ANALYSING THE RASPBERRY PI AS A COST-EFFECTIVE SOLUTION FOR SMES

Cardiff University School of Computer Science and Informatics

ABSTRACT

This project will analyse the Raspberry Pi as a low energy consumption and low-cost solution for SMEs and start-ups. This will look at the Raspberry Pi vs a traditional server set up vs a cloud solution.

Rhys Connor
Supervisor: Martin Caminada
Moderator: Crispin Cooper

Acknowledgements

I would like to thank my supervisor Martin Caminada and my parents for all the support throughout the project. As well as all the support received from Cisco Systems, Country Connect in helping facilitate physical elements of the project and Dave Smith for providing layer 1 support on the UCS used in the project.

Management Summary

This section provides a summary discussing how the Raspberry-Pi (RPi) 4 can help their business save money as the cost of energy rises, globally. This report will focus on the utilisation of the RPi 4 within Small-Medium Enterprises (SMEs). This report will highlight that the RPi 4 can host business-critical services that would traditionally be hosted on a rackmount server or workstation.

Migrating services onto a RPi 4 over the traditional server used in this project can equate to up to £420 pounds per year, when migrating from a 2 server to 2 RPi 4 configuration. A configuration of 4 RPi 4's configured for redundancy will also run at less than 1/5th of the energy used by the traditional server. The cost savings mentioned above reference the electricity cost saved and do not include the savings on hardware costs.

If the business is too large for the business-critical services to be run, this report highlights some other practical uses of the RPi 4, like the ones listed below:

- Portainer host
- Systems monitor
- RPi replacing traditional desktops
- Stratodesk Client or similar

The RPi 4 will host the business-critical services tested in this project for less than 1/5th of the cost of Google Cloud Platforms lowest specification offering. This cost includes the RPi 4 hardware and electricity running cost calculated in the 'Cost To Run Analysis' section of the report. The RPi 4 will also run at 1/50th of the electricity cost of the traditional server tested in this project and has an initial hardware cost of less than 1/10th of the traditional server used in the project.

Table of Contents

Acknowledgements 2
Management Summary 3
Table of Contents 4
Table of Figures 6
Table of Acronyms 9
1 Introduction 10
Research Question 1 10
Research Question 2 10
Research Question 3 10
Research Question 4 10
2 Background 11
Explanation of Services 11
RPi 16
Traditional Server Set-up 17
Cloud 17
Hardware Costs of Each System 18
3 Implementation 19
Network Infrastructure 19
Docker Infrastructure 20
VM Infrastructure 21
Infrastructure as Code (IaC) 21
IaC Examples 24
4 Hardware and OS Decisions 25
Hardware Decisions 25
OS/Platform Decision 27
5 Raw Performance 33
Approach 33
RPi 3 35
RPi 4 38
UCS C220-M3S 42
Evaluation and Analysis 45
6 Services Tested 47
DNS 48
DHCP 52
Webserver 54

AD 63
NAS 66
7 Power Draw of Systems 74
Results Found 74
Evaluation and Analysis 78
8 Conclusion 81
9 Example Configurations 82
Pi-Hole Configuration on 2xRPi 4 Configuration 82
Redundant Webserver on 3xRPi 4 Configuration 84
Webserver and Business Critical Services on 4xRPi 4 Configuration 85
10 Future Work 86
11 Reflection on Learning 87
Challenges Faced 87
References 88

Table of Figures

Figure 1 – A Live/Production Pi-Hole dashboard using Pi-Hole's Dark Theme 11
Figure 2 - Pricing Structure for AdGuard (AdGuard, 2022) 12
Figure 3 – An example Honeynet from imperva.com (Imperva, 2022) 13
Figure 4 – A demo glass-isc-dhcp dashboard (Miles, 2020) 14
Figure 5 - Portainer Device Management Dashboard 15
Figure 6 - NoTouch Centre (Stratodesk, 2022) 16
Figure 7 Mechanical Drawing of Raspberry-Pi 4 to illustrate the size of the device (Raspberry Pi Foundation, 2022) 17
Figure 8 - A pricing table for Microsoft Azure (Microsoft Azure, 2022) 18
Figure 9 - Network Diagram of Home Configuration 19
Figure 10 - Network Diagram of DMZ Infrastructure for UCS 20
Figure 11 - Portainer Configuration for RPi 20
Figure 12 - Diagram of Portainer Config for RPi 20
Figure 13 - Provision Machine Bash Script Used 23
Figure 14 - Portainer Docker Compose Example 24
Figure 15 - Pihole Docker Compose Example 25
Figure 16 - Hardware Requirements for Server OS Installations 26
Figure 17 - Table Showing Hardware Requirements of Bitwarden (Bitwarden, Inc, 2022) 26
Figure 18 – Cisco UCS M3 Boot Screen 27
Figure 19 - docker-compose.yml file to Create a Pi-Hole Container 29
Figure 20 - Proxmox Enterprise Licensing Costs (Proxmox, 2022) 31
Figure 21 - Table of Hardware Requirements for VMWare ESXi (VMWare, 2022) 31
Figure 22 - Table of Hardware Requirements for Proxmox (Proxmox, 2022) 32
Figure 23 – Table from Veritis comparing CSPs (Veritis, n.d.) 33
Figure 24 - Physical Network Example of How iperf Test Was Performed 34
Figure 25 - First Run of defconfig Linux Kernel Compile on RPi 3 35
Figure 26 - Second Run of defconfig Linux Kernel Compile on RPi 3 36
Figure 27 - Final Run of defconfig Linux Kernel Compile on RPi 4 36
Figure 28 - Table of Linux Kernel Compile Times on RPi 3 37
Figure 29 - Boxplot Graph of RPi 3 Time to Compile Linux Kernel 37
Figure 30 - Result of Iperf Test on RPi 4 37
Figure 31 - Linux Kernel Compile Failed Attempt on RPi 4 38
Figure 32 - First Run of defconfig Linux Kernel Compile on RPi 4 39
Figure 33 - Second Run of defconfig Linux Kernel Compile on RPi 4 40
Figure 34 - Third Run of defconfig Linux Kernel Compile on RPi 4 40
Figure 35 - Table of Linux Kernel Compile Times on RPi 4 41
Figure 36 - Boxplot Graph of RPi 4 Time to Compile Linux Kernel 41
Figure 37 - Result of Iperf Test on RPi 4 41
Figure 38 - A Failed attempt to benchmark compiling the Linux Kernel on the UCS 42
Figure 39 - First Run of defconfig Linux Kernel Compile on UCS 42
Figure 40 - Second Run of defconfig Linux Kernel Compile on UCS 43
Figure 41 - Third Run of defconfig Linux Kernel Compile on UCS 43
Figure 42 - Table of Linux Kernel Compile Times on UCS C220-M3S 44
Figure 43 - Boxplot Graph of UCS C220-M3S Time to Compile Linux Kernel 44
Figure 44 - Result of Iperf Test on UCS C220-M3S 44
Figure 45 - Table Outlining Iperf Performance of Systems 45

Figure 46 – Chart to Show Iperf Results 45
Figure 47 - Status of RPi 3 Link When iperf Testing Performed 45
Figure 48 - Bar Chart Highlighting Standard Deviation Between Systems 46
Figure 49 - Bar Chart Highlighting the Average Time to Compile Across Systems 47
Figure 50 - DNS Benchmark GUI (Gibson Research Corporation, 2018) 48
Figure 51 - Table of Pi-Hole on RPi 4's Time to Resolve DNS Queries 49
Figure 52 - Table of Pi-Hole on UCS C220-M3S Time to Resolve DNS Queries 49
Figure 53 - Table of Virgin Media's Time to Resolve DNS Queries 49
Figure 54 - Tables of BT Time to Resolve DNS Queries 50
Figure 55 - Table of Google Time to Resolve DNS Queries 50
Figure 56 - Table of OpenDNS Time to Resolve DNS Queries 50
Figure 57 - Section of the /etc/dhcp/dhcpd.conf File used by isc-dhcp-server 52
Figure 58 - HTOP Output Running Dhammer DHCP Stress Test on RPi 4 53
Figure 59 - HTOP Output Running Dhammer DHCP Stress Test on UCS C220-M3S 53
Figure 60 - Screenshot of Webserver Main Page 54
Figure 61 - HTOP Output from Webserver Handling 400 Requests on RPi 4 55
Figure 62 - Example of CLI Output from Wrk Testing RPi 4 55
Figure 63 - Table of Results For 50 Connections on RPi 4 55
Figure 64 - Table of Results For 100 Connections on RPi 4 56
Figure 65 - Table of Results For 200 Connections on RPi 4 56
Figure 66 - Table of Results For 400 Connections on RPi 4 56
Figure 67 - Table of Results for 20 Connections on UCS C220-M3S Over a 24-Hour Period 57
Figure 68 - HTOP Output from Webserver Handling 50 Requests on UCS C220-M3S 57
Figure 69 - Example of CLI Output from Wrk Testing UCS C220-M3S 58
Figure 70 - Table of Results for 50 Connections on UCS C220-M3S 58
Figure 71 - Table of Results for 100 Connections on UCS C220-M3S 58
Figure 72 - Table of Results for 200 Connections on UCS C220-M3S 59
Figure 73 - Table of Results for 400 Connections on UCS C220-M3S 59
Figure 74 - Table of Results for 20 Connections on UCS C220-M3S Over a 24-Hour Period 60
Figure 75 - Web Requests for Country Connect Website 60
Figure 76 - Average Requests Per Second for Webserver Running on Each System 62
Figure 77 - Average Latency for Webserver Running on Each System 62
Figure 78 - Successful Connection of Client to AD DC 63
Figure 79 - HTOP Output from RPi 4 Whilst 1 User Is Logging onto Client Machine 64
Figure 80 - HTOP Output from UCS AD DC Whilst 1 User Is Logging onto Client Machine 65
Figure 81 - Table of AD Results for Both Systems 66
Figure 82 - Code Snippet from Script to Upload Files to NAS Using FTP and Python 67
Figure 83 – Code Snippet for Downloading from NAS 67
Figure 84 - Code Snippet for Uploading to NAS 67
Figure 85 - HDD Performance of Windows 10 Client for RPi 4 68
Figure 86 - HDD Performance of Hard Drive Attached to RPi 4 NAS 68
Figure 87 - HDD Performance of Windows 10 Client for UCS C220-M3S 69
Figure 88 - HDD Performance of Hard Drive Attached to UCS C220-M3S NAS 69
Figure 89- HTOP Output from NAS Transferring 1TB File from Windows 10 Client to RPi 4 NAS 69
Figure 90 - Table of RPi 4 NAS Transfer Times from Windows 10 Client to NAS 70
Figure 91 - Table of RPi 4 NAS Transfer Times from NAS to Windows 10 Client 70
Figure 92 - HTOP Output from NAS Transferring 1TB File from Windows 10 Client to UCS NAS 70
Figure 93 - Table of UCS C220-M3S NAS Transfer Times from Windows 10 Client to NAS 70

