



**Abertay
University**

Raspberry Pi OpenMPI Password Cracking Cluster

Neirin McDonald

Jake Mills

Stuart Rankin

Ewan Scott

CMP311: Professional

Project Development & Delivery

Ethical Hacking

2019/20

Abstract

This white paper has been created to present the findings and method that team Space Hopper used to create a Raspberry Pi cluster utilizing the program John The Ripper to crack passwords. Dr Ethan Bayne, a staff member at Abertay University, was the client for this project. The project brief was to “create a *Raspberry Pi* network of *various models* which utilizes *OpenMPI* and *John The Ripper* to create a *heterogenous password cracking cluster*, then *measure its performance against a Hacklab workstation and/or GPU*” which is what we aimed to achieve on this project. We also aimed to create various scripts for the cluster.

This aim was met by splitting the project into 4 distinct sections. First section was the construction of the cluster, this was done by flashing all Raspberry Pis we were given with Raspbian Buster Lite¹ and then giving them all different static IPs and hostnames. Using an 8-port switch a small network was created for the cluster, the Pis were powered by a USB power hub. Then an 128GB USB drive was connected to the master node so that a Network File System (NFS) share was set up on the cluster, so that all nodes could access the same files. With all the nodes properly configured from here the cluster was controlled by a desktop computer through SSH. The programs SLURM and OpenMPI were then installed in this part of the project to allow the master node to properly control the cluster, this completed the construction of the project.

The next section involved installation of two important packages, Python and John The Ripper. Python was chosen as the scripting language due to the fact most students studying Ethical hacking at Abertay understand and know how to program in it. Installation was done by using “wget”, “tar” and the standard Linux “make, make install” to install John on the master node, Python was the same except “./configure” was used to install Python onto the NFS share so that all nodes could access its libraries without them having to install Python to the node's main storage.

The last 2 stages consisted of creating a benchmark script to measure the clusters effectiveness and then comparing these results to a hacklab computer. This had to be changed however due to COVID-19, as the University was closed and we couldn't access the workstations. Because of this Stuart Rankin's computer was used as a comparative benchmark. With the benchmarks done, conclusions could be drawn about the cluster.

The desktop was found to be faster than the cluster when cracking passwords, however this was no surprise as Stuart Rankin's computer is a modern desktop PC with a high clock and core CPU. But while the cluster isn't the fastest other positives were found:

- The cluster would work as a good showcase for Open days to potential new students.
- It could work around the clock to crack passwords while hacklab workstations need to be used for other uses.
- With SLURM the cluster is now very flexible and could be used to perform multiple different jobs at the same time.
- It could be repurposed relatively easily in the future.
- It used various pieces of old and unused pieces of hardware.

So, while it was found to not be the fastest password cracker in the future it could serve multiple other functions to the staff and students of Abertay University.

Contents

Introduction	4
Background	4
Aim	5
Procedure.....	7
Overview of Procedure	7
Results.....	10
Raspberry Pi cluster benchmark results	10
Discussion.....	12
General Discussion.....	12
Conclusions	13
Future Work.....	14
Call to action	15
References	16
Appendices.....	17
Appendix A - E-mails	17
Appendix B – Scripts	17
demo.py	17
johnDemoWordlist.sh	19
johnDemo.sh.....	19
benchmark.py	19
wordlistBenchmark.sh	21
johnTest.sh.....	21
Appendix C - Results	21
johnTest.sh Result.....	21
Comparison john --test Result.....	22
Appendix D - Burndown Chart	23
Appendix E Gantt Chart	24
Appendix F - Deliverables & requirements.....	25
Appendix G – Minutes.....	27
Appendix I - Manual.....	36

Introduction

Background

The limits of what a machine can do are continuously being pushed further and further, making computers faster and smaller. According to Moore's Law², "processor speeds, or overall processing power for computers will double about every 18 months" (University of Missouri-St. Louis, no date). Unfortunately, our progress comes with a downside. Due to creating new and faster computers, we discard hundreds of thousands of pounds worth of older computers every year, many of which are deposited in landfills increasing the amount of pollution which is bringing on disasters such as climate change. The average person in the UK creates 20-25kg of e-waste every year, and "43 million tons of electronic waste was generated in 2016" (Knapton, 2017)³. Not only is there such a large amount of e-waste produced, it is growing faster than any other type of refuse. These old computers simply become outdated and cannot compete with new better models, however they are still functional. By creating a cluster of these older machines, you can combine their power into something that can still compete with new machines, therefore negating the need to throw them away.

The date of the first cluster is unknown, but it was likely created either in the late 1950s or in the 1960s by consumers who needed more storage space as their home computers were not able to store all their files. It was later in 1967 when Gene Amdahl published a paper on parallel processing, where he explains that performing work in parallel will reduce the time taken to perform said work. In this paper he defined Amdahl's Law, which describes how much faster a task can be run depending on the number of processors used and what percentage of the task will benefit from being run in parallel.

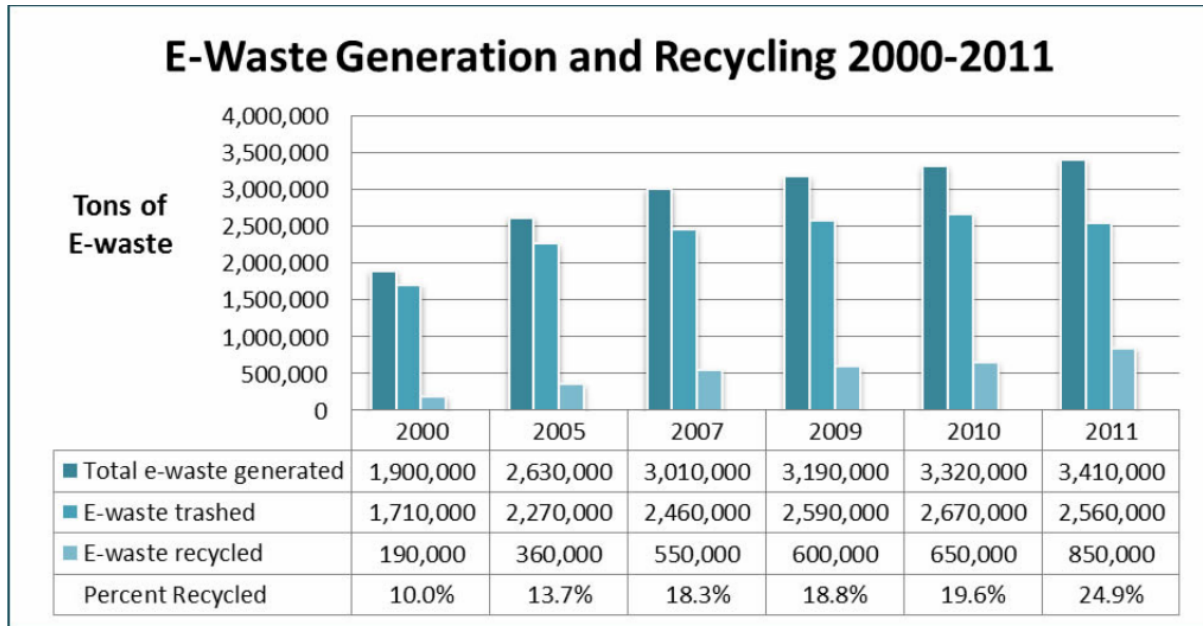
The development of early clusters is related to the development of early networks, since the use of these networks was to link different hardware together to link their resources, which effectively is creating a cluster. The earliest commodity-network based computer cluster that was developed is the ARPANET, which was created by linking four separate computers together. This eventually developed into the modern-day internet. The internet itself can be described as a gigantic cluster.

A single Raspberry Pi can in no way compete with a workstation PC, however there is strength in numbers. By creating an MPI cluster of Raspberry Pis, they will be able to have the power to compete with any machine. The size of the cluster can easily be modified to add more Raspberry Pis to increase the power of the cluster, the cluster will also have the ability to easily remove Raspberry Pis if needed. This cluster will utilise the password cracking software "John the Ripper" to be able to crack passwords at the same speed, or even faster, than a hacklab workstation PC.

An issue with the cluster is the use of space. Utilising multiple pieces of hardware will take up more space, which is unavoidable. For our cluster we will attempt to set up appropriate housing for the nodes in the cluster to maximise space efficiency and to use effective cable management.

An alternative solution that is currently being implemented is recycling plants. The problem with this approach is that it is not feasible to recycle everything and only 15.5% of e-waste is recycled properly globally (Vaute, 2018)⁴. There is far too much technology for it all to be recycled. Apple made a robot that can dismantle an entire iPhone in a few seconds, but

even that still is not capable of keeping up with the humongous workload that Apple provides for it. See figure 1 which shows how much e waste has been created from 2000 to 2011.



*Figure 1
showing the amount of e-waste trashed vs the amount of e-waste produced.*

Aim

The aim of this project is to create a working cluster of Raspberry Pis that can utilise John the Ripper to crack lists of password hashes in multiple different hash types. The cluster will have scalability, with inbuilt scripts that will allow the user to easily add/remove raspberry pis to and from the cluster. We aim to give our cluster a friendly, easy to use interface so that the cluster can be used at demonstrations to people with technical knowledge. We will create a user tutorial that will be built into the cluster to demonstrate the capabilities of the cluster and to show the user how they can utilise the cluster's functionalities. A user manual will also be created to give detailed instructions on how to use the cluster and how to resolve common problems that may be found while using the cluster.

