DigitalOcean.” https://www.digitalocean.com/pricing/. Accessed: 2016-05-10.

[36] “Virtual Hosting in Zone.ee.” https://www.zone.ee/et/teenus/pilveserver/hinnad/. Accessed: 2016-05-10.

[37] “Apache http server benchmarking tool.” https://httpd.apache.org/docs/2.4/programs/ab.html. Accessed: 2016-05-11.