Figure 94 - Table of UCS C220-M3S NAS Transfer Times from NAS to Windows 10 Client 70
Figure 95 - Diagram of NAS Upload 71
Figure 96 - Diagram of NAS Download 71
Figure 97 - Table of Rate of Transfer Downloading from RPi 4 NAS 71
Figure 98 - Chart of Rate of Transfer Downloading File from RPi 4 NAS 72
Figure 99 - Table of Rate of Transfer Uploading to RPi 4 NAS 72
Figure 100 - Chart of Rate of Transfer Uploading File to RPi 4 NAS 72
Figure 101 - Table of Rate of Transfer Downloading from UCS C220-M3S NAS 72
Figure 102 - Chart of Rate of Transfer Downloading from UCS C220-M3S 73
Figure 103 - Table of Rate of Transfer Uploading to UCS C220-M3S 73
Figure 104 - Chart of Rate of Transfer Uploading File to UCS C220-M3S NAS 73
Figure 105 - Output of top Command from RPi 4 2GB 74
Figure 106 - Output of HTOP Command from RPi 4 4GB 75
Figure 107 - Table of Power Draw Readings Gathered from USB-C to USB-C Multimeter 75
Figure 108 - Image of USB-C to USB-C Multimeter Reading Idle power draw of RPi 4 76
Figure 109 - Multimeter Configuration to Check Accuracy of USB-C to USB-C Multimeter 77
Figure 110 - Table of Power Draw Readings Gathered from Outlet Multimeter 77
Figure 111 - Table of Power Draw Readings Gathered from Panduit G5 Web GUI 77
Figure 112 - Power Draw as Measured from Panduit Web GUI 78
Figure 113 - Power Draw as Measured from CIMC 78
Figure 114 - Graph Illustrating the Power Draw of Systems 79
Figure 115 - Table Comparing Average Time to Compile on Systems 79
Figure 116 - Table Comparing Idle Power Draw of Systems 80
Figure 117 - Table Outlining Energy Usage of RPi 4 in Idle Hours 80
Figure 118 - Table Outlining Energy Usage of UCS C220-M3S in Idle Hours 80
Figure 119 - Table Outlining Power Usage of Systems in a Working Week 81
Figure 120 - Example Network Config for SME Using 2xRPi 4 for DNS 82
Figure 121 - Example Network Config for SME Using 3xRPi 4 for Company Webserver 84
Figure 122 - Example Network Config for SME Using 4xRPi 4 85

Table of Acronyms

Active Directory	AD
Active Directory Domain Controller	AD DC
Central Processing Unit	CPU
Cisco Integrated Management Console	CIMC
Cloud Service Provider	CSP
Command Line Interface	CLI
Customer Relationship Management	CRM
Demilitarized Zone	DMZ
Domain Name Services	DNS
Dynamic Host Configuration Protocol	DHCP
Error Correction Code	ECC
Extensible File Allocation Table	exFAT
File Allocation Table	FAT
File Transfer Protocol	FTP
Gigabyte	GB
Google Cloud Platform	GCP
Graphical User Interface	GUI
Infrastructure as Code	IaC
Intrusion Detection System	IDS
Intrusion Prevention System	IPS
Managed Service Provider	MSP
Megabits Per Second	Mbps
Network Attached Storage	NAS
Network File System	NFS
Network Interface Card	NIC
New Technology File System	NTFS
Not Suitable for Work	NSFW
Operating System	OS
Packet Capture/Packet Analysis	PCAP/PA
Power Distribution Unit	PDU
Random Access Memory	RAM
Raspberry-Pi	RPi
Redundant Array of Inexpensive Disks	RAID
Samba	SMB
Serial Advanced Technology Attachment	SATA
Serial Attached SCSI	SAS
Single Board Computer	SBC
Small Computer Systems Interface	SCSI
Small-Medium Enterprises	SMEs
Unified Computing System	UCS
Value Added Tax	VAT
Virtual Central Processing Unit	vCPU
Virtual Desktop Infrastructure	VDI
Virtual Hard Disk Drive	vHDD
Virtual Machine	VM

1 Introduction

The objective of this project is to analyse various models of RPi's as low energy consumption and low-cost solution for Small-Medium Enterprises (SMEs) and start-ups. This research will look at the RPi, analyse its performance and running costs when compared to a more traditional server set up. This research will then compare the cost of these solutions to other technologies a company could also use. The project will also evaluate the reasons why companies may want to, or not want to use cloud or traditional server setups over a RPi.

This project aims to address the issue of rising energy costs and hardware costs for businesses by either initially hosting or migrating services from a traditional server or cloud infrastructure to a RPi or a cluster of RPi's. Both the running costs and the initial hardware costs (if applicable) will be investigated during the course of this project. Some of the services that will be tested on these systems will be Active Directory (AD), Domain Name Services (DNS), Dynamic Host Configuration Protocol (DHCP), Network Attached Storage (NAS) and a company Webserver.

Although some practical uses of the RPi for business have been covered in the past, these have been typically in-depth analysis of a single-use case of the RPi for a business. For example:

- Low-Cost network monitoring system (Maulana & Al-Khowarizmi, 2021)
- Low-Cost Real-Time System monitor (Nguye, et al., 2015)
- Intrusion Detection System (de la Cruz, et al., 2016)
- Low-Cost Small Business Brewing (Acácio de Andrade, et al., 2020)

The systems that will be analysed throughout this project are a RPi 4 Model B 2GB, RPi 4 Model B 4GB and a Cisco Unified Computing System (UCS) C220-M3S.

Depending on the time constraints of the project, services such as Self-Hosted Company CRM, Honeypot, and a Packet Capture/Packet Analysis (PCAP/PA) server may be investigated.

To analyse these systems there will be a compilation of metrics gathered; raw performance metrics, power drawn, and performance statistics of the services hosted on these devices. Initial assumptions are that the RPi will draw less power than traditional technologies but will also have significantly less performance. This may however be ideal for services such as AD which are not as resource intensive as PA.

The project can be broken down into the following research questions:

Research Question 1

What is the raw performance of the RPi vs server?

Research Question 2

What is the difference in the power drawn of the RPi vs server?

Research Question 3

Using the data pulled from Requirements 1 & 2 what is the comparative performance per watt of the RPi vs server?

Research Question 4

What is the performance of the RPi vs server when hosting the key requirement services listed below?

- AD

- DNS
- NAS
- Company Webserver
- DHCP

This project is intended to give a detailed insight into the tools and services that SMEs could utilise a RPi for without causing any impact on regular business operations.

2 Background

This section will outline the project and the items covered within this project, the hardware used, the services to test and platforms used to manage the systems and services.

Explanation of Services

Pi-Hole/DNS

Pi-Hole is a free open-source DNS sinkhole (Pi-Hole, 2022) This can also be referred to as a network-wide advertisement blocker and DNS forwarder. Pi-Hole can be utilised by SMEs and Start-ups as a tool to manage the websites that employees can access and filter out any websites that are Not Suitable For Work (NSFW). This can be beneficial as it allows the company to monitor all the devices that are trying to access NSFW sites and identify any sites that may need blocking in the future. See the below Pi-Hole dashboard example:

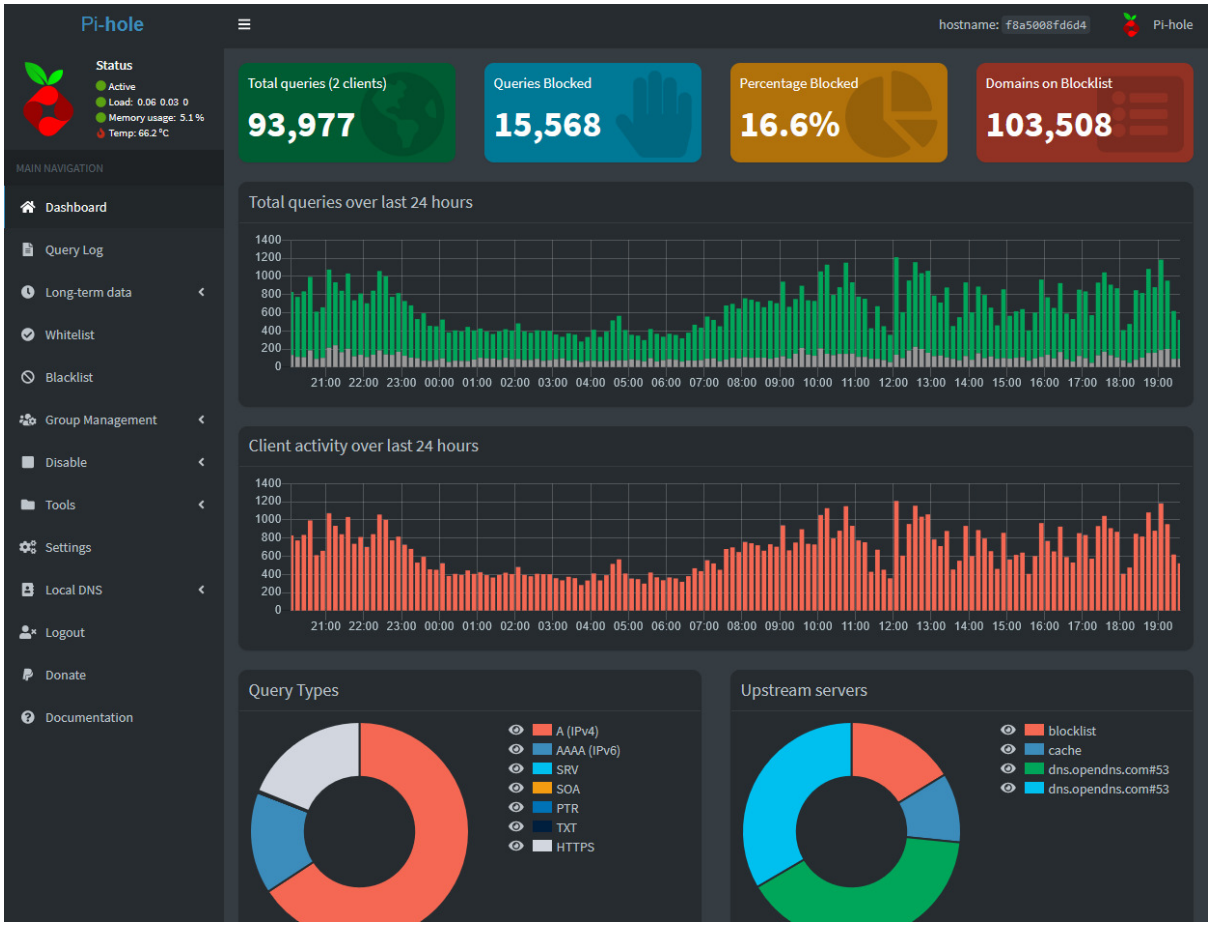

Figure 1 – A Live/Production Pi-Hole dashboard using Pi-Hole's Dark Theme

Note: This configuration of Pi-Hole is running on a docker container on a Raspberry-Pi 4 Model B 4GB

The decision to use Pi-Hole as opposed to the likes of AdGuard or other ad blockers is for the two reasons stated below:

- Network wide
- Open-Source

The network wide implementation of Pi-Hole allows it to be set up on one server then left to protect the network. Where other tools, in general tend to be device level so require an additional app to be installed on the user's device. As well as being network wide Pi-Hole is also Open-Source, this means that there is no licensing costs for the business to use this software. Unlike AdGuard which has the pricing structure shown below, their licensing only works on a number of devices basis from what can be seen on their webpage as opposed to the personal and commercial licencing used by some software vendors:

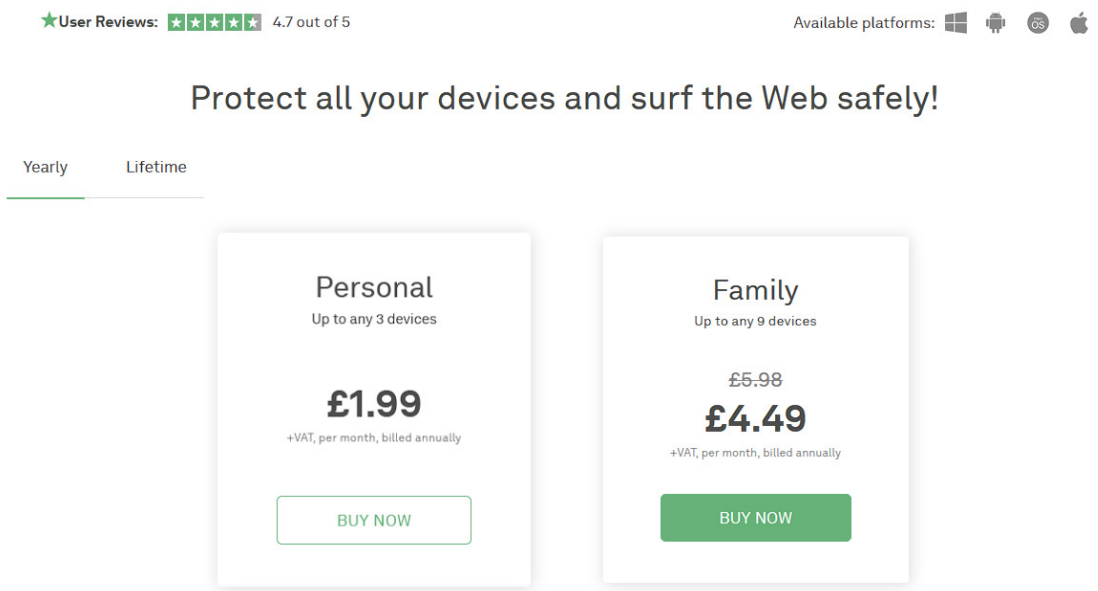

Figure 2 - Pricing Structure for AdGuard (AdGuard, 2022)

NAS

A NAS can be utilised in several ways for the types of businesses considered in this project. NAS can be utilised as a generally shared network drive for all users, set up as a backup location for user's documents and folders. NAS can be employed as a shared drive with folders for individual teams that are managed using user groups. Shared folders can be configured using SMB for Linux. With Windows this is supported natively, the file system must be formatted in a format that is readable by both Windows and Linux. For example, File Allocation Table (FAT), Extensible File Allocation Table (exFAT) and New Technology File System (NTFS).

Webserver

A webserver is utilised by businesses to host their customer-facing webpage and/or any custom internal tools that they may have built to streamline their internal business processes.

AD

AD is a Microsoft developed IdAM solution that has alternatives and applications that facilitates Linux servers to be the AD for the Windows clients. Microsoft defines AD as "*Active Directory stores*

information about objects on the network and makes this information easy for administrators and users to find and use. Active Directory uses a structured data store as the basis for a logical, hierarchical organization of directory information." (Microsoft, 2022). This is beneficial for all scales of business as it can allow for the creation of users and user groups. It also allows for the management of user permissions which can help a company to secure their network and devices from end-users installing malicious software. This can all be managed through a single AD server, instead of the local user approach where system administrators would have to go through each system when someone leaves to remove their system/service accesses. AD resolves this by being the central management system for the businesses users and their permissions.

Honeypot

A honeypot is a technological solution that imitates another server that would be of interest to a malicious attacker. This can be beneficial as these can be configured to send notifications to the IT team of the business when access is attempted. With many of these Honeypots' companies can also implement what is known as a Honeynet, this is a network of Honeypots that mimic a full company network. See the below diagram of an example Honeynet.

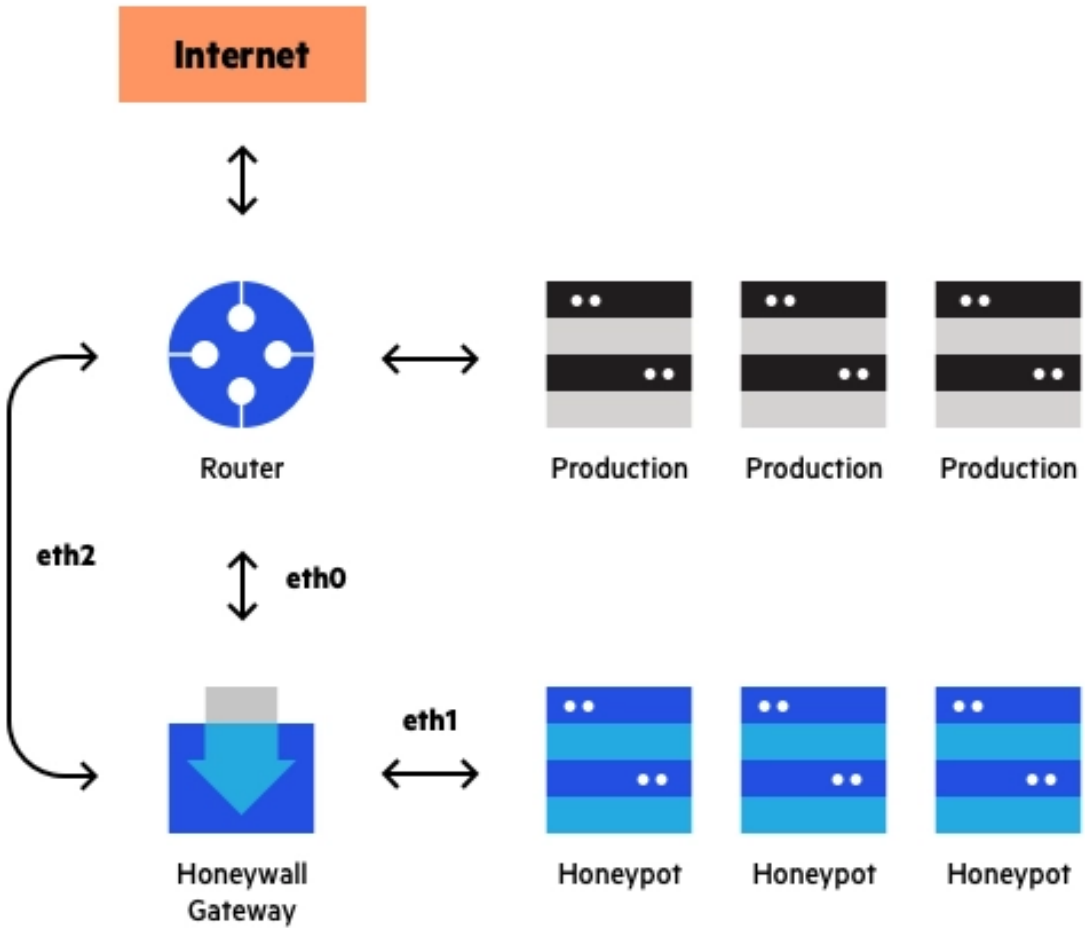

Figure 3 – An example Honeynet from imperva.com (Imperva, 2022)

DHCP

DHCP is a protocol used within networking to provide client devices connecting to the network an IP address and all the additional network configuration information required, such as, subnet mask, DNS server and default gateway. This is under the optional requirements for this project as a lot of

business routers and ISP-provided routers will host their DHCP server making this a non-essential requirement. However, it may be beneficial for businesses to host their DHCP server like Linux's isc-dhcp-server with a glass-isc-dhcp (Miles, 2020) web management portal. As this will allow them to have easier control over their DHCP leases and DHCP configuration than they may get with their ISP provided router. Below is an example of a glass-isc-dhcp dashboard:

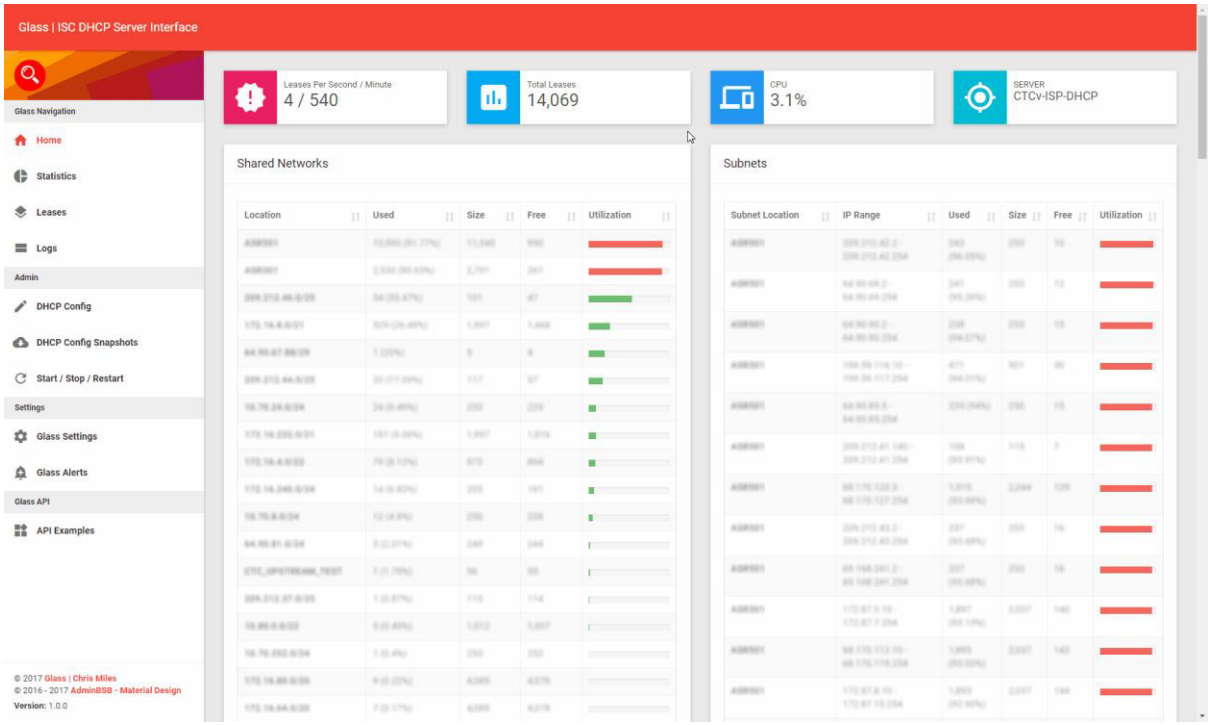

Figure 4 – A demo glass-isc-dhcp dashboard (Miles, 2020)

Customer Relationship Management (CRM)

CRM is a tool used by businesses to manage their customer relationships and even store information about customers who are potential leads. This tool will typically have a Webserver and Database element to it; however, a lot of CRM companies are now offering their services as a SaaS solution. Eliminating the need for a business to host this internally on their own infrastructure.