There were several objectives for the project which are:

- To create a Raspberry Pi cluster
- To install John The Ripper and Python to the cluster
- Write various Python scripts which utilize John The Ripper to crack passwords and perform other functions.
- Benchmark the final cluster product against a hacklab computer.

All these objects had been researched in CMP308 (Professional Project Planning and Prototyping) thoroughly, this allowed the team to create a Gantt chart (see appendix E Gantt chart) which played into the team's strengths. The tasks are split in a way which gives everyone similar workloads but also plays into what the teammate is good at. The research also gave each team member a look at what it takes to create a cluster in terms of putting one together, installing and running the necessary programs and how to maintain a cluster.

This has given the team a lot of confidence that they know what to do when they need to complete a task on the Gantt chart.

Everyone in the project also has good technical knowledge, this is known as every team member is in 3rd year on Abertay's Ethical Hacking course. Technical knowledge of subjects such as networking, Linux OS's, John The Ripper and Python are needed to get to this point in the course, all of these subjects are used in this project. Because of these reasons the team knows that the projects aims/objectives can be met.

Procedure

Overview of Procedure

The next section explains how the team created and built the cluster for the project. A full manual explaining how to fully build and create this cluster was made during this project in case anyone wanted to work on this cluster in the future. This can be found in appendix I – manual.

Building the Cluster

Jake Mills was responsible for physically building the cluster with the equipment that was given to us. The first step taken was to flash the Pis with Raspbian Buster Lite. The image was written to the SD cards using the program “Etcher”.

Once the Raspberry Pis were all successfully flashed, it was time to create the network the cluster would be using. Firstly, SSH was enabled manually by using the command “raspi-config” on every Raspberry Pi. Each Pi was then set up individually one after another for the sake of convenience in recording each node’s IP address. Each Pis IP address was assigned manually by editing their configuration files and were recorded.

After assigning static IP addresses to each node it was time for basic configuration of each Pi, starting with the hostname. This was accomplished by using the “hostname” command and editing the “hostname” and “hosts” files of each Pi. The times of each node had to be synchronized to allow them to function together as a cluster so this was carried out by installing the “ntpdate” package on each Pi. A reboot was performed on each node afterwards.



Figure 2
All hardware pieces use in the cluster.

3. Shared Storage

In order for the Raspberry Pis to function together as a cluster they need to be able to access shared resources, allowing a file to be accessed by any node in the cluster. One of the nodes in the cluster was designated as the master node and a USB flash drive was connected to this node and mounted on it to be used for Network File System (NFS).

The flash drive was formatted to use the ext4 filesystem and a mount directory was created on the master, this was then mounted on and automatic mounting was enabled. The permissions set on the mounted drive were not needed to be strict since the cluster is well supervised.

Once the NFS was mounted on the master node, it was then needed to be exported to the rest of the cluster and was mounted on the compute nodes by installing the NFS client, creating a mount folder and setting up automatic mounting.

4. Installing and Configuring SLURM

To help run tasks on the cluster, SLURM, a scheduling program was installed onto the cluster. The SLURM control packages were installed onto the master node. The SLURM configuration file is edited so that the configuration file contains the hostname and the IP address of the master node. There are multiple methods for allocating resources in SLURM, however we wish to use the consumable resources method so that jobs are given out to nodes based on their CPU usage. To select this method, the "SelectType" field in the configuration file is edited. The information about all of the compute nodes are then inserted into the configuration file so that SLURM can allocate tasks to them. A partition needs to be created for the nodes. The example partition is deleted and a new one is inserted, and the nodes are added.

After editing the configuration file, SLURM needs to be configured to allow jobs to access resources. This is accomplished by creating a file called "cgroup.conf", which contains all of the resources. Then a whitelist file is created to allow devices in the cluster to access required resources. The configuration file, cgroup file and whitelist files are copied to the NFS to allow compute nodes to be controlled by SLURM, in addition to the Munge key file.

SLURM control services are enabled with the systemctl command and the cluster is rebooted.

Now that SLURM is set up on the master node, it needs to be configured on the compute nodes. The SLURM client is installed on the compute node and the hosts file is edited to include the hostnames and IP addresses of all of the other nodes in the cluster. The configuration file, cgroup file and munge key are copied from the NFS onto the node and Munge is enabled and ran to test that it will work on the node. By using Munge to generate a key on the master node and have the compute node attempt to decrypt the key. If the key is successfully decrypted, then Munge is set up correctly on the node. If not, then the Munge key may not be consistent across each node in the cluster. This is repeated for each compute node in the cluster.

Next, SLURM was enabled and was run on the cluster. By using the "sinfo" command on the master node, all of the nodes that SLURM can detect are shown. SLURM can be tested by running the hostname command on the cluster. If SLURM is correctly set up, then the hostnames of each node apart from the master will be displayed.

5. Installing OpenMPI

OpenMPI, an open-source Message Passing Interface, will enable the nodes to connect with each other while running tasks, meaning that they can crack passwords together. The “srun” command is used on the master node to install OpenMPI onto each node. This was installed by running an “srun apt install” command from the master node to distribute the command to all nodes, this way the command only had to be entered once but the whole cluster would download OpenMPI.

6. Installing Python

The programming language chosen for the cluster was python since it is very basic and is very well recognised. It is also used by most students studying Ethical Hacking at Abertay University. As all nodes need to access the library files it would be inefficient to install Python onto every Pi so instead it was installed onto the NFS which every node can access. To do this command was used “wget <https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tgz>” to get the Python package. It was then unzipped using the command “tar xvf Python-3.7.3.tgz”, this was all done in the file /clusterfs/python so it was located on the NFS. Then the directory /clusterfs/usr was made to store the install of Python in. Then using the command “./configure \ --enable-optimizations \ --prefix=/clusterfs/usr \ --with-ensurepip=install” was used to configure the Python to install in a custom location and to also install the pip program. Finally using “make” and then “make install” installs Python onto the NFS.

Python was installed after this however errors were being returned from the nodes, after some investigation it was found that some dependencies were missing, the command “srun -n=3 sudo apt install libatlas-base-dev” retrieved the missing files. The command Using the command “srun -n=3 /clusterfs/usr/bin/python3 -c "print('Hello')” confirmed that Python worked on the cluster.

7. Installing John The Ripper

Since John The Ripper already has MPI support built into the software itself it only needs to be built on the master node, when jobs are submitted John will handle everything else when it comes to passing the specific jobs to the nodes. This was done by using “wget” to retrieve the source package and downloading it onto the NFS. From here the package was unzipped using “tar”, then the package was configured and made for install onto the master node using “make”, “./configure” and “make install” to install the program onto the master node.

8. Writing the Demo Script

Once the cluster was set up and all of the programs successfully installed, the writing of the demo scripts could begin. Three files were created on the cluster, demo.py, johnDemo.sh and johnDemoWordlist.sh. The bash files were two SLURM batch scripts that would use John the Ripper either to brute-force or use a wordlist to crack the default Raspberry Pi password “raspberrypi”. The python file when run would provide the user the option between brute-force or a wordlist crack and run the relevant SLURM script. Once the job was finished on the cluster the script would then display the relevant SLURM output file.

9. Comparing the Cluster to a Home PC

To benchmark the cluster another python script and SLURM batch file were created. The script timed how long it took the cluster to crack “raspberrypi” 10 times and displayed the best, median and mean times. The times were noted and the SLURM file edited to change the number of cores and the script was run again. This was repeated for 1 core through to 7 cores. Once the testing of the cluster was complete, a variation on the Python script was run on a home PC and the time was recorded to compare our cluster’s performance.

Results

Raspberry Pi cluster benchmark results

The final product contains 4 Slave Pis totalling 7 CPU cores and another Pi 1 for the master. On the NFS there are also files to aid in both demonstration and benchmarking. These files are also shown under their file name in Appendix B. The file `demo.py` and the related `johnDemo.sh` and `johnDemoWordlist.sh` provide a demonstration of the working cluster with John the Ripper cracking the default Raspberry Pi password of “raspberrypi”. On the NFS there is the wordlist `rockyou.txt`, however due to the size of it, in the demonstration and benchmark scripts a variation on the default John the Ripper wordlist is used. `Demo.py` gives the user the option between cracking with a wordlist or brute-force however brute-force is not recommended until the cluster is massively increased in size or the `johnDemo.sh` file is edited to give John the Ripper a charset detailing that the password is lowercase.

The file `benchmark.py` and the bash script `wordlistBenchmark.sh` allows the user to easily benchmark the nodes by manually changing the bash script’s number of tasks (CPU cores) and running the python script. This was done and the number of nodes were incremented to gain an insight into how significantly more Pis would benefit the cluster.

The client originally asked that the cluster be compared to one of the Hacklab workstations, however due to COVID-19 this could not be done until a much later date. Instead, one of the team member’s (Stuart Rankin) home PC was used instead. The PC specifications are the following:

- CPU – AMD Ryzen 2700x
- RAM – Corsair Vengeance 16GB (2x8GB) DDR4 3200MHz
- Motherboard – Gigabyte X470 AORUS Ultra Gaming
- GPU – GeForce GTX 1080 WINDFORCE OC

The following graphs illustrate the results of the benchmark script on the cluster and PC respectively.

