**Project Report**

**Purpose:**

Built a multi-purpose Raspberry Pi Cluster that serves as a webserver to deploy an educational website for the client for teaching, as well as a Virtual Private Network (VPN) server, to safely connect into the local network and the local network devices from anywhere in the world.

**Requirements:**

* Website availability to the students of Saint Martin’s University.
* Website needs to include 13 weeks of study materials
* Easily understandable documentation
* Website needs to be easily navigable
* Professor needs access to the server remotely

**Organization:**

Team:

* Sanju Byanjankar
* Joshua Schulz

Stakeholder:

* Dr. Xuguang Chen

**Engineering:**

Research Specifications:

* C Programming, HTML/CSS
* Parallel Programming with MPI and OpenMP
* Linux scripting in Raspbian Operating System
* Networking with Raspberry Pi
* Virtual Network Computing
* Virtual Private Network

Tools and Resources to be used for design:

Hardware:

* 3 Raspberry Pi 3 Model B+
* 16GB SD card per Raspberry Pi
* Power supply
* 3 Layer raspberry pi case
* HDMI cables
* Ethernet cables
* 5-port Network Switch
* USB keyboard
* USB mouse
* Monitor or TV with HDMI output

Software:

* HTML/CSS languages are used for front-end software design
* Putty used for remote desktop connection in Local Area Network
* Virtual Network Computing (VNC) Connect is used for remotely connecting with servers.
* piVPN is installed on the server and OpenVPN is installed on the remote client, to create an encrypted connection between the remote device and the Local Area Network that the server is on.
* Haproxy is used as a load balancer

**Starting up Raspberry Pi Cluster:**

1. Downloading the Raspberry pi Operating system

* Browse <https://www.raspberrypi.org/downloads/>
* Download the Latest version of NOOBS on to your computer

1. Installing the operating system on a micro SD card

* Before installing the Operating system, we need to format the SD card first to use FAT32 filesystem.
* Insert SD card into SD card adapter and plug it into the computer.
* Browse <https://www.sdcard.org/downloads/formatter/> . Download and install a free software for SD card formatting compatible with your device.
* Run the SD card formatter software, and choose the correct drive to be formatted. I repeat. Be careful and choose the right drive. Do a complete format.
* Locate the NOOBS zip file you downloaded, right click and extract it to the SD card. Make sure it’s not a single folder. If it is then, copy and paste all the files and folders from the NOOBS folder into the SD card.

1. Starting it up

* Plug in all the SD cards into each of the raspberry pis.
* Power up one of your raspberry pis. Plug in keyboard and mouse into the USB port, and plug in the monitor via HDMI cable. We have to configure each of the raspberry pis individually since we probably have only one keyboard, mouse and monitor. Feel free to do it all at once if you have enough equipment.
* I used 5 port Network Switch to connect to the internet, which is connected to my router on the other side. You can directly connect your raspberry pis to the router using network cable.
* There is no on and off power button on a raspberry pi, therefore as soon as you plug in the main power, it starts booting.
* Check to see if the pi has internet access by opening up the browser and going to your favorite website. From terminal, you can try “curl <http://www.google.com>
* Configuring a Raspberry pi is very easy. Make sure to change the Pi name and password for security reasons.

**Setting up Raspberry Pi Web Server:**

Configuring each raspberry pi to have a static ip address is a necessity for it to work as a web server. When a user browse a website, your router should be able to direct the traffic to the correct server. For this reason, the ip address has to be static, and dynamic ip address may cause issues with browsing the web page. In short, assigning static ip addresses to your raspberry pis means that you’ll always know exactly where to find them on your network.

First of all, make sure you have the latest version of Raspbian. You can do that by opening your terminal and typing “cat /etc/network/interfaces”. You’ see the following output.

# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd

# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:

source-directory /etc/network/interfaces.d

auto lo

iface lo inet loopback

iface eth0 inet manual

allow-hotplug wlan0

iface wlan0 inet manual

wpa-conf /etc/wpa\_supplicant/wpa\_supplicant.conf

allow-hotplug wlan1

iface wlan1 inet manual

wpa-conf /etc/wpa\_supplicant/wpa\_supplicant.conf

As long as “iface etho0 inet manual” is there, you are good to go. If not upgrade to a newer version of Raspbian. Now, to configure static ip addresses we need to get the ip addresses of the raspberry pis from the network.