Intrusion Protection System (IPS) and Intrusion Detection System (IDS)

PCAP/PA is often used within systems to perform IPS and IDS for the LAN. This is a useful tool used by security analysts to monitor network traffic and can be used to identify any atypical and/or malicious network traffic. This can be helpful to identify if the business's internal network has been compromised or if suspicious activity is occurring on the network. This has been previously investigated using a RPi 3 Model B, this worked, however, the researchers noticed a limitation with the Random Access Memory (RAM) of the system (peaking at around 90% capacity) (de la Cruz, et al., 2016). This hardware limitation is where the RPi 4 Model B 4GB or 8GB models can further the throughput capacity as this system was tested with 5 clients and peaked around 29Mbps of network traffic.

Portainer

Portainer is "A centralised service delivery platform for containerized apps" (Portainer, 2022). This platform can be used as a web management Graphical User Interface (GUI) where users who are less experienced using the Command Line Interface (CLI) and the management of Docker containers through the CLI. The image below shows an example of the Portainer management GUI:

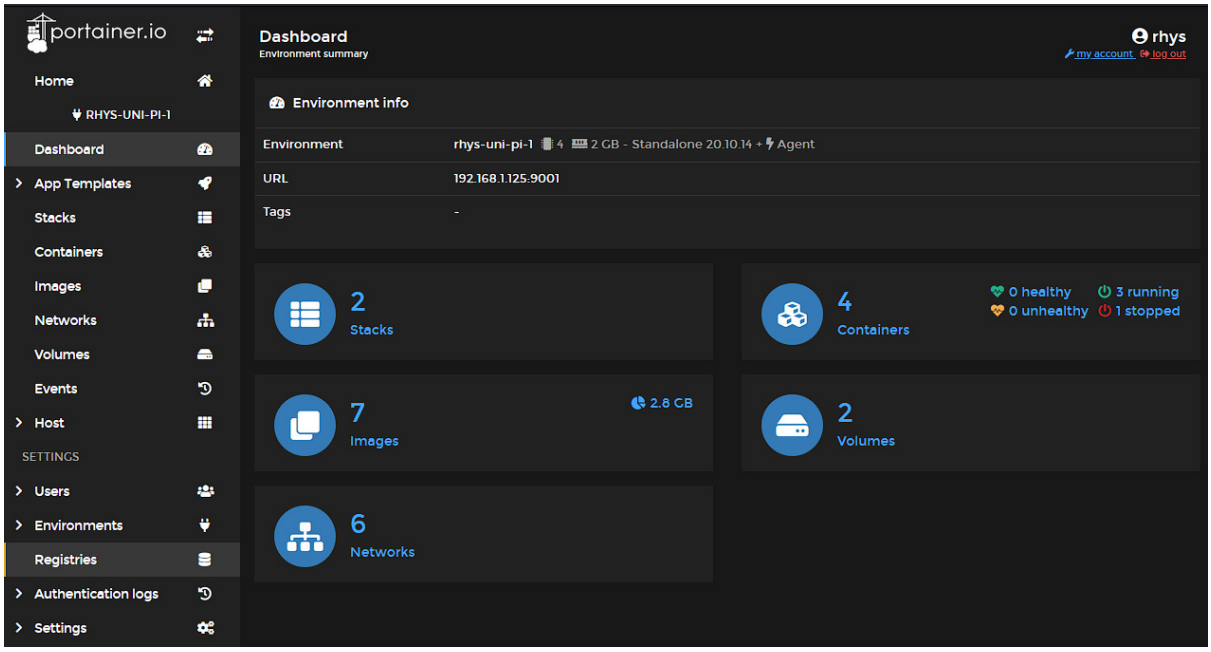

Figure 5 - Portainer Device Management Dashboard

Other Services of Note

Below is a list of services that are not likely to be tested within this project. They are noteworthy as they either already have RPi distributions of the services themselves, or they have low hardware requirements that would allow them to be run on a RPi.

- Stratodesk – Can be used by Managed Service Providers (MSPs)
- Systems monitor (Grafana)
- The use of the RPi as a Desktop – Power consumption difference to SFF PC for web browsing
- Private Cloud using Nextcloud, Own Cloud or similar services
- 3CX VoIP server
- Mail server
- NTP server

Citrix describes a Virtual Desktop Infrastructure (VDI) as “the hosting of desktop environments on a central server.” (Citrix, 2022), VDI Servers are typically used in a variety of use cases as outlined below (VMWare, 2022):

- Remote Work
- Bring Your Own Device
- Task or Shift Work

VDI’s for SMEs can be used by paying an MSP like Country Connect Ltd to host the SMEs hosted desktop infrastructure meaning all the SME has to pay for is the service cost and the cost of electricity for the thin client. Alternatively, the business could host their own VDI with thin clients or RPi’s.

Stratodesk is a tool that can be utilised by MSPs to manage clients connecting to a VDI with Stratodesk's own NoTouch Operating System (OS) which is Linux based and allows for the MSP to manage all thin clients and their configurations through one browser. NoTouch OS also supports both x86 and ARM Central Processing Unit (CPU) architectures meaning Thin Clients and RPi’s can run this OS with Stratodesk offering their own RPi image for the NoTouch OS.

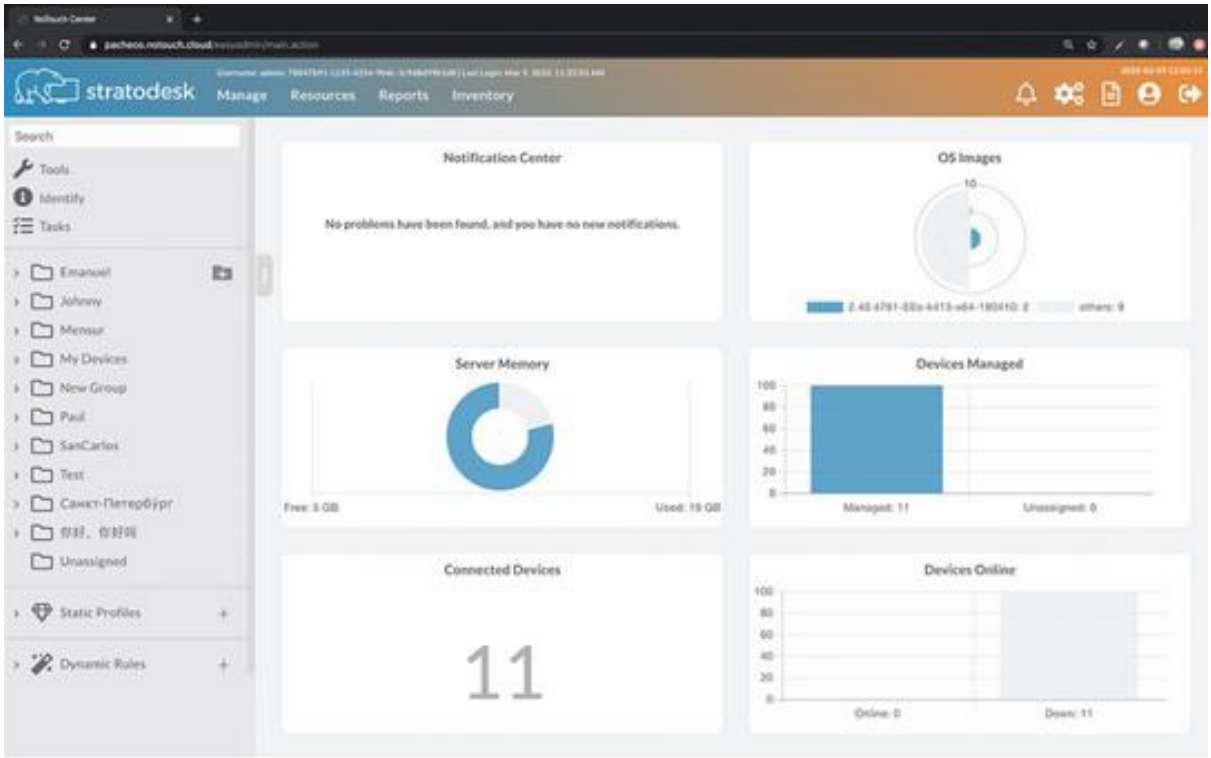

Figure 6 - NoTouch Centre (Stratodesk, 2022)

RPi

The RPi is a credit card sized computer that can be used for a wide range of applications from Robotics, Desktop computers, Interactive Museum exhibits and government call centres (Ltd, Raspberry-Pi. 2022). The aim of the Raspberry-Pi computers is to drive "down the cost of general-purpose computing…" (Raspberry Pi Foundation, 2022). The base cost of the RPi ranges from £34 for the 1GB Raspberry-Pi 4 Model B to £73.50 8GB Raspberry-Pi 4 Model B (The Pi Hut, 2022). Mechanical drawing of RPi 4 Model B below (Raspberry Pi Foundation, 2022):

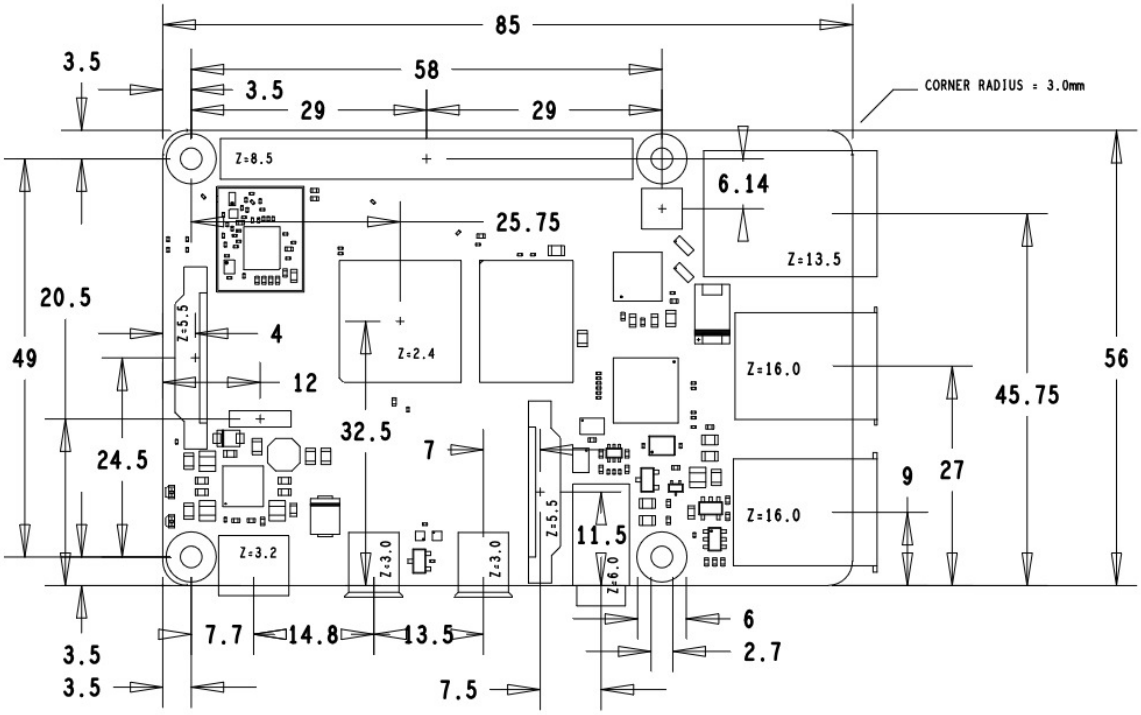

Figure 7 Mechanical Drawing of Raspberry-Pi 4 to illustrate the size of the device (Raspberry Pi Foundation, 2022)

Traditional Server Set-up

Lots of start-ups and SMEs they would "traditionally" utilise older refurbished enterprise hardware from companies such as Intelligent Servers (Intelligent Servers, 2022) and Bargain Hardware (Bargain Hardware, 2022). Both companies offer business class refurbished IT hardware like Servers, Workstations, and Desktop PC's/Laptops. This hardware can be brought at a wide range of prices starting from £125, including Value Added Tax (VAT) for a barebones server up to a £69995 for a pre-configured Dell MD3420 (Bargain Hardware, 2022).