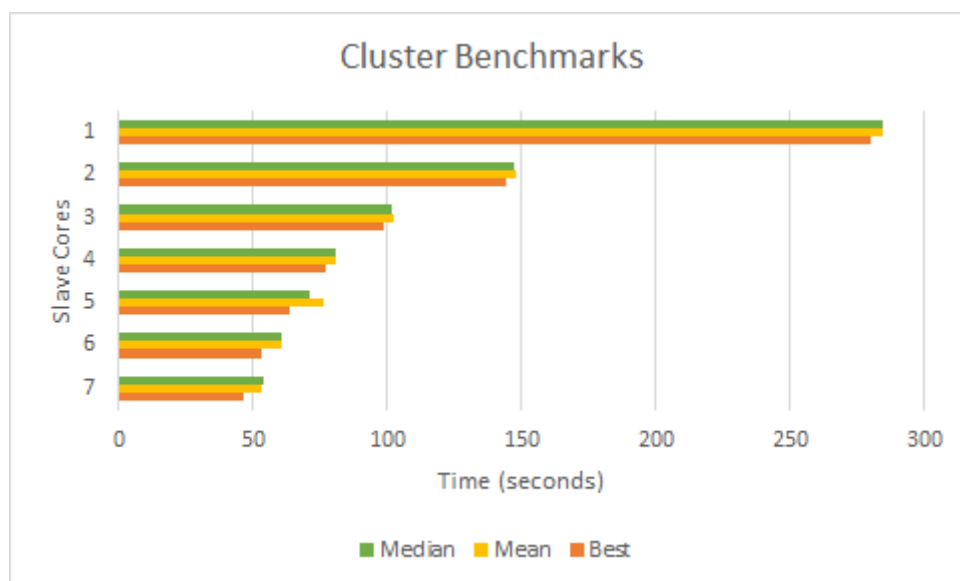


Figure 3
Benchmark.py results on cluster

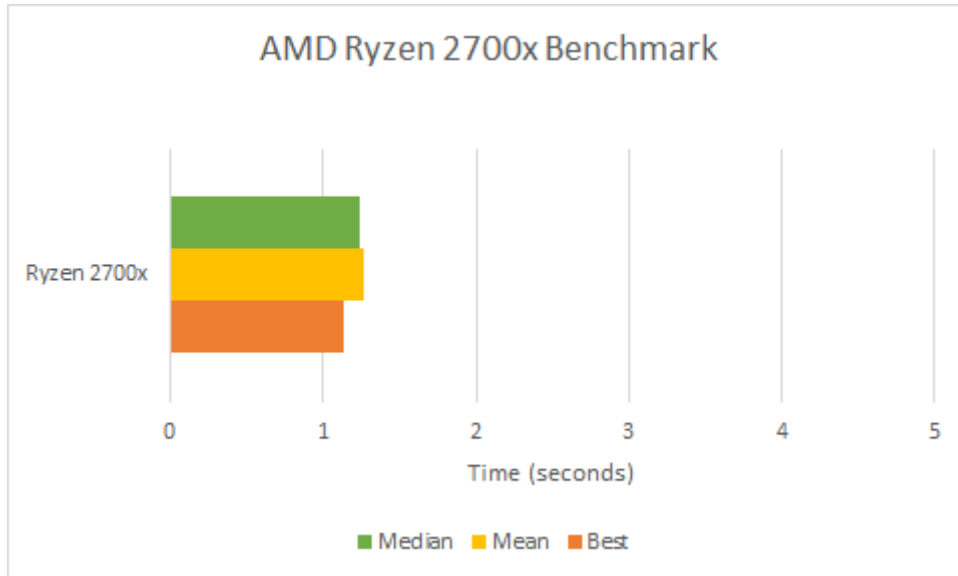


Figure 4
Benchmark results from comparison PC

	1 Node	2 Nodes	3 Nodes	4 Nodes	5 Nodes	6 Nodes	7 Nodes	Ryzen 7 2700x
Best	279.98	144.1945	98.9365	77.1512	63.9045	52.9528	46.5489	1.1309
Mean	284.8681	147.6378	102.1322	80.636	76.5941	60.355	52.9719	1.2709
Median	284.4606	146.9869	101.7228	80.7424	71.1106	60.7231	53.9173	1.242

Figure 5
Table of full results

Figure 3 provides a clear representation of how the cluster can utilise hardware that on its own fails to crack a password in a reasonable time as well as showing how effective adding more nodes can be. Looking at all the Figures it shows that the cluster was easily beaten by the comparison PC however it is important to keep in mind that the AMD Ryzen 2700x is a current mid-range CPU, containing 8 cores and 16 threads. As well as this, the 2700x also has a Base Clock of 3.7GHz which is more than double the clock speed of the latest Raspberry Pi, the Raspberry Pi 4 Model B which has a quad core with a clock speed of 1.5GHz. The comparison's PC current specifications could cost upwards of £1000 compared to the Raspberry Pi 4's current price of around £50.

The common way to benchmark John the Ripper is using the in-built feature which is run using the flag "--test". This was done on the cluster and on the comparison PC and both results can be found in Appendix C under "johnTest.sh Result" and "Comparison john -- test Result". Due to memory constraints on the cluster johnTest.sh would crash, to prevent this the swap file on each node was increased to 2GB which likely caused notable performance loss.

Whilst the results are not as good as expected, it still shows that with more nodes being added it could crack passwords much faster. As well as this, the cluster still achieves other goals which will be reviewed in the discussion section.

Discussion

General Discussion

As can be seen in the results, the cluster was able to successfully crack the list of given hashes using John the Ripper. By utilising more slave cores, the time taken to complete the password cracking was reduced. The reduction in time is more substantial when increasing the number of cores used from 1 to 2 and the reduction in time is less and less as more cores are added. This shows that by adding additional Raspberry Pis to the cluster the time could be reduced further, enabling the cluster to approach the power of a workstation PC, allowing outdated hardware to be repurposed instead of being thrown away. We were only able to write a demo script for the cluster and not a tutorial script or a scalability script, due to unforeseen, uncontrollable factors that arose during the project development stage.

A user manual was created which contains clear instructions for a user to set up their own MPI cluster and steps on how to manage the cluster if problems should arise. The manual explains steps such as assigning static IP addresses and hostnames to each node, setting up an NFS and installing the necessary programs such as SLURM and OpenMPI.

To add or remove nodes to or from the cluster, the user will need to add or remove them manually, the conditions brought on by the COVID-19 pandemic severely hampered our progress since we were no longer able to all have physical access to the cluster, which was needed to write scripts and perform node configuration.

The cluster itself was set up properly and can function very well, performing the task that we instructed it to complete. The NFS is set up to allow the nodes to access shared files and has room to be improved upon in the future if conditions allow. The ROI is great since the cluster does not cost anything to produce. It is made of old hardware that is ready to be thrown away. The cluster could be used to attract new students at open days, causing financial benefits for the client.

The cluster presently has a text-based interface which functions well enough now since the cluster only has a few scripts to run. With more scripts added and possibly more functions added to the cluster, the user interface should be updated so that users who are new to Raspberry Pis and clusters can easily use the cluster.

Conclusions

This project has been successful in creating a good finished product, even with the team having to counteract the restrictions that COVID-19 placed onto the project. There are several benefits the project has created for the client which will prove to be useful in the future. The first one is the fact that the client now has a fully functioning Raspberry Pi cluster using the OpenMPI software which was specified by the client in the project proposal.

Because of this cluster the client now has several other benefits which they can utilize. Due to the way the cluster has been set up it allows for easy expandability for the future, allowing the client to add as many Raspberry Pis as they'd like to the cluster. With the addition of more nodes comes more compute power for the cluster, meaning John The Ripper will most likely crack passwords faster. The client has also expressed that they have a lot of unused Raspberry Pis which we saw first-hand during this project, by adding these to the cluster unused Pis would be put to good use in the cluster.

This cluster is also very flexible because of the way it utilizes a Network File System (NFS) between all nodes. This was used in the project so that all nodes could access the same files at the same time, primarily it is used by the current cluster to store all the Python library files and script batch job files. Knowing this, and the fact the NFS has currently over 100GB of free space leaves a substantial amount of room for more programs which could be used by the cluster. Meaning that in the future if it needs to be repurposed it could easily be changed to serve an alternative function.

If a password file is particularly difficult to crack students will often leave a Hacklab workstation running John The Ripper to crack the file while they leave for sometimes several hours, this takes up a workstation which another student could use. Because of this another benefit is the fact that the cluster is completely dedicated to cracking passwords. Unlike the other Hacklab workstations which need to be used for other activities this cluster can be given a file to crack and left alone until it is finished, making a workstation available to be used by another student.

While the team researched the project, we found that there wasn't any documentation about how to create a cluster with the same configuration that's been created in this project. Because of that a manual has been created (see Appendix I) for future reference which has 3 parts. First section explains how to create the Cluster, this was made in case the Cluster needed to be remade. The second section shows a user how to create batch jobs and scripts for the cluster to run. Finally, the last section just shows some useful commands users will need to effectively use the cluster. This should prove very useful to the client in the future if they want to work with this cluster further.

The client also expressed that they thought the cluster would be a good system to show to potential students on open days. Since the cluster has been completed to a good standard and has a demo script showcasing the cluster's capabilities, the team does believe it would be a great showcase to demonstrate to potential students' what sort of things they could be working on during their time at Abertay University.

Finally, one of the features of the cluster which will be of great use to the client is the program SLURM which was implemented into the cluster during the project. The program offers many features which will become useful if the cluster is expanded to include more nodes. These features are:

1. SLURM can allow a cluster to be defined into partitions, this allows a user to section off certain nodes to do different jobs at the same time as other partitions.