In terminal, type “netstat -nr” and write down the Gateway address. In my case, my gateway address or router’s address is 192.168.0.1. Use your favorite text editor to open the dhcpcd.conf file. I used nano which is very simple and it does come pre-installed in raspberry pi, but you also can use vim, Gedit and many more. Type “nano /etc/dhcpcd.conf” to open up the file and add the following lines at the bottom of the file. I used 192.168.0.200 for the first pi, 192.168.0.201 for the second and 192.168.0.202 for the third, so that it is very easy to remember.

interface eth0

static ip\_address=192.168.0.200

static routers=192.168.0.1

static domain\_name\_servers=192.168.0.254

Restart the pi by unplugging and replugging the main power or the best way is to go to the terminal and type “sudo reboot”.

If everything went well then you should be able to ssh into your pis from any computers on the same network. To verify, try “ssh pi@192.168.0.200 and it should prompt you to enter password for your raspberry pi. Now the question is, how to make secure shell connection to the raspberry pis without having to type in the password? We can accomplish this task by generating using rsa keys.

Steps to secure shell from one raspberry pi to another using rsa keys.

* In terminal, make sure you are in root directory. Type “ssh-keygen” to generate a new key and save in the default “.ssh” folder with no password.
* Jump into that directory by using “cd .ssh” command and type “cp id\_rsa.pub pi01” to copy the id\_rsa.pub into a new file “pi01”. This will keep things organized.
* Now, ssh into the second raspberry pi, i.e. 192.168.201, and do the same thing. Generate a key and save it in a new file pi02. While on /.ssh directory, type “scp 192.168.0.200:/home/pi/.ssh/pi01” to copy pi01 file where the key is saved in first raspberry pi into the raspberry pi you currently in i.e. 192.168.0.201. Add pi01 to the authorized keys list. Type “cat pi01>>authorized\_keys. Exit out of current raspberry pi and switch into the first raspberry pi.
* At this point ssh into the third pi i.e. 192.168.0.201 and repeat the steps from the beginning.
* Final step is to jump to the first pi and copy files pi02 and pi03 into .ssh folder of the current pi. Use “scp 192.168.0.201:/home/pi/.ssh/pi02” and “scp 192.168.0.201:/home/pi/.ssh/pi03” to copy. Use “cat pi02>>authorized\_keys” and “cat pi03>>authorized\_keys” to add them into the authorized keys list of the first raspberry pi.

If you did everything correctly up till here, you should be able to ssh between the pis without any authentication using “ssh pi@192.168.0.201” command.

**Start a web server on Raspberry Pi.**

Create an html file using nano editor and give it a name of your choice. Save it with an extension .html, so that the server knows it’s a web file. I created a home page named index.html which is saved in the root folder of the pi. All the links from inside the home page is also saved in the same directory. For example: I have files named, chapter1.html, chapter2.html and so on.

Raspberry pi has very limited space. Therefore, next we are going to get a simple node server and make it run in all our raspberry pis. You can go to  <http://howtonode.org/hello-node> and copy the “hello world” example and custom configure it. I did it in the following way.

var http = require('http');

var express = require('express');

var app = express();

//express pointed to the root directory of the pi

app.use(express.static('/home/pi'));

var server = http.createServer(app);

//all raspberry pi servers must be listening to port 8000

server.listen(8000);

app.get('/', function (req, res) {

//making sure the server knows the html file format.

res.header('Content-type', 'text/html');

res.sendFile('index.html');

});

console.log("Server running at http://127.0.0.1:8000/");

Node webserver doesn’t have similar features as apache web server, and therefore I had to download “express” package to navigate through the files in the root directory.

Use nano text editor and copy the code into a file named basic\_node\_webserver.js

Type “nano basic\_node\_webserver.js” and paste it into the editor and save it. After that, starting uup is very easy. To run the node server use the command “node basic\_node\_server.js” and you should see the following.

$ node basic\_node\_webserver.js

Server running at http://127.0.0.1:8000/

Repeat the same for all other pis. Make sure they are running well before moving further.

You should be able to browse the server from all other devices that are on the same network. Go to your favorite browser, and browse “192.168.0.200:8000”, “192.168.0.201:8000”., “192.168.0.203:8000”. In that case, you should be able to open up the index.html file located in the root directory of the pi. In my case, all the files are just on the root directory. You can create a folder and put all the files in one for more organized work. Remember, all the files need to be on the same directory for easy navigation. This navigation is helped by the package we downloaded called express, which basically looks for files in the same directory on a user’s command. So, all three pis are responding to the requests to port 8000.