Cloud

Over the past decade, the utilisation of Cloud infrastructures has become more and more prevalent within large enterprise. The subscription-based offering provided by the Cloud Service Provider (CSP), and its ease of scalability was beneficial to large companies as their hardware costs are higher than SMEs where they can also get exclusive contracts with cloud providers. These can however be costly through the subscription service option, see the below pricing for Microsoft Azure's mid-range Virtual Machine (VM) offering (Microsoft Azure, 2022):

Instance	Core(s)	RAM	Temporary storage	Pay as you go	1 year reserved	3 year reserved	Spot *	Add to estimate
A1 v2	1	2 GiB	10 GiB	£0.0269/hour	--	--	£0.0092/hour ~66% savings	+
A2 v2	2	4 GiB	20 GiB	£0.0568/hour	--	--	£0.0193/hour ~66% savings	+
A2m v2	2	16 GiB	20 GiB	£0.0740/hour	--	--	£0.0251/hour ~66% savings	+
A4 v2	4	8 GiB	40 GiB	£0.1188/hour	--	--	£0.0403/hour ~66% savings	+
A4m v2	4	32 GiB	40 GiB	£0.1554/hour	--	--	£0.0527/hour ~66% savings	+
A8 v2	8	16 GiB	80 GiB	£0.2487/hour	--	--	£0.0843/hour ~66% savings	+
A8m v2	8	64 GiB	80 GiB	£0.3264/hour	--	--	£0.1107/hour ~66% savings	+

Figure 8 - A pricing table for Microsoft Azure (Microsoft Azure, 2022)

Hardware Costs of Each System

Raspberry Pi

Current prices of the RPi 4 Model B can be found below:

- 1GB model £34
- 2GB model £43.50
- 4GB model £54
- 8GB model £73.50

These prices include VAT and reflect the price of the RPi 4 Model B as of February 2022 on the reseller site The Pi Hut (The Pi Hut, 2022). Although the RPi 3 is no longer commercially available, the starter kits of these models can be typically found for £40-60 on resale sites such as eBay, Facebook Marketplace, and Gumtree.

Cisco UCS C220-M3

A similar model of Cisco UCS C220-M3S like the one used during this project can be purchased as refurbished units for sale on sites like IT in Stock for £760 Ex VAT or £912 including VAT (IT in Stock, 2022).

GCP

Although there are no initial hardware costs associated with the GCP, the substantial subscription cost associated with this technology negates the hardware cost in a lot of instances.

3 Implementation

Network Infrastructure

Home

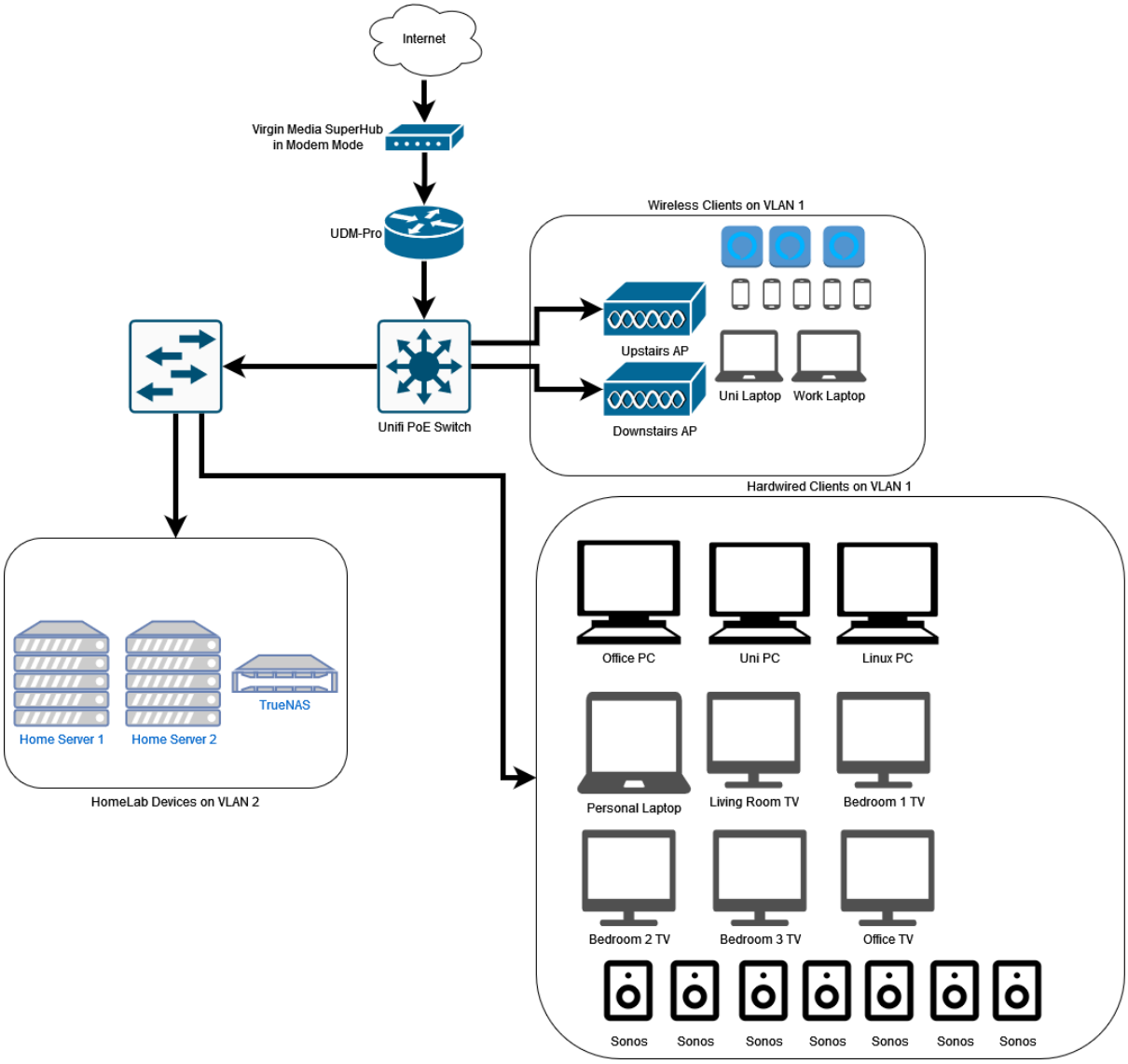

Figure 9 - Network Diagram of Home Configuration

The network configuration used for the RPi's included network segregation with the home devices, IoT devices and Servers all sitting on 3 separate VLANs outlined below:

- VLAN 1 – 192.168.1.x/24
- VLAN 2 – 192.168.2.x/24
- VLAN 3 – 192.168.3.x/24

The RPi's will operate on VLAN 3, with Docker configured with a MAC VLAN to allow for separate services to operate on different IP addresses.

Demilitarized Zone (DMZ)

The figure below outlines the network infrastructure used for the traditional server setup.

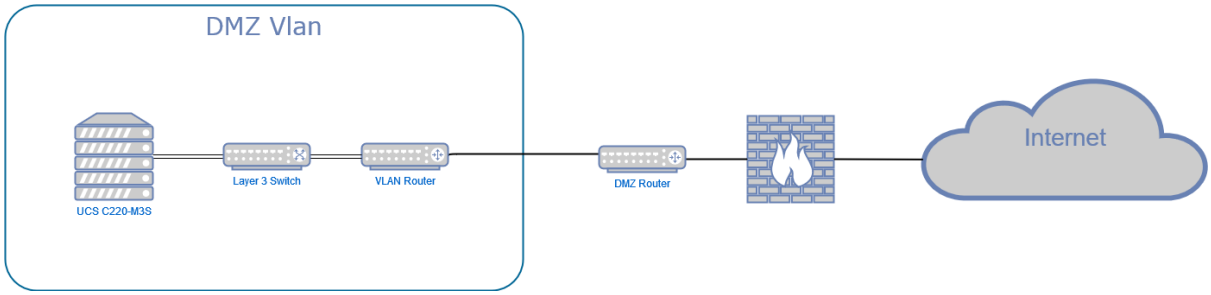

Figure 10 - Network Diagram of DMZ Infrastructure for UCS

Docker Infrastructure

For the services tested these will be run using docker compose and Portainer to manage the containers running.

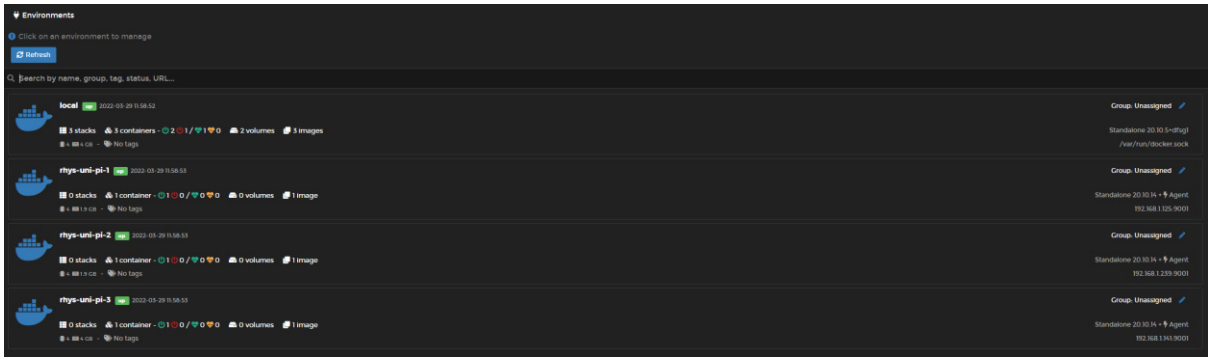

Figure 11 - Portainer Configuration for RPi

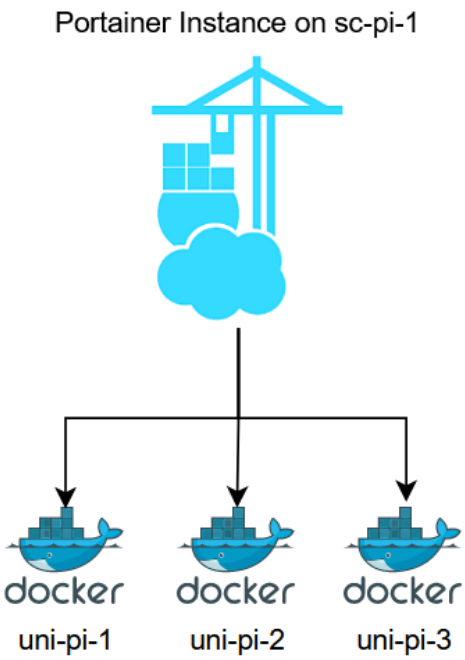

Figure 12 - Diagram of Portainer Config for RPi

The instance of Portainer is running within a Docker container on sc-pi-1, As well as Portainer sc-pi-1 is also running Pi-Hole in a container.