2. Even in the same partition multiple nodes can be put to do different jobs at the same time.
3. Allows a user to track currently running jobs and how long they've been running.
4. Allows a user to see if all nodes are up or working as intended.
5. By using the SLURM "sbatch" command heterogeneous jobs⁸ can be submitted to the cluster.

Listing all the features SLURM has would make the list too long, but simply put SLURM has many features which would greatly assist the cluster in the future. As a whole the cluster is a very flexible and usable system which can perform the jobs the client wanted to get from the cluster.

Future Work

Due to COVID-19 we weren't able to achieve all the tasks we set out to complete. This was due to the need of physical access which is essential to complete some of the tasks which were planned. Because of this, only the demo script was created to showcase the cluster's capabilities, this reduction of the scope was agreed to by the client Ethan Bayne (see appendix A - emails). The following other tasks were not completed:

1. Creation a tutorial script to take a user through the process of how to create and run their own John The Ripper job on the cluster.
2. Creation of a "node expansion script" which would allow a user to easily add a new node onto the cluster without having to change several configuration files manually.

These scripts were added to the scope because they would greatly benefit the cluster in terms of ease of use and expandability in the future. Because of this any future work would start with the creation and implementation of these scripts into the cluster.

We'd also look at the prospect of adding more scripts and or programs to the cluster to improve its usability. When researching clusters the team found lots of projects which used different types of interfaces to interact and control their cluster. One of the most interesting interfaces found was a Webpage interface which could be used to control the cluster. In the case of this cluster a Webpage interface could be used to upload files directly to the cluster, where the user could continue to use the cluster from this webpage interface. This would remove the need to have to manually place files onto the master node through a storage medium, it would also reduce interaction with the terminal interface currently needed to control and make changes to the cluster.

Other work that the team would've liked to added to the project would've been the addition of more nodes to the cluster as a whole. This would be beneficial for multiple reasons, for one it would use up more Raspberry Pis which are currently not being used by the client and put them to good use. Secondly it would make the cluster faster at cracking passwords with the added compute power, lastly with more nodes means that the cluster could be partitioned more with SLURM which could allow more asynchronous tasks to be performed by the cluster at once.

The final piece of future work we'd want to look at doing would be the addition of purpose built housing for the Pi cluster. As it currently stands the cluster is powered using USB micro B wires which are connected to a USB power hub, this along with the ethernet cables used to connect the Pis to the switch this creates a cluster which isn't as neat as it could be. Figure 6 shows how the cluster looks in its current configuration.



*Figure 6
Pi cluster set up.*

While this setup works and can sit on a desk easily, it isn't practical. Keeping track of which wire goes where and trying to move the cluster is an awkwardly difficult task. With a proper housing built for it this problem could be easily mitigated. Also, with the chance of the cluster becoming bigger in the future with more nodes being added, bringing with them more wires and need for space, this problem will only become a more apparent issue meaning housing for the cluster could become necessary.

Call to action

If there's any more work the client would like performed on the cluster, team leader Neirin McDonald would be happy to continue to work on it more in his spare time for academic year 2020/2021. If there's any other issues with the cluster which can't be fixed, or if there's something you'd like to know about the cluster that's not covered in this whitepaper the team can be contacted through these email addresses:

Ewan Scott - Spacehopper Liaison: 1700230@abertay.ac.uk

Neirin McDonald – Spacehopper team leader: 1701141@abertay.ac.uk

Jake Mills – Spacehopper hardware: 1203200@abertay.ac.uk

Stuart Rankin – Spacehopper minute taker: 1701198@abertay.ac.uk


References




- ¹ Raspberry Pi (no date) *Raspbian*. Available at: <https://www.raspberrypi.org/downloads/raspbian/>
- ² University of Missouri-St. Louis (no date) *Moore's Law*. Available at: https://www.umsl.edu/~siegelj/information_theory/projects/Bajramovic/www.umsl.edu/~abcdcf/Cs4890/link1.html (Accessed: 15 April 2020).
- ³ Knapton, S. (2017) "Discarded phones, computers and electronics behind world's fastest growing waste problem", *The Telegraph*, 13 December. Available at: <https://www.telegraph.co.uk/science/2017/12/13/discarded-phones-computers-electronics-behind-worlds-fastest/> (Accessed: 15 April 2020).
- ⁴ Vaute, V. (2018) "Recycling is not the answer to the e-waste crisis", *Forbes*, 29 October. Available at: <https://www.forbes.com/sites/vianneyvaute/2018/10/29/recycling-is-not-the-answer-to-the-e-waste-crisis/#7c27c7ee7381> (Accessed: 15 April 2020).
- ⁵ Electronics TakeBack Coalition (2013) *Facts and figures on e-waste and recycling*. Available at: <http://www.electronicstakeback.com/wp-content/uploads/Facts-and-Figures-on-E-Waste-and-Recycling.pdf> (Accessed: 15 April 2020).
- ⁶ Document prepared and revised by Natalie Coull, Colin McLean, Andrea Szymkowiak
- ⁷ Graham, G., 2005. *The White Paper FAQ (Frequently Asked Questions)/That White Paper Guy – Gordon Graham*. [online] Available at: https://www.thatwhitepaperguy.com/white-paper-faq-frequently-asked-questions/#what_is > [Accessed 9 May 2016].
- ⁸ slurm (no date) *Heterogenous Job Support*, Available at: https://slurm.schedmd.com/heterogeneous_jobs.html
- ⁹ Count upon security (May 07, 2015) Step-by-Step Clustering John the Ripper on Kali, Available at: <https://countuponsecurity.com/2015/05/07/step-by-step-clustering-john-the-ripper-on-kali/>
- ¹⁰ Pfister, G. (1998) *In search of clusters*. Upper Saddle River, NJ: Prentice Hall PTR.

Appendices

Appendix A - E-mails


Email from Ewan Scott to Ethan Bayne:


 **EWAN SCOTT**
Mon 23/03/2020 22:19
Ethan Bayne ✓

Hello, we are wondering what will happen with our Raspberry Pi Cluster project now that the lockdown will take place, since we can no longer hand the cluster to each other.

Reply from Ethan Bayne:

 **Ethan Bayne**
Tue 24/03/2020 09:28
EWAN SCOTT ✓



Hi Ewan,

How far along are you with the cluster? If you can break down for me what you have done to date and what you would not manage to implement, I would be able to advise further.


Cheers,






Ethan

Dr Ethan Bayne | Lecturer in Computer Science
Division of Cyber Security | School of Design and Informatics
Abertay University | Bell Street
DUNDEE | DD1 1HG

E: e.bayne@abertay.ac.uk
T: 01382 308510

Confirmation from Ewan of current progress:


 **EWAN SCOTT**
Tue 24/03/2020 15:13
Ethan Bayne ✓

We have created a working cluster that can have nodes added or removed, which uses NFS storage, SLURM, Munge and OpenMPI to communicate between nodes. We still need to write the submission scripts, demo script, and tutorial script.

.....

Confirmation of scope reduction:

 **Ethan Bayne**
Wed 25/03/2020 09:38
EWAN SCOTT ✓

Hi Ewan,

I would advise to reduce the scope. Could your group write a demo script to finalise the project and then pivot your attention to the write up?

Cheers,

Ethan

Dr Ethan Bayne | Lecturer in Computer Science
Division of Cyber Security | School of Design and Informatics
Abertay University | Bell Street
DUNDEE | DD1 1HG

E: e.bayne@abertay.ac.uk
T: 01382 308510

Appendix B – Scripts

demo.py

File path: `/clusterfs/john/scripts/demo.py`

```
import os
import time
import subprocess
```

```

import atexit

def cancel(temp):
    os.system("scancel " + temp)

#Change to correct directory
os.chdir("/clusterfs/scripts/john")

#Clears John's cached cracked passwords
os.system("rm /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john.pot")

print("For demonstration purposes the Raspberry Pi's password 'raspberry' is
used.")

option = 0
while option not in ('1', '2'):
    option = str(input("1. Wordlist\n2. Incremental (Brute-Force)\n"))

if option == '1':
    #Cracks in wordlist mode
    jobid = str(subprocess.check_output("sbatch
/clusterfs/scripts/john/johnDemoWordlist.sh", shell=True))
else:
    #Cracks in brute-force mode
    jobid = str(subprocess.check_output("sbatch
/clusterfs/scripts/john/johnDemo.sh", shell=True))

#gets jobid from previous bash script output
split = jobid.split(" ")
jobid = split[-1]
jobid = jobid[0:-3]
jobid = jobid.strip()
print("Job ID = " + jobid)

#Cancels current job on quit
atexit.register(cancel, temp=jobid)

#Waits for slurm file to be created
while not os.path.exists("slurm-"+jobid+".out"):
    time.sleep(1)

#If file exists
if os.path.isfile("slurm-"+jobid+".out"):
    #Open output file and read into lines
    f = open("slurm-" + jobid + ".out", "r")
    lines = f.read().splitlines()
    #While nothing in file
    while not lines:
        #reread file
        lines = f.read().splitlines()

#Loops until completed
while(True):