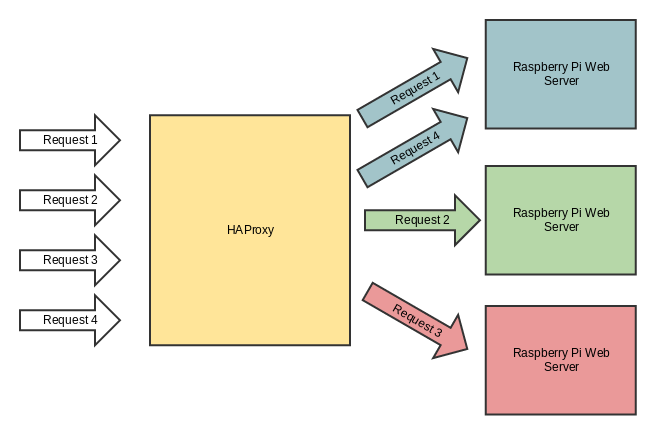
**Tying up all the raspberry pis with HAProxy:**

HAProxy stands for High Availability Proxy which is developed by HAProxy Technologies LLC. It is an open source load balancer proxy for TCP and HTTP applications.

Now, *why do we need raspberry pi cluster and haproxy necessary for website deployment?*

This is a legit question to ask. Why do we need multiple raspberry pis when a single raspberry pi is capable of listening and responding to the requests? Well, if this raspberry pi was actually a computer serving requests on the internet for your website, when your websites get popular you will encounter tremendous amount of traffic on your server and may not be able to handle. Therefore, it is better to have multiple computers serving the requests instead of one. It just like a division of labor in a certain company.

Adding more computers or raspberry pis to the network can improve or minimize the downtime of the website. Even if one or two servers goes down the other servers can still handle the requests. Also, HAProxy improves the performance of servers by distributing their workloads to multiple raspberry pis by minimizing response times and increasing throughput.



We will be using HAProxy to load balance incoming requests to a single IP address (website address), so that each request can go to one of the several Raspberry pi’s behind the router. Next step is to download and install HAProxy into one of our Raspberry Pi

Use command “sudo apt-get install haproxy” to download and install.

Use command “node basic\_node\_webserver.js” in all the raspberry pis.

Check to see if all the servers are running by browsing the ip addresses from another device on the same network. Don’t for get to include the port number 8000 after the IP addresses.

Configuring HAProxy:

You can choose any of the raspberry pis to install the HAProxy or even choose any computer on the same network at home. Follow the steps below to configure properly.

* Go to the raspberry pi with HAProxy installed
* We are going to edit the config file. So, let’s use nano editor to open haproxy config file.

Type “sudo nano /etc/haproxy/haproxy.cfg”

* Let’s edit the front-end and the back-end portion of the code.

frontend raspberrypinodes

bind \*:80

mode http

default\_backend nodes

backend nodes

mode http

balance leastconn

option forwardfor

http-request set-header X-Forwarded-Port %[dst\_port]

http-request add-header X-Forwarded-Proto https if { ssl\_fc }

option httpchk HEAD / HTTP/1.1\r\nHost:localhost

server raspberrypi01 192.168.0.200:8000 check

server raspberrypi02 192.168.0.201:8000 check

server raspberrypi03 192.168.0.202:8000 check

listen stats

bind \*:1936

stats enable

stats uri /

stats hide-version

stats auth someuser:password4321

Finally, it’s time to see it in action. Save the updates in the config file and get onto the pi with HAProxy installed. Type “ sudo service haproxy restart”.

Browse the IP address of the raspberry pi in your favorite browser from a different device in same network. Notice that it doesn’t show the port 8000 anymore. That is a default for port 80, which basically means that HAProxy is receiving request on port 80 and proxies it to port 8000 on one of the raspberry pis that is listening to port 8000.

If you try to refresh the page, it switches between the servers and that is because HAProxy is load balancing between the raspberry pis. In the back-end portion, instead of “leastconn” if I use “roundrobin”, the servers will basically take turns to respond to the request. The reason I chose leastconn is because it helps us to choose the pi with the least connection which will better optimize connections.

If you reached here and accomplished every thing till now then you deserve a Pie. We have built a Raspberry Pi Cluster Webserver. Staying focused again.

This is how it works in the large scale. When you browse a popular website, it hits the central web address and your request gets routed to one of the many servers.

The last portion in the config file is “listen stats” and is a a stats tracker. It is optional but since we added it in the config file we should be able to go to the port we specified and see the stats on each of our servers.

**Domain name and domain server**

**Vnc**

**Vpn**

**Monitoring and controls**

**Next Steps**

**Milestones:**



Risks:

* There is always a security risk.
* Server might not be able to handle excessive traffic because of the low RAM on each of the servers.

Process for managing risks:

* Paying attention to simple things will eliminate most of the security risks. For example, custom creating passwords and usernames, creating a password easy to remember but hard to guess, making sure the software is up-to-date, making sure the port forwarding is done properly and many more.
* Since, this project is not the commercial version, it cannot handle too much traffic. Raspberry Pi 3 Model B+ has a RAM of 1GB and is completely embedded into the chip. It’s impossible to increase the RAM size in Raspberry Pi but it can be optimized using a software