The command below was used to configure the MacVLAN network to allow different containers to use different IP addresses to the host:

- sudo docker network create -d macvlan --subnet=192.168.1.0/24 --gateway=192.168.1.1 -o parent=eth0 sc-net

VM Infrastructure

For the traditional server Proxmox and VMs were used to test the services. This allows for more direct comparison of the server vs the RPi 4 as the VMs that will be hosting the services will have the following resources allocated to them:

- 4 vCPU Cores
- 2 or 4GB RAM
- 32GB vHDD

The system specifications above match that of the RPi 4, This will allow for comparisons to be drawn about the electricity costs of the server running X number of VMs vs a cluster of RPi 4's.

Infrastructure as Code (IaC)

This section outlines the IaC scripts and processes used for each of these.

Docker

Docker was used to run some of the services outlined above, because the declarative approach meant that the state wanted from the system is declared within the docker-compose.yml file. This is often hardware agnostic, provided the service to be run is supported on both. RedHat describe the declarative approach to IaC as the following "A declarative approach defines the desired state of the system, including what resources you need and any properties they should have, and an IaC tool will configure it for you. " (RedHat, 2020). This ensures that the service state is the same on all systems tested.

Portainer

To manage the Docker hosts and the containers running on them, Portainer was used with one central Portainer Host and Portainer agents being installed on each host. The command displayed below was used to install the Portainer agent on these (Portainer, 2022):

```
$ sudo docker run -d -p 9001:9001 --name portainer_agent --restart=always -v
  /var/run/docker.sock:/var/run/docker.sock -v
  /var/lib/docker/volumes:/var/lib/docker/volumes portainer/agent:2.9.3
```

The Portainer host was set up using the command below (Portainer, 2022):

```
$ sudo docker run -d -p 8000:8000 -p 9443:9443 --name portainer \
  --restart=always \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v portainer_data:/data \
  portainer/portainer-ce:2.9.3
```

Using Portainer you also gain access to a whole repository of application templates, there is also the opportunity to create and store custom templates. The custom templates allow for granular control of the containers through a web GUI. Unlike Docker which uses CLI as its management interface. Using these templates will also ensure easy reproducibility, scalability and consistency.

Bash

In IaC, a Bash script would be considered the imperative approach to IaC. The imperative approach involves the script/code specifying the steps required to achieve the desired state of the system. This means that all the steps required, need to be clear and executable by the system. For example, if the script/code specified yum as the package manager and this was being executed on a basic install of Debian, this would not work as Debian installs use apt as the default package manager. For the purpose of this project, a Bash script was used to provision the machines. Setting these systems up with Docker, Docker Compose and adding the Portainer agent to the system.

```
sudo apt-get update

sudo apt-get install pip -y

sudo apt-get install ca-certificates curl gnupg lsb-release wget -y

#install docker
curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

#Install docker-compose using pip
sudo pip install --upgrade pip

sudo pip install docker-compose

#Check docker is installed and get the Portainer agent
d=$(docker --version)
if [[ $? != 0 ]]; then
    echo "Command failed."
elif [[ $d ]]; then
    echo "Docker is installed"
    #Comment out the below line to not run the portainer agent
    sudo docker run -d -p 9001:9001 --name portainer_agent --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v
/var/lib/docker/volumes:/var/lib/docker/volumes portainer/agent:2.9.3
    #Uncomment the below to add a MacVLAN to the docker config - Change the
subnets to match your use case
    #sudo docker network create -d macvlan --subnet=192.168.1.0/24 --
gateway=192.168.1.1  -o parent=eth0 sc-net
else
    echo "Docker is not installed"
fi

#Check that docker compose is installed
dc=$(docker-compose --version)
if [[ $? != 0 ]]; then
    echo "Command failed."
elif [[ $dc ]]; then
    echo "Docker Compose is installed"
else
    echo "Docker Compose is not installed"
fi

sudo apt upgrade -y

reboot
```

Figure 13 - Provision Machine Bash Script Used

As the OS used was Debian or RPi OS which is Debian based, the bash script above could be used to setup the machines with Docker and Docker Compose as well as update the OS to the latest version.

For Debian instances run on the UCS C220-M3S, Sudo needed to be installed prior to running the script, or all instances of sudo needed to be deleted from the script. The script can be run in two ways, the first way is as shown below:

```
$ sudo apt-get install git
$ git clone https://github.com/rhys909/FYP--Provision-Script.git
$ cd FYP—Provision-Script
$ sudo chmod a+x provision-machine.sh
$ sudo ./provision-machine.sh
```

Or alternatively the following commands can be used:

```
$ touch provision-machine.sh
$ sudo nano provision-machine.sh
$ <COPY THE CONTENTS INTO NANO>
$ sudo chmod a+x provision-machine.sh
$ sudo ./provision-machine.sh
```

For the base Debian installs, where sudo is not installed the sudo will need to be removed from the front of the commands. Alternatively, the command 'apt-get install sudo' will need to be executed prior to the commands above.

IaC Examples

Portainer

```yaml
version: '3'

services:
  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    restart: unless-stopped
    security_opt:
      - no-new-privileges:true
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./portainer-data:/data
    networks:
      steep-corner-net:
        ipv4_address: 192.168.1.2
    ports:
      - 9000:9000

networks:
  steep-corner-net:
    external:
      name: sc-net
```

Figure 14 - Portainer Docker Compose Example

Pi Hole

```
version: "3"
# Script taken from the below github repo
# More info at https://github.com/pi-hole/docker-pi-hole/ and https://docs.pi-hole.net/
services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "67:67/udp"
      - "80:80/tcp"
      - "443:443/tcp"
    environment:
      TZ: 'Europe/London'
      WEBPASSWORD: '@R9a=+v(Xkg9%,[W'
    # Volumes store your data between container upgrades
    networks:
      steep-corner-net:
        ipv4_address: 192.168.1.3
    volumes:
      - './etc-pihole/:/etc/pihole/'
      - './etc-dnsmasq.d/:/etc/dnsmasq.d/'
    # Recommended but not required (DHCP needs NET_ADMIN)
    #   https://github.com/pi-hole/docker-pi-hole#note-on-capabilities
    cap_add:
      - NET_ADMIN
    restart: unless-stopped

networks:
  steep-corner-net:
    external:
      name: sc-net
```

Figure 15 - Pihole Docker Compose Example

4 Hardware and OS Decisions

Hardware Decisions

The RPi 4 B was the first edition of RPi to offer varying RAM sizes. The RPi 4 was also the first iteration of the RPi to include separate lanes to the CPU for network and USB. Prior to the RPi 4 the USB and network shared one CPU lane.

The RPi 4 has the following hardware outlined on their RPi 4 datasheet (Raspberry Pi LTD, 2019):

- Quad core 64-bit ARM-Cortex A72 @ 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options

- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- Supports dual HDMI display output up to 4Kp60
- IEEE 802.11 b/g/n/ac Wireless LAN
- Gigabit Ethernet port (supports PoE with add-on PoE HAT)

Comparatively, the RPi 3 has the following hardware outlined on their RPi 3 datasheet (Raspberry Pi LTD, n.d.):

- Quadcore Broadcom Cortex-A53 @ 1.4GHz
- 1GB LPDDR2 SDRAM
- IEEE 802.11.b/g/n/ac wireless LAN
- Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps)
- H.264, MPEG-4 decode (1080p30)
- H.264 encode (1080p30)

The main differences between these systems are the RAM, CPU and network. The CPU processing power according to Pass Mark almost doubles from the RPi 3 to the RPi 4. The score of the RPi 3's Cortex A53 gets a CPU score of 357 compared to the Cortex A72 of the RPi 4's CPU score of 666 (PassMark Software, 2022). These scores are calculated from other user's submissions after running the Pass Mark benchmarking software.

Although it is already known that the RPi will be less powerful than the traditional server, it is also known that Linux has lower hardware requirements than a Windows server instance (see the below table).

Hardware Requirements	Debian 11 Server	Windows Server
CPU	1GHz	1.4GHz
RAM	512MB	512MB (2GB with Desktop Experience installed)
HDD	10GB	32GB

Figure 16 - Hardware Requirements for Server OS Installations

The above table must also be considered alongside the hardware requirements of the services you wish to run on the server. The table below shows an example of the hardware requirements for Bitwarden, a password manager that can be hosted on a local server:

	Minimum	Recommended
Processor	x64, 1.4GHz	x64, 2GHz dual core
Memory	2GB RAM	4GB RAM
Storage	10GB	25GB
Docker Version	Engine 19+ and Compose 1.24+	Engine 19+ and Compose 1.24+

Figure 17 - Table Showing Hardware Requirements of Bitwarden (Bitwarden, Inc, 2022)

For this project the 'Traditional' server instance will be demonstrated using a Cisco UCS C220-M3S. This is a 1U rackmount server with 2x Intel X79 CPU sockets, up to 512GB DDR3 Error Correction Code (ECC) RAM, up to four 3.5" or up to eight 2.5" SAS/SATA drives, 2 PCIe Gen 3 slots and 2 1GE LAN interfaces on the motherboard (Cisco Systems Inc, 2017). The 2 LAN interfaces will typically be connected with one allowing access to the servers Cisco Integrated Management Console (CIMC) platform. This allows for remote configuration and management of the server. The second LAN port is utilised by the OS/Hypervisor installed on the system. For this project the hypervisor used will be

ESXi installed onto the UCS C220-M3S. The hardware configuration of the UCS C220-M3S in this project is as follows:

- 2x Intel Xeon E5-2643 8 Core 16 thread CPUs @ 3.3GHz
- 16x 8GB 1600MHz DDR3 ECC RAM
- 1x LSI 9271-8i MegaRAID SAS Host Bus Adapter
- 4x 280GB 7200RPM 6Gb/s Toshiba HDD
- 4x 280GB 7200RPM 6Gb/s Seagate HDD