```

```
#If job not complete ("completed" not in final line")
if "completed" not in lines[-1]:
    #Re open file and re read and wait a second to prevent pointless rereads
    f.close()
    f = open("slurm-" + jobid + ".out", "r")
    lines = f.read().splitlines()
    time.sleep(1)
else:
    #If completed show output file to user
    os.system("cat /clusterfs/scripts/john/slurm-"+jobid+".out")
    break
#Wait for input to close
print("Enter anything to finish")
input()
f.close()
else:
    print("Error! Can't find relevant slurm output file")
    raise ValueError("slurm-"+jobid+".out isn't a file!")
```

johnDemoWordlist.sh

/clusterfs/scripts/john/johnDemoWordlist.sh

```
#!/bin/bash
#SBATCH --ntasks=7

cd $SLURM_SUBMIT_DIR

mpirexec -n 7 /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john --
wordlist=/clusterfs/scripts/john/password.lst /clusterfs/scripts/john/demo.txt
```

johnDemo.sh

/clusterfs/scripts/john/johnDemo.sh

```
#!/bin/bash
#SBATCH --ntasks=7

cd $SLURM_SUBMIT_DIR

mpirexec -n 7 /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john --
incremental:Lower /clusterfs/scripts/john/demo.txt
```

benchmark.py

/clusterfs/scripts/benchmark/benchmark.py

```
import os
import time
import subprocess

#Change to correct directory
```

```
os.chdir("/clusterfs/scripts/benchmark")

#Clears John's cached cracked passwords
os.system("rm /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john.pot")

#Benchmarks 10 times
benchmarks = []
counter = 0
while counter < 10:
    os.system("rm /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john.pot")
    start = time.time()
    os.system("sbatch wordlistBenchmark.sh")
    counter += 1

    print("Running benchmark number " + str(counter))

    output = str(subprocess.check_output("squeue", shell=True))

    output = output.split("\n")

    while len(output) != 2:
        output = str(subprocess.check_output("squeue", shell=True)).split("\n")
        end = time.time()
        benchmarks.append(end-start)

#Puts in order
benchmarks.sort()

size = len(benchmarks)
print(benchmarks)

#Removes Slurm Output files
os.system("rm slurm-*")

#Mean Calculation
average = 0.0
for i in benchmarks:
    average += i

average = average/size

#Median Calculation
if size % 2 == 0:
    median1 = benchmarks[size//2]
    median2 = benchmarks[size//2 - 1]
    median = (median1 + median2)/2
else:
    median = benchmarks[size//2]

#Displays averages
print("The mean is: " + str(average))
print("The median is: " + str(median))
```

wordlistBenchmark.sh

/clusterfs/scripts/benchmark/wordlistBenchmark.sh

```
#!/bin/bash
#SBATCH --nodes=7
```

```
cd $SLURM_SUBMIT_DIR
```

```
mpiexec -n 7 /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john --
wordlist=/clusterfs/scripts/john/password.lst /clusterfs/scripts/john/demo.txt
```

johnTest.sh

/clusterfs/scripts/benchmark/johnTest.sh

```
#!/bin/bash
#SBATCH --ntasks=7
```

```
cd $SLURM_SUBMIT_DIR
```

```
mpiexec -n 7 /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john --test
```

Appendix C - Results

johnTest.sh Result

/clusterfs/scripts/benchmark/johnTest.sh Result

```
MPI in use, disabling OMP (see doc/README.mpi)
Node numbers 1-7 of 7 (MPI)
Benchmarking: descrypt, traditional crypt(3) [DES 32/32]... (7xMPI) DONE
Warning: "Many salts" test limited: 56/256
Many salts: 522240 c/s real, 532897 c/s virtual
Only one salt: 496640 c/s real, 501656 c/s virtual
```

```
Benchmarking: bsdicrypt, BSDI crypt(3) ("_J9..", 725 iterations) [DES 32/32]...
(7xMPI) DONE
Speed for cost 1 (iteration count) of 725
Warning: "Many salts" test limited: 33/256
Many salts: 17932 c/s real, 18112 c/s virtual
Only one salt: 17920 c/s real, 17920 c/s virtual
```

```
Benchmarking: md5crypt, crypt(3) $1$ (and variants) [MD5 32/32 X2]... (7xMPI)
DONE
Warning: "Many salts" test limited: 4/256
Many salts: 8533 c/s real, 8694 c/s virtual
Only one salt: 8533 c/s real, 8613 c/s virtual
```

```
Benchmarking: md5crypt-long, crypt(3) $1$ (and variants) [MD5 32/32]... (7xMPI)
DONE
Raw: 4876 c/s real, 4876 c/s virtual
```

Benchmarking: bcrypt ("2a05", 32 iterations) [Blowfish 32/32]... (7xMPI) DONE
Speed for cost 1 (iteration count) of 32
Raw: 457 c/s real, 457 c/s virtual

Comparison john --test Result

Ran with Open Multiprocessing (OpenMP)

Benchmarking: descrypt, traditional crypt(3) [DES 128/128 AVX]... (16xOMP)
DONE

Many salts: 42991K c/s real, 3068K c/s virtual

Only one salt: 29163K c/s real, 2342K c/s virtual

Benchmarking: bsdictcrypt, BSDI crypt(3) ("_J9..", 725 iterations) [DES 128/128
AVX]... (16xOMP) DONE

Speed for cost 1 (iteration count) of 725

Many salts: 1370K c/s real, 99816 c/s virtual

Only one salt: 1282K c/s real, 93580 c/s virtual

Benchmarking: md5crypt, crypt(3) \$1\$ (and variants) [MD5 128/128 AVX 4x3]...
(16xOMP) DONE

Many salts: 397056 c/s real, 27650 c/s virtual

Only one salt: 393984 c/s real, 27493 c/s virtual

Benchmarking: md5crypt-long, crypt(3) \$1\$ (and variants) [MD5 32/64]...
(16xOMP) DONE

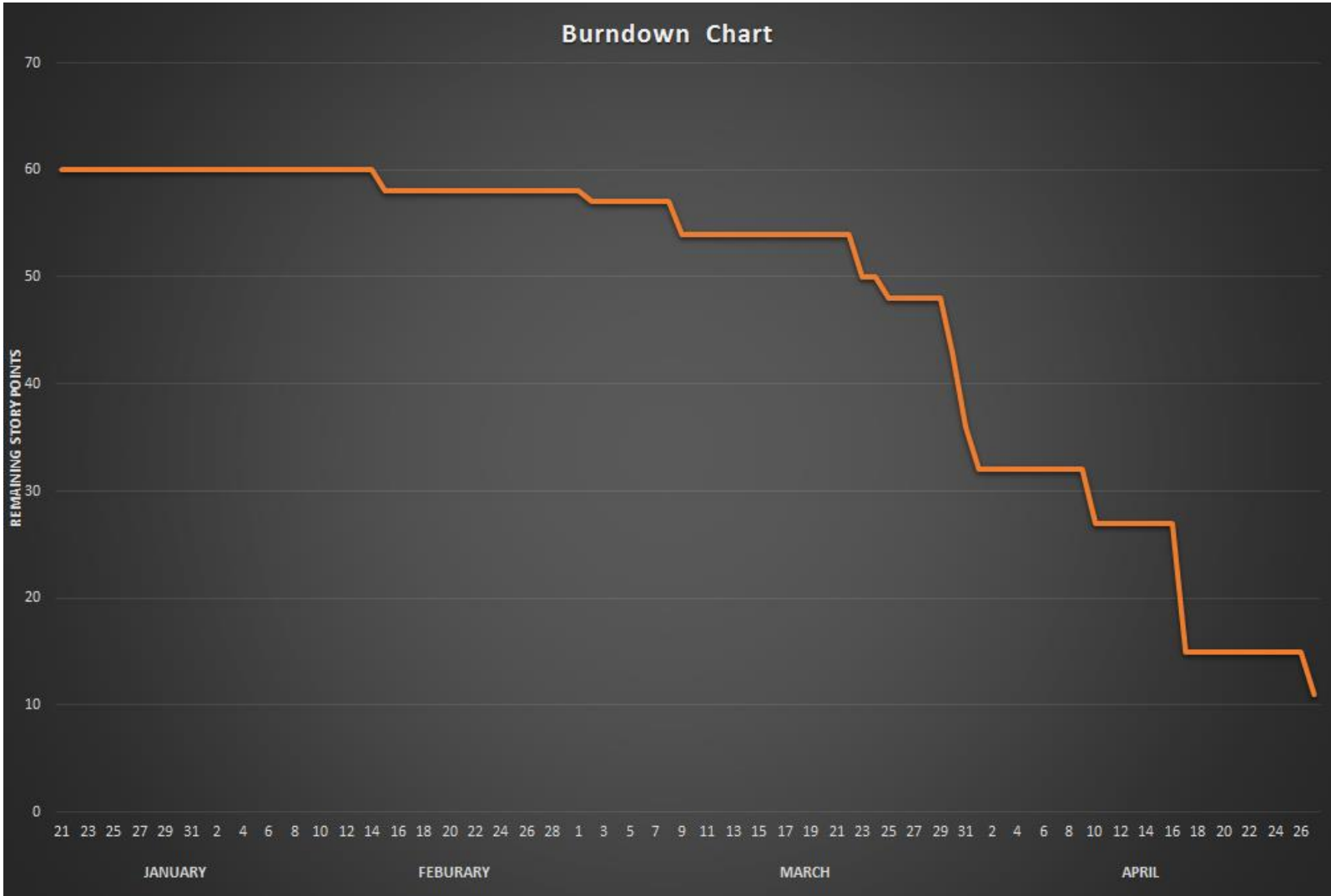
Raw: 84992 c/s real, 5998 c/s virtual

Benchmarking: bcrypt ("2a05", 32 iterations) [Blowfish 32/64 X3]... (16xOMP)
DONE

Speed for cost 1 (iteration count) of 32

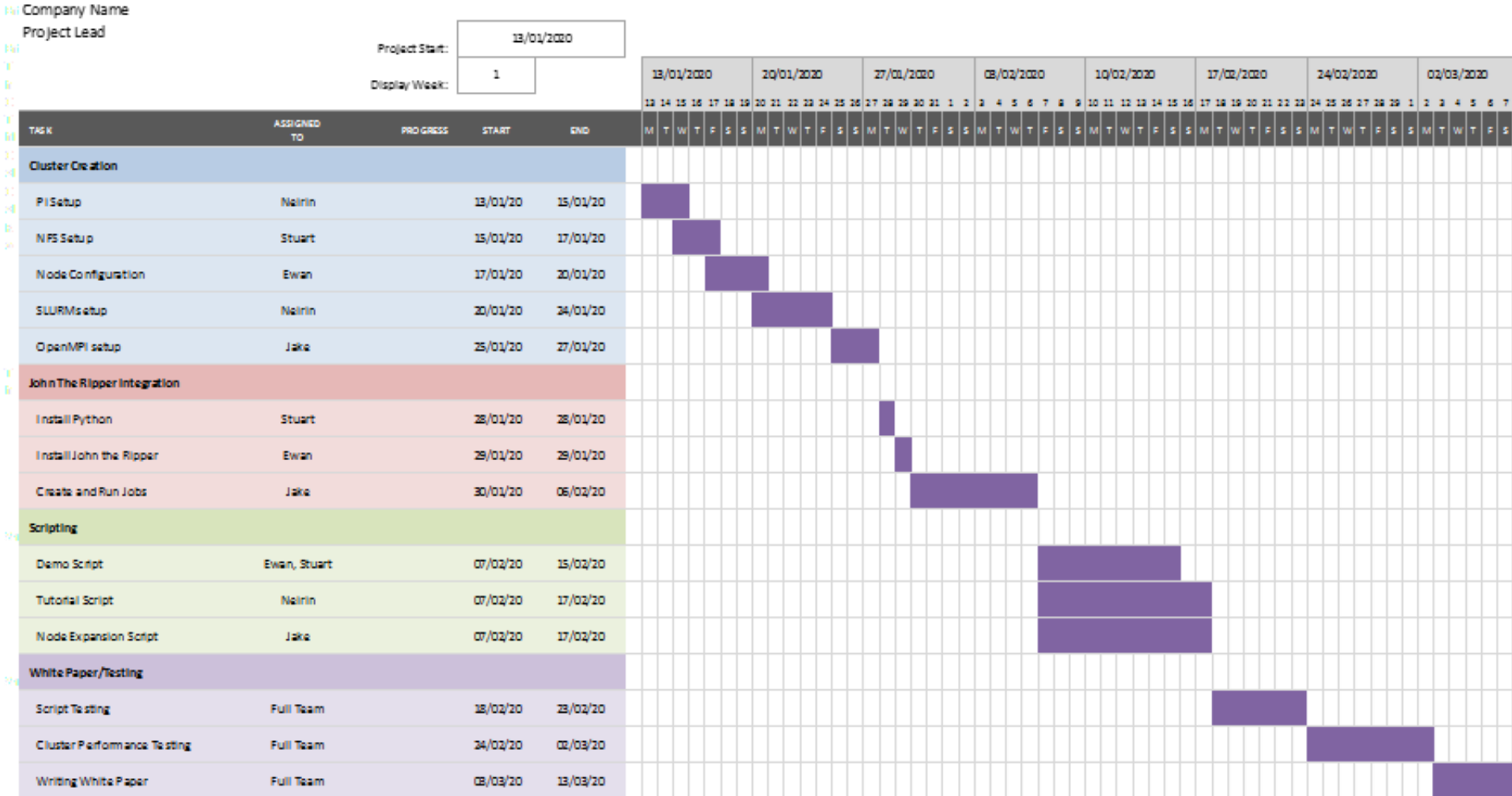
Raw: 14544 c/s real, 1032 c/s virtual

Appendix D - Burndown Chart



Appendix E Gantt Chart

<https://www.vertex42.com/ExcelTemplates/simplegantt-chart.html>



Appendix F - Deliverables & requirements

Agreement Form: Requirements

Group Number:

Team name: Space Hopper

Team members (print): NEIRIN MCDONALD, EWAN SCOTT, JAKE MILLS, STUART RANKIN


Project Title: Raspberry Pi MPI cluster password cracker

Please refer to the attached documentation for full details on the project. The requirements are listed in Table 1. The signatures below indicate that the requirements for this project have been agreed by the project stakeholders.


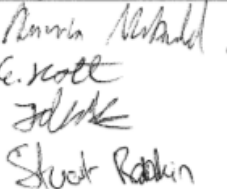
Any changes to the project documentation should be made using the correct change authorization procedure agreed with the subject specialist.

Table 1

ID	List of Agreed Requirements (fill in)
1	<ul style="list-style-type: none"> • Ability to crack password hashes • Interface is at a good usable standard • Ability to add/ remove nodes from cluster • Good user tutorial • User manual
2	
3	
4	
5	

Stakeholders	Signatures	Date
Team members	Neirina McDonald E. Scott Jake Mills Stuart Rankin	17/01/2020 17/01/2020 20/01/2020 20/01/2020
Subject Specialist		2020-01-17
Client (if applicable)		2020-01-17

Agreement Form: Project Deliverables

Group Number, Names of Team Members, and Programme	Group number: Team name: Space Hopper Members: Ewan Scott Neirin McDonald Stuart Rankin Jake Mills
Subject Specialist's Name	Dr Ethan Bayne (e.bayne@abertay.ac.uk) Mr David McLuskie (d.mcluskie@abertay.ac.uk)
The deliverables listed below will be submitted by the team by the 21st of April 2020 (21/04/2020)	
Part A Deliverables	<ul style="list-style-type: none"> • Raspberry pi MPI cluster • Demo script • Tutorial script • Node expansion script/ instructions • User manual • Requirements specification • Software design specification • Testing documentation
Subject Specialist's Signature	
Team Members' Signatures	

Appendix G – Minutes

Spacehopper-meeting 1

2020-01-24

Start Time: 13:00

End Time: 14:00

Hacklab, Abertay University

Attendees: Stuart Rankin, Jake Mills, Ewan Scott, Neirin MacDonald

Absentees:

Quorum Present

Agenda

- Investigate Router
- Discuss with Ethan about hardware needs
- Plan work to be done

Notes

Team discussed hardware that is still required

Ewan communicated needs with Ethan

After Neirin got to the meeting the team gained a new switch to replace the broken one and the team gained 2 more Pis as well as components to go with them such as ethernet and power cables.

Atmosphere and Challenges

The atmosphere was optimistic and excited to get started

Main challenge for the team was Neirins absence as team leader for the first half of the meeting.

Next Meeting

2020-01-31

Start Time: 13:00

Hacklab, Abertay University

Next Meeting Agenda

- Discuss progress made
 - Discuss any challenges that come up
-

Spacehopper – meeting 2

2020-01-31

Start Time: 14:00

End Time: 14:35

Hacklab, Abertay University

Attendees: Stuart Rankin, Jake Mills, Ewan Scott

Absentees: Neirin MacDonald

Quorum Present

Agenda

- Discuss Progress made
- Discuss any challenges

Notes

Jake caught attendees up on the progress he has made, configuring and labelling the Pi's

Ewan and Stuart discussed/decided on what to work on whilst the cluster is being configured. Ewan will work on the document and keep researching OpenMPI/SLURM. Stuart is to begin work on the python scripts and also continue OpenMPI/SLURM research so the knowledge can easily be applied once cluster is set up.

Atmosphere and Challenges

The atmosphere was similar to last week's meeting. With no significant challenges appearing, Neirin's absence did not majorly affect the atmosphere

and Neirin was easily caught up through the messaging app 'Signal'.

Next Meeting

2020-02-07

Start Time: 14:00

Hacklab, University

Next Meeting Agenda

- Discuss Progress made
 - Discuss any challenges
 - Approve previous minutes
-

Spacehopper – meeting 3

2020-02-07

Start Time: 12:45

End Time: 13:30

Hacklab, Abertay University

Attendees: Stuart Rankin, Jake Mills, Neirin MacDonald, Ewan Scott

Absentees:

Quorum Present

Agenda

- Approve Minutes
- Discuss Progress
- Discuss Challenges

Notes

Jake explained about the issue he had run into of the planned Master Node failing to boot/display. Ewan and Jake went down to get another Pi however

the technician was out so instead Jake will continue with one of the slave Pi's becoming the Master for now.

Ewan talked about what work he had made on the document. Stuart discussed the progress he had made on the scripts and also explained the issue of scripting for the cluster without the cluster being completed i.e what user inputs will be required etc.

Atmosphere and Challenges

The atmosphere was good despite the challenges the have popped up the team are confident they can continue to stay on schedule without sacrificing much. The Pi's should be flashed by Monday so work should be able to continue.

Next Meeting

2020-02-14

Start Time: 13:00

Hacklab, Abertay University

Next Meeting Agenda

- Approve Minutes
- Discuss Progress
- Discuss Challenges

Spacehopper- meeting 4

2020-02-14

Start Time: 16:15

End Time: 16:50

Hacklab, Abertay University

Attendees: Stuart Rankin, Jake Mills, Ewan Scott, Neirin MacDonald

Absentees:

Quorum Present

Agenda

- Approve Minutes
- Discuss Progress
- Discuss Challenges

Notes

Jake discussed the progress he had made with the cluster.