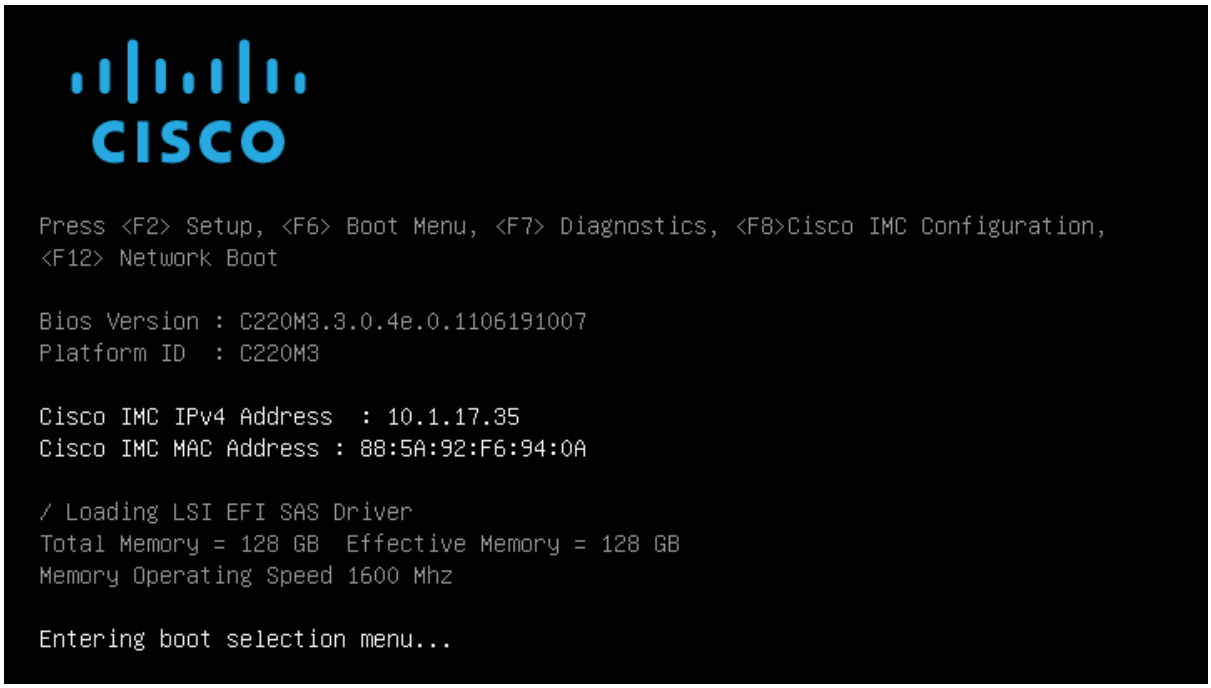

Figure 18 – Cisco UCS M3 Boot Screen

The Cisco UCS C220-M3S, outlined above, was used in the place of the traditional server as one was available for this projects use at Cisco Systems, Green Park, Reading Lab DMZ. This allows remote access to this UCS. With this equipment available, the use of Panduit G5 IP managed Power Distribution Unit's (PDUs) gives greater insight into the power drawn by the server. Using the Cisco labs DMZ, access was available to the UCS without access to Cisco's internal corporate network. CISCO employees were available to provide layer 1 support should issues arise.

OS/Platform Decision

This section will explain the platforms/OS used and why these platforms were chosen.

- Debian
- Raspian
- Proxmox/ESXi
- GCP

Hypervisors such as Proxmox/ESXi have been used to accurately represent both what is being utilised in business and what research suggests is best practice for servers. A hypervisor, also referred to as a Virtual Machine Monitor, is a piece of software that is used to create, manage, and run VMs (VMWare, 2022). Hypervisors can either be run on top of the host OS, these are classified as Type 2 hypervisors. Type 1 hypervisors behave like a lightweight OS. Use of a hypervisor has become common practice as it allows for the server's resources to be split up into VMs that are

easier to increase the CPU cores, RAM, and HDD space. Virtualisation is more cost effective when compared to traditional server installations. Virtualisation allows for Systems Administrators and IT teams to increase the power of a server with the click of a few buttons as opposed to a barebones installation where the server could be down for hours whilst old CPUs, RAM modules and SSD/HDDs are removed, and new models added. The VM approach on the other hand, can take a Systems Administrators and IT team 10-20 Minutes of downtime for the service which has a lot less impact on everyday business use than the hours that may be needed for a barebones upgrade (Jackson, et al., 2020).

Following the advent of VMs, subsequentially DevOps, another approach to hosting applications became more widespread. Containerization was popularised by Docker. Docker became widely used, popular and removed the argument between Developers and Operations. Where traditionally applications would sometimes not work on the production servers, after previously working on the developer's machine. Docker provided a platform for Developers to build applications and run within a Docker container. Safely knowing that this will be platform independent so this removes the previous issue, where applications would run on the Developers machine and not run in the production environment. Docker Compose was then developed as a tool to enable multiple container applications, a developer could write one docker compose file and deploy a Linux-Apache-MySQL-PHP stack for a web application. See the example below of the Docker Compose file used to deploy Pi-Hole in the project:

```yaml
version: "3"

# More info at https://github.com/pi-hole/docker-pi-hole/ and https://docs.pi-hole.net/
services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "67:67/udp"
      - "80:80/tcp"
      - "443:443/tcp"
    environment:
      TZ: 'Europe/London'
      WEBPASSWORD: 'INSERT-PASSWORD'
    networks:
      your-network:
        ipv4_address: 192.168.1.3
    # Volumes store your data between container upgrades
    volumes:
      - './etc-pihole/:/etc/pihole/'
      - './etc-dnsmasq.d/:/etc/dnsmasq.d/'
    # Recommended but not required (DHCP needs NET_ADMIN)
    #   https://github.com/pi-hole/docker-pi-hole#note-on-capabilities
    cap_add:
      - NET_ADMIN
    restart: unless-stopped

networks:
  your-network:
    external:
      name: name-of-your-docker-network
```

Figure 19 - docker-compose.yml file to Create a Pi-Hole Container

RPi OS

For this project the OS to be used for the RPi will be RPi OS Lite (64-bit) for the RPi 4 and used for the RPi 3 to test the raw performance of this system. This is the chosen OS as this is developed and maintained by the RPi Foundation, the version used for the testing is as follows (Raspberry Pi Foundation, 2022):

- Release date: January 28th, 2022
- System: 64-bit
- Kernel version: 5.10
- Debian version: 11 (bullseye)
- Size: 435MB

- SHA256 file integrity hash: d694d2838018cf0d152fe81031dba83182cee79f785c033844b520d222ac12f5

Testing of the Lite version would be used as this share's similarities to the Server releases of other Linux Distributions like Debian Server, Ubuntu Server etc.

Debian

The OS that will be used on the UCS C220-M3S is Debian 11, this is because this particular distro of Linux is what the RPi OS is built upon and derived from (as shown above in the RPi OS outline). As RPi OS is Debian based this makes it the most comparable to RPi OS in both commands and structure of the OS. With the RPi OS also being Debian 64bit and Debian based means that the commands run on a Debian machine will also be the same as the commands run on a RPi OS machine. One exception would be the leading sudo as this is not installed by default with a barebones Debian 11 Server install. Details of the version used are listed below (Debian Org, 2021):

- Release Date: December 18th, 2021
- Kernel Version: 5.10
- Size: 378MB
- SHA512 file integrity hash: c685b85cf9f248633ba3cd2b9f9e781fa03225587e0c332aef2063f6877a1f0622f56d44cf0690087b0ca36883147ecb5593e3da6f965968402cdbdf12f6dd74

Proxmox vs ESXi

Before deciding which vendors when selecting a hypervisor, the first consideration must be Hypervisor vs Barebones installation. It was decided that a hypervisor would be used as the utilisation of this allows for better resource scaling for the services operating on the VMs (Jackson, et al., 2020).

In the case of this project, it was concluded that Proxmox would be the most suitable technology as Proxmox is an Open-Source software so does not require licensing. Although it does not require licensing Proxmox also offer the licenses detailed below that cover support for the software if required:

Pick the Right Plan for your Team

15,000+ satisfied customers have a Proxmox VE subscription. Get your own in 60 seconds.

PREMIUM	STANDARD	BASIC	COMMUNITY
All you'll ever need	Most popular	For growing businesses	Starting out
€ 890/year & CPU socket	€ 445/year & CPU socket	€ 295/year & CPU socket	€ 95/year & CPU socket
Buy now	Buy now	Buy now	Buy now
Access to Enterprise repository	Access to Enterprise repository	Access to Enterprise repository	Access to Enterprise repository
Complete feature-set	Complete feature-set	Complete feature-set	Complete feature-set
Support via Customer Portal	Support via Customer Portal	Support via Customer Portal	Community support
Unlimited support tickets	**10 support tickets/year**	**3 support tickets/year**	
Response time: 2 hours* within a business day	Response time: 4 hours* within a business day	Response time: 1 business day	
Remote support (via SSH)	Remote support (via SSH)		

* Guaranteed first response time on critical support requests

All prices are net prices. VAT will be added, if applicable.

Download Subscription Agreement with FAQ

Figure 20 - Proxmox Enterprise Licensing Costs (Proxmox, 2022)

For ESXi VMWare show the following hardware requirements:

	Minimum	Recommended
CPU	Single socket with two cores	dual socket with four or more cores per CPU
RAM	4 GB	8 GB or more
Network	Single 1 GbE network adapter	Dual 1 GbE network adapters
Storage / OS Storage	Single 4 GB drive	Redundant drives
Shared Storage / VM Storage	NFS, iSCSI or Fibre Channel for virtual machine storage	NFS, iSCSI or Fibre Channel for virtual machine storage

Figure 21 - Table of Hardware Requirements for VMWare ESXi (VMWare, 2022)

Alternatively, the hardware requirements for Proxmox are not provided in the same way as VMWare ESXi. Instead of displaying the hardware requirements as a minimum and recommended in the way that VMWare do, Proxmox provide theirs as evaluation and recommended hardware requirements. The evaluation specs can be considered the minimum hardware required to get Proxmox up and running and the recommended being the specifications that Proxmox recommend for production deployments, these specifications are as follows:

	Minimum	Recommended
CPU	64bit	Intel EMT64 or AMD64 with Intel VT/AMD-V CPU flag
RAM	1 GB	2 GB for OS and Proxmox VE services. Plus, designated memory for guests. For Ceph or ZFS additional memory is required, approximately 1 GB memory for every TB used storage
Network	One Network Interface Card (NIC)	Redundant Gbit NICs, additional NICs depending on the preferred storage technology and cluster setup – 10 Gbit and higher is also supported
Storage / OS Storage	Hard Drive	Hardware Redundant Array of Inexpensive Disks (RAID) with batteries protected write cache ("BBU") or non-RAID with ZFS and SSD cache
Shared Storage / VM Storage	Hard Drive	For local storage use a hardware RAID with battery backed write cache (BBU) or non-RAID for ZFS. Neither ZFS nor Ceph are compatible with a hardware RAID controller. Shared and distributed storage is also possible.

Figure 22 - Table of Hardware Requirements for Proxmox (Proxmox, 2022)

Although it is physically possible to run Proxmox on an RPi 4, it was decided that for this project this wouldn't be used as the focus is on the energy saving that could be found when running business-critical services on a RPi 4 instead of the UCS C220-M3S.

GCP

GCP is one of the big 3 cloud providers, these are listed below:

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform

Comparing the offerings of these three providers in the table below (Figure 23) , it can be seen that GCP has a cheaper but similar offering to AWS at the entry level. It also offers higher CPU counts but less RAM in the high end at similar costs to both AWS and Azure.

Machine Type	AWS	Azure	GCP
Smallest Instance	An instance with 2 virtual CPUs and 8 GB RAM will cost you around USD69/month.	An instance with 2 virtual CPUs and 8 GB RAM will cost you around USD70/month.	Instance with 2 virtual CPUs and 8 GB RAM will cost you around USD52/month.
Largest Instance	Largest instance that includes 3.84 TB RAM and 128 vCPUs will cost you around USD 3.97/hour.	Largest instance that includes 3.89 TB RAM and 128 vCPUs will cost you around USD 6.79/hour.	Largest instance that includes 3.75 TB RAM and 160 vCPUs will cost you around USD 5.32/hour.

Figure 23 – Table from Veritis comparing CSPs (Veritis, n.d.)

These differences mean that GCP is good to use for the comparisons in this project. As the small businesses will likely use the systems on the lower end of the scale where GCP is more cost effective.