Ewan and Neirin went down to get more Pi's however the technician wasn't in. The team discussed what work needed to be done on the document.

Atmosphere and Challenges

The atmosphere was good as progress was being made and the team are on schedule. There were no significant challenges the came up during the meeting.

Next Meeting

2020-02-21
Start Time: 13:00
Hacklab, Abertay University

Next Meeting Agenda

- Approve Minutes
- Discuss Progress
- Discuss Challenges

Spacehopper- meeting 5

2020-02-21
Start Time: 14:00
End Time: 15:00
Hacklab, Abertay University

Attendees: Stuart Rankin, Jake Mills, Ewan Scott, Neirin MacDonald

Absentees:

Quorum Present

Agenda

- Approve Minutes
- Discuss Progress
- Discuss Challenges

Notes

Neirin explained how he had managed to get the Pi that wasn't working to boot. Jake spoke about the progress he had made on configuring SLURM on the other Pi's. Next meeting was re-arranged due to Securi-Tay

Atmosphere and Challenges

The atmosphere was positive with the knowledge that the Pi was fixed and now working. There were no challenges with the team just the SLURM configuration of the cluster.

Next Meeting

2020-03-03

Start Time: 14:00

Hacklab, Abertay University

Next Meeting Agenda

- Approve Minutes
 - Discuss Progress
 - Discuss Challenges
-

Spacehopper – meeting 6

2020-03-03

Start Time: 13:30

End Time: 14:00

Hacklab, Abertay University

Attendees: Stuart Rankin, Neirin MacDonald, Jake Mills, Ewan Scott

Absentees:

Quorum Present

Agenda

- Approve Minutes
- Discuss Progress
- Discuss Challenges

Notes

Jake explained how the issues he was having with SLURM were caused by the switch setting dynamic IP addresses for the Pi when it required fixed IP addresses. Neirin and Jake went to see the technician in order to get either a router or a switch that didn't force dynamic addresses. Ewan will email either technician or client to organise getting switch or router.

Atmosphere and Challenges

The atmosphere wasn't as good as previous meetings due to the delay caused by the switch. There was no challenges between members only the switch delay.

Next Meeting

2020-03-06

Start Time: 13:30

Hacklab, Abertay University

Next Meeting Agenda

- Approve Minutes

- Discuss Progress
- Discuss Challenges
- Discuss if the switch issue has been solved

Spacehopper – meeting 7

2020-03-06

Start Time: 14:00

End Time: 14:30

Hacklab, Abertay University

Attendees: Stuart Rankin, Neirin MacDonald, Ewan Scott

Absentees: Jake Mills

Quorum Present

Agenda

- Approve Minutes
- Discuss Progress
- Discuss Challenges

Notes

Jake, Neirin and Ewan had met the day prior and had managed to get a new switch. The new switch solved the issue of dynamic IPs.

According to Jake we still needed a router as we had to configure some settings however he wasn't there to explain further.

With the delays to the configuration, Stuart decided to try and start some of the scripts which could then be slightly changed or fixed when the cluster is set up.

Atmosphere and Challenges

The atmosphere was good with the dynamic IPs being fixed.

There was no other challenge other than the absence of Jake.

Next Meeting

2020-03-13

Start Time: 14:00

Hacklab, Abertay University

Next Meeting Agenda

- Approve Minutes
- Discuss Progress
- Discuss Challenges

Appendix I - Manual

Reference Tutorial - <https://medium.com/@glmdev/building-a-raspberry-pi-cluster-784f0df9afbd>

This manual is designed so as to be a step-by-step guide to replicating the project. However it can also be used for giving a better understanding of the cluster and how it works while also providing reference for individual tasks.

1. Assigning a Static IP Address

For ease of use IPs should be incremented per node, for instance: node1 is ip xxx.xxx.xxx.101, node2 is ip xxx.xxx.xxx.102 and so on.

- a. Open `dhcpcd.conf` :

```
sudo nano /etc/dhcpcd.conf
```
- b. Edit the fields under `interface eth0` to match the network configuration and assign a free IP address.

2. Accessing the Raspberry Pi Settings Interface

- a.

```
sudo raspi-config
```
- b. Select "Expand filesystem", this will allow the Pi to use the rest of the free space on the Pi.
- c. Go back to the first page and select "Localisation Options", then change "Timezone" and set this to your local time zone.
- d. Go back to the first page and select "Interfacing Options", then select "SSH" and enable this feature.

3. Setting Hostnames

Hostnames in the cluster have to all be different. For example, you could have the cluster set up as "Master, RPI1, RPI2, RPI3...." and so on. For an easier time during SLURM configuration, it's recommended that the work nodes should be named similarly like in the example with a number increasing on the end.

- a. Perform the following steps for each node.
- b.

```
sudo hostname <hostname here>
```
- c. Open the hostname file and edit accordingly :

```
sudo nano /etc/hostname
```
- d. Open the hosts file and edit accordingly :

```
sudo nano /etc/hosts
```

4. Sync System Time

The time on all nodes need to be the same or the cluster **wont work!**

- a.

```
sudo apt install ntpdate -y
```
- b.

```
sudo reboot
```

5. Connecting to Pi with SSH

This allows the user to control the cluster from one computer.

- a.

```
ssh pi@<node ip>
```

6. Locate Flash Drive for Shared Storage

This following command is used to find the location of the flash drive. It is usually located in the directory /dev/sda and can be identified by its size.

- a. `lsblk`

7. Formatting Flash Drive to ext4 Format

During this section be mindful not to format the wrong location.

- a. `sudo mkfs.ext4 /dev/<flash drive location>`
 - i. For example, if the flash drive was located in /dev/sda, the command would be `<sudo mkfs.ext4 /dev/sda>`.

8. Creating a Mount Directory

The following commands must be run on the Master node.

- a. `sudo mkdir /clusterfs`
- b. `sudo chown nobody.nogroup -R /clusterfs`
- c. `sudo chmod 777 -R /clusterfs`

9. Enabling Automatic Booting

- a. Input the command `blkid` and copy the UUID from the flash drive.
- b. Navigate to the fstab file and add the line below.
- c. `UUID=xxx /clusterfs ext4 defaults 0 2`
- d. Mount the flash drive.
- e. `sudo mount -a`

10. Set Loose Permissions

- a. `sudo chown nobody.nogroup -R /clusterfs`
- b. `sudo chmod -R 766 /clusterfs`

11. NFS

a. Exporting NFS

- i. Perform the following sections on the Master node.
- ii. Download and install the NFS server.
- iii. `sudo apt install nfs-kernel-server -y`
- iv. Navigate to exports file
- v. `sudo nano /etc/exports`
- vi. Add the following line below - all one line - then save the file.
- vii. `/clusterfs <node ip>(rw, sync, no_root_squash, no_subtree_check)`
- viii. Update the NFS kernel server.
- ix. `sudo exportfs -a`

b. Mounting NFS on Compute Nodes

- i. Using the following commands to install NFS on each node added to the cluster. These will create the shared storage directory and set permissions.
- ii. `sudo apt install nfs-common -y`
 1. `sudo mkdir /clusterfs`
 2. `sudo chown nobody.nogroup -R /clusterfs`
 3. `sudo chmod -R 777 /clusterfs`

- iii. Navigate to fstab file:
`sudo nano etc/fstab`
- iv. Add the following line below - all one line - then save the file. This will allow automatic mounting of the shared storage whenever the cluster is turned on.
- v. `<master node ip>:/clusterfs /clusters nfs defaults 0 0`
- vi. Mount the shared storage.
- vii. `sudo mount -a`
- viii. Files can now be accessed and created in /clusterfs by nodes.

12. SLURM

a. Master Node Configuration

i. Open /etc/hosts in a text editor

1. `sudo nano /etc/hosts`
2. Add the following lines:-
 - a. `<node 2 ip> <node 2 hostname>`
 - i. `<node 3 ip> <node 3 hostname>`
 - ii. `<node 4 ip> <node 4 hostname>`

Keep adding for the amount of nodes you have.

ii. Install SLURM Controller

1. `sudo apt install slurm-wlm -y`

iii. Copy and move default SLURM configuration file

1. `cd /etc/slurm-llnl`
2. `mv slurm.conf.simple slurm.conf`
3. `cp /usr/share/doc/slurm-client/examples/slurm.conf.simple.gz .`
4. `gzip -d slurm.conf.simple.gz`

iv. Edit SLURM Configuration File

1. Open SLURM configuration file in text editor
2. `sudo nano /etc/slurm-llnl/slurm.conf`
3. Edit the corresponding sections of the file to match below.
 - a. `SlurmctldHost=Master(<Master ip>)`
 - b. `SelectType=select/cons_res`
 - i. `SelectTypeParameters=CR_Core`
 - c. `NodeName=Master NodeAddr=<Master ip> CPUs=4 State=UNKNOWN`

- i. `nodeName=<node 2 hostname>
NodeAddr=<node 2 ip> CPUs=4
State=UNKNOWN`
- ii. `nodeName=<node 3 hostname>
NodeAddr=<node 3 ip> CPUs=4
State=UNKNOWN`
- iii. `nodeName=<node 4 hostname>
NodeAddr=<node 4 ip> CPUs=4
State=UNKNOWN`
- d. `PartitionName=mycluster Nodes=RPI[2-4]
Default=YES MaxTime=INFINITE State=UP`

v. Configure Resource Access for Jobs

1. Create the file `cgroup.conf`
2. `sudo nano /etc/slurm-llnl/cgroup.conf`
3. Add the following to the empty file
 - a. `CgroupMountpoint="/sys/fs/cgroup"`
 - i. `CgroupAutomount=yes`
 - ii. `CgroupReleaseAgentDir="/etc/slurm-llnl/cgroup"`
 - iii. `AllowedDevicesFile="/etc/slurm-llnl/cgroup_allowed_devices_file.conf"`
 - iv. `ConstrainCores=no`
 - v. `TaskAffinity=no`
 - vi. `ConstrainRAMSpace=yes`
 - vii. `ConstrainSwapSpace=no`
 - viii. `ConstrainDevices=no`
 - ix. `AllowedRamSpace=100`
 - x. `AllowedSwapSpace=0`
 - xi. `MaxRAMPercent=100`
 - xii. `MaxSwapPercent=100`
 - xiii. `MinRAMSpace=30`
4. The following commands will configure cgroup kernel isolation, a Linux feature that will restrict the system resources that jobs have access to.
 - a. `sudo nano /etc/slurm-llnl/cgroup_allowed_devices_file.conf`
5. Add the following to the empty file
 - a. `/dev/null`
 - i. `/dev/urandom`
 - ii. `/dev/zero`
 - iii. `/dev/sda*`
 - iv. `/dev/cpu/*/*`
 - v. `/dev/pts/*`
 - vi. `/clusterfs*`

vi. Copy The Completed Configuration Files to Shared Storage

1. `sudo cp slurm.conf cgroup.conf
cgroup_allowed_devices_file.conf /clusterfs`
 - a. `sudo cp /etc/munge/munge.key /clusterfs`

vii. Starting SLURM Controller

1. Start MUNGE
 - a. `Sudo systemctl enable munge`
 - b. `Sudo systemctl start munge`
2. Start SLURM Daemon
 - a. `Sudo systemctl enable slurmd`
 - b. `Sudo systemctl start slurm`
3. Start SLURM Controller Daemon
 - a. `Sudo systemctl enable slurmctld`
 - b. `Sudo systemctl start slurmctld`

b. Compute Node Configuration

- i. Perform the following steps for each node you wish to add to the cluster.
- ii. Install the SLURM Client
 1. `Sudo apt install slurmd slurm-client -y`
- iii. Update /etc/hosts file to include all nodes not already listed, excluding the node currently being configured. See example below.
 1. This particular example assumes the node is RPI2.
 - a. `<Master ip> Master`
 - b. `<RPI3 ip> RPI3`
 - c. `<RPI4 ip> RPI4`
 - d. Once the file has been updated to include all other node details this step is complete.
- iv. Update the node with the configuration file from shared storage
 1. `sudo cp /clusterfs/munge.key
/etc/munge/munge.key`
 2. `sudo cp /clusterfs/slurm.conf /etc/slurm-
llnl/slurm.conf`
 3. `sudo cp /clusterfs/cgroup* /etc/slurm-llnl`
- v. Start Munge Services
 1. `sudo systemctl enable munge`
 2. `sudo systemctl start munge`
- vi. Start SLURM Daemon
 1. `sudo systemctl enable slurmd`

2. `sudo systemctl start slurm`

vii. Test Munge

1. `ssh pi@Master munge -n | unmunge`
2. Go over all previous steps in the Compute Node Configuration section again if there are any errors to ensure that no mistakes were made.

c. Test SLURM

- i. The following steps are run on the Master node.
- ii. Replace X with the number of nodes that are currently running.
- iii. `srun --nodes=X hostname`
- iv. If the test is successful then all active nodes will output their name.

13. Install OpenMPI

- a. Run the following command on the master node.
- b. `sudo srun --nodes=3 apt install openmpi-bin openmpi-common libopenmpi3 libopenmpi-dev -y`

14. Install Python

- a. Run the following commands on the master node. These will install all necessary dependencies for Python.
- b. `sudo apt install -y build-essential python-dev python-setuptools python-pip python-smbus libncursesw5-dev libgdbm-dev libc6-dev zlib1g-dev libsqlite3-dev tk-dev libssl-dev openssl libffi-dev`
- c. `srun --nodes=<total number of nodes in cluster> sudo apt install libatlas-base-dev;` (The nodes can't run Python without this)
- d. Navigate to the /clusterfs directory and create a new directory called 'build'.
- e. `cd /clusterfs && mkdir build && cd build`
- f. Download the Python package from the internet.
- g. `wget https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tgz`
- h. Decompress the .tgz file.
- i. `tar xvzf Python-3.7.3.tgz`
- j. Navigate to the unpacked Python directory and create a new directory called /clusterfs/usr, then configure the Python install with PIP and ensure that it's installed to /clusterfs/usr. These steps may take some time to run.
- k. `cd Python-3.7.3`
- l. `mkdir /clusterfs/usr`
- m. `./configure \ --enable-optimizations \ --prefix=/clusterfs/usr \ --with-ensurepip=install`
- n. `make`
- o. `make install`

- p. Use the following command to verify the Python installation. If successful each node will output “Hello”.
- q.

```
srun --nodes=<number of nodes in cluster> /clusterfs/usr/bin/python3 -c "print('Hello')"
```
- r. This following command verifies the PIP installation.
- s.

```
srun --nodes=1 /clusterfs/usr/bin/pip3 --version
```
- t. If the last two commands work, Python has been successfully installed.

15. Installing John The Ripper

- a. John The Ripper has MPI support built into it, it just needs to be enabled with `./configure`. Because of this it only needs to be installed on the master node as John can hand out jobs to the nodes even if they don't have John installed.
- b. The process of installing John The Ripper is similar to that of Python. First, run the following commands to create an install location for John.
- c.

```
cd /clusterfs && mkdir john && cd john
```
- d. Download the John Jumbo package.
- e.

```
wget https://www.openwall.com/john/k/john-1.9.0-jumbo-1.tar.gz
```
- f. Unpack the tarball.
- g.

```
tar xvzf john-1.9.0-jumbo-1.tar.gz
```
- h. Navigate to the unpacked John directory and enable MPI support before proceeding with the installation.
- i.

```
cd john-1.9.0-jumbo-1/src/
```
- j.

```
./configure --enable-mpi
```
- k.

```
make
```
- l.

```
make install
```
- m. Verify the installation by running the following command. If the command returns no errors the John installation has been completed successfully.
- n.

```
srun --ntasks=7 mpiexec -n 7 /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john --test
```

How to Write Cluster Scripts

This section explains how to create scripts for the cluster, how to control the cluster and some useful commands for the cluster.

If the NFS share (`/clusterfs`) has not been mounted on the nodes once the cluster is power on, SSH (Username = “pi”, password = “raspberr”) into every node and enter the following command:

```
sudo mount -a
```

This mounts all filesystems mentioned in `fstab`, specifically the NFS.

The next step is on the master, run the following command:

```
sudo scontrol update nodename=RPI[1-4] state=IDLE
```

This changes all the SLURM states to IDLE (i.e up). If more nodes have been added change 1-4 to the correct number of nodes in the cluster.

With this done the cluster should be up and running, if not refer to the SLURM documentation or for any other errors such as OpenMPI errors or John the Ripper refer to the relevant documentation or the useful commands section.

The cluster can then be used to crack passwords. To do this you can either use a SLURM batch script or the command SRUN. The following script, johnDemo.sh can be used as a reference. The --ntasks flag refers to the number of cores to use but there are many flags that can be used. For example, --nodes which refers to the number of nodes so --nodes 3 would use one core on 3 nodes so if there were 4 cores on the node it would only use one.

```
#!/bin/bash
#SBATCH --ntasks=7

cd $SLURM_SUBMIT_DIR

mpirun -n 7 /clusterfs/johnBuild/john-1.9.0-jumbo-1/run/john --
wordlist=/clusterfs/scripts/john/password.lst
/clusterfs/scripts/john/demo.txt
```

It is important to keep the mpirun n flag and the sbatch ntasks/nodes the same number.

mpirun must run John the Ripper in either SLURM batch file or srun to use John the Ripper with OpenMPI.

A SLURM batch file is then run using sbatch.

```
sbatch /clusterfs/scripts/john/johnDemo.sh
```

The command srun can be used to submit a job in real time using the same flags. Below is the srun variation of johnDemo.sh

```
srun --ntasks=7 mpirun -n 7 /clusterfs/johnBuild/john-1.9.0-jumbo-
1/run/john -wordlist=/clusterfs/scripts/john/password.lst
/clusterfs/scripts/john/demo.txt
```

Useful Commands

```
sinfo
```

Gives information about the cluster's current state and other information.

```
scontrol show node <node name>
```

Shows why a node is down.

```
sudo scontrol update nodename=<node name>[<range>] state=IDLE
```

Sets state of all nodes in the cluster to IDLE (i.e up).

```
htop
```

Shows all processes running on a device.

```
sudo mount -a
Mounts all devices in fstab (used for NFS).
```

```
srun --nodes=<number of nodes> hostname
Simple test for nodes.
```

```
srun --nodes=<number of nodes> /clusterfs/usr/bin/python3 -c
"print('Hello') "
Simple python test on cluster.
```

```
sudo dphys-swapfile swapoff
sudo nano /etc/dphys-swapfile
    Edit CONF_SWAPSIZE to desired amount
sudo dphys-swapfile setup
sudo dphys-swapfile swapon
```

Increase/Change swapfile (a swap file allows an OS to use hard disk space to simulate extra memory) on a node. Useful due to the limited memory on Raspberry Pi's