5 Raw Performance

Following the discussion of the CSP's and their offerings, this section will outline the raw performance of each system tested during this project. This will include how the metrics were gathered in the approach, the results of this testing and an evaluation of the raw performance of each system.

Approach

CPU/Memory

To stress-test the CPU and memory in the system the following tools can be used to test these:

- GeekBench
- Sysbench
- Hard Info
- Phoronix Test Suite

The test suite that will be used for this project will be Sysbench/GeekBench. Testing will also be performed using Phoronix Test Suite to compile Firefox and timing how long this takes to compile. Compiling an application such as Firefox or the Linux Kernel is a good raw performance test of a system as this process puts a lot of strain on both CPU and Memory in the system. This activity is also very close to the real-world use of a software build server that is utilised in a development environment. Compiling software such as the Linux Kernel, Google Chrome and Firefox is often also used for performance testing, when technology reviewers like LinusTechTips, Level1Techs and Gamers Nexus are reviewing technologies.

The commands listed below were used to install and run the raw performance benchmarks:

```
$ wget http://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-test-suite_7.8.0_all.deb
$ sudo apt install gdebi-core
$ sudo gdebi phoronix-test-suite_7.8.0_all.deb
$ phoronix-test-suite --version
$ phoronix-test-suite benchmark build-linux-kernel
```

Testing will initially start with the sysbench benchmarks, as these are the least time consuming and can give an immediate insight into the raw performance of individual components of each system. The next stage of testing will move onto the Linux Kernel compile as it is a real-world use and

stresses the whole system. As highlighted by Passmark Software's online comparison we can expect that the RPi 4 will perform about 1/10th of the performance of the UCS C220-M3S in CPU heavy workloads (Passmark Software, 2022), The link below gives a more detailed comparison of the RPi 4 CPU vs the CPU in the UCS used:

- https://www.cpubenchmark.net/compare/ARM-Cortex-A72-4-Core-1800-MHz-vs-Intel-Xeon-E5-2643/4078vs1217

The timed Linux kernel build provides 2 options of how to run the test:

- Defconfig
- Allmodconfig

The defconfig option builds the config file with the default settings (based on the systems architecture), where the allmodconfig builds a config file that makes as many parts of the Linux kernel as possible a module, this generates more strain on the cpu and takes significantly longer than the defconfig run of the Timed Linux Kernel Compile.

Once the results have been gathered the equation below was used to work out the standard deviation of the runs, the standard deviation will then be used to compare the consistency of each of the systems performance. The standard deviation calculation was done using the STDEV function within Excel and the equation taken from Microsoft docs outlining the method used by Excel (Microsoft, 2022).

$$\sigma = \sqrt{\frac{\sum(x - \bar{x})}{(n - 1)}}$$

Network

To test the network performance of the devices the physical network configuration below was used:

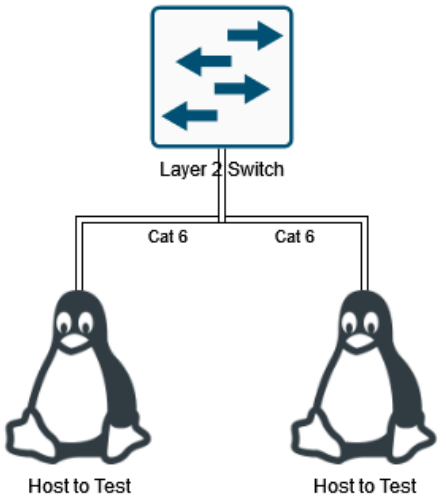

Figure 24 - Physical Network Example of How iperf Test Was Performed

To test the NIC on the devices the following commands were run:

Host A

- iperf -s

Host B

- iperf -c <IP ADDRESS OF HOST A>

In all instances the hardware of each device was connected into a gigabit switch over Cat6 cable which is rated up to gigabit speeds. This was used over Cat6a as none of the NICs used on any of the devices were capable of more than gigabit connectivity.

RPi 3

Results

Timed Linux Kernel Compile

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 26 Minutes [17:04 UTC]
        Running Pre-Test Script @^[[C^[[C^[[C

        Started Run 1 @ 16:41:00
        Running Interim Test Script @ 19:59:53
        Started Run 2 @ 20:00:19^[[C^[[A
        Running Interim Test Script @ 23:47:56
        Started Run 3 @ 23:48:23
        Running Post-Test Script @ 03:35:04

    Build: defconfig:
        11931.908607006
        13656.140014172
        13600.6888659

    Average: 13062.91 Seconds
    Deviation: 7.50%
```

Figure 25 - First Run of defconfig Linux Kernel Compile on RPi 3

https://openbenchmarking.org/result/2203196-FO-RPI3RUN1804

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 2 Hours, 45 Minutes [15:43 UTC]
        Running Pre-Test Script @ 12:59:33
        Started Run 1 @ 13:01:49
        Running Interim Test Script @ 16:42:59
        Started Run 2 @ 16:43:25
        Running Interim Test Script @ 20:27:39
        Started Run 3 @ 20:28:06
        Running Post-Test Script @ 00:13:09

    Build: defconfig:
        13268.182693005
        13452.620759964
        13501.715219021

    Average: 13407.51 Seconds
    Deviation: 0.92%
```

Figure 26 - Second Run of defconfig Linux Kernel Compile on RPi 3

https://openbenchmarking.org/result/2203205-FO-RPI3RUN2568

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 4 Hours, 27 Minutes [15:59 UTC]
        Running Pre-Test Script @ 11:32:56
        Started Run 1 @ 11:35:06
        Running Interim Test Script @ 15:11:49
        Started Run 2 @ 15:12:16
        Running Interim Test Script @ 18:58:17
        Started Run 3 @ 18:58:45
        Running Post-Test Script @ 22:49:32

    Build: defconfig:
        13002.81704092
        13560.407849073
        13846.389673948

    Average: 13469.87 Seconds
    Deviation: 3.18%
```

Figure 27 - Final Run of defconfig Linux Kernel Compile on RPi 4

https://openbenchmarking.org/result/2203201-FO-RPI3RUN3300

Run	Time To Compile
1	11932
2	13656
3	13601
4	13268
5	13453
6	13502
7	13003
8	13560
9	13846
Average	13313
Standard Deviation	571

Figure 28 - Table of Linux Kernel Compile Times on RPi 3

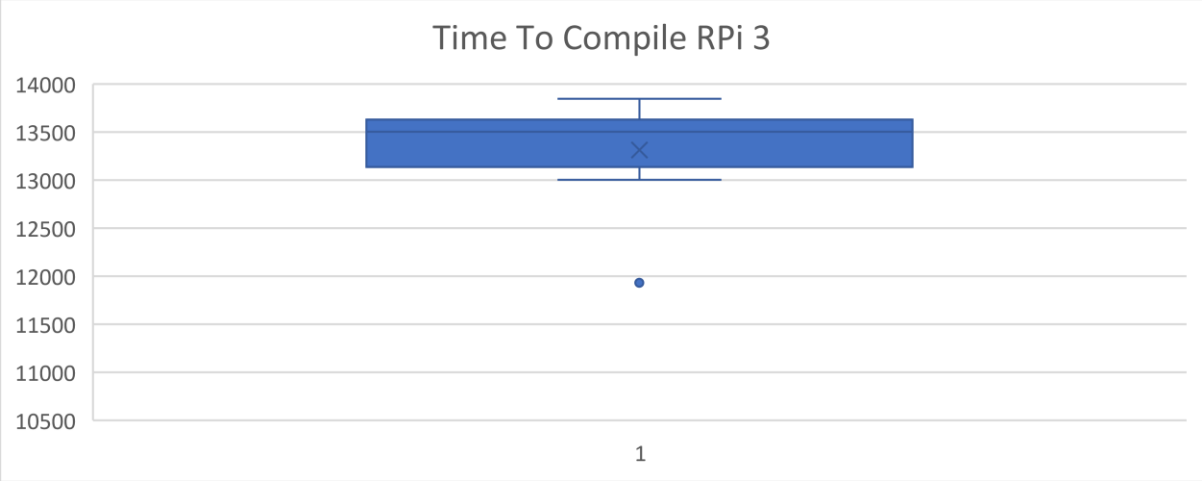

Figure 29 - Boxplot Graph of RPi 3 Time to Compile Linux Kernel

Networking – iperf

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.125 port 5001 connected with 192.168.1.74 port 47184
[ ID] Interval       Transfer     Bandwidth
[  4] 0.0000-10.0284 sec   113 MBytes  94.1 Mbits/sec
```

Figure 30 - Result of Iperf Test on RPi 4

RPi 4

Results

Timed Linux Kernel Compile

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: allmodconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 25 Minutes [12:31 UTC]
        Running Pre-Test Script @ 12:07:07
        Started Run 1 @ 12:08:32
        The test quit with a non-zero exit status.
        Running Interim Test Script @ 12:23:07
        Started Run 2 @ 12:23:22
        The test quit with a non-zero exit status.
        Running Interim Test Script @ 12:36:15
        Started Run 3 @ 12:36:30
        The test quit with a non-zero exit status.
        Running Post-Test Script @ 12:51:51
```

Figure 31 - Linux Kernel Compile Failed Attempt on RPi 4

https://openbenchmarking.org/result/2203072-FO-FIRSTRUNR33

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 25 Minutes [14:09 UTC]

        Running Pre-Test Script @ 13:44:48
        Started Run 1 @ 13:45:52
        Running Interim Test Script @ 14:30:19
        Started Run 2 @ 14:30:34
        Running Interim Test Script @ 15:15:19
        Started Run 3 @ 15:15:33

        Running Post-Test Script @ 16:00:15

    Build: defconfig:
        2665.0524389744
        2684.0575749874
        2680.6470789909

    Average: 2676.59 Seconds
    Deviation: 0.38%
```

Figure 32 - First Run of defconfig Linux Kernel Compile on RPi 4

https://openbenchmarking.org/result/2203070-FO-SECONDRUN71

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 46 Minutes [16:51 UTC]
        Running Pre-Test Script @ 16:06:42
        Started Run 1 @ 16:07:47
        Running Interim Test Script @ 16:52:13
        Started Run 2 @ 16:52:28
        Running Interim Test Script @ 17:37:11
        Started Run 3 @ 17:37:25
        Running Post-Test Script @ 18:22:10

    Build: defconfig:
        2665.0982151031
        2681.6229691505
        2684.2355310917

    Average: 2676.99 Seconds
    Deviation: 0.39%
```

Figure 33 - Second Run of defconfig Linux Kernel Compile on RPi 4

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 1 Hour, 8 Minutes [22:19 UTC]
        Running Pre-Test Script @ 21:11:27
        Started Run 1 @ 21:12:45
        Running Interim Test Script @ 21:56:32
        Started Run 2 @ 21:56:48
        Running Interim Test Script @ 22:41:00
        Started Run 3 @ 22:41:14
        Running Post-Test Script @ 23:25:26

    Build: defconfig:
        2625.7922449112
        2651.388343811
        2650.5820090771

    Average: 2642.59 Seconds
    Deviation: 0.55%
```

Figure 34 - Third Run of defconfig Linux Kernel Compile on RPi 4

Run	Time To Compile (Rounded to nearest second)
1	2665
2	2684
3	2681
4	2665
5	2682
6	2684
7	2626
8	2651
9	2651
Average	2665
Standard Deviation	17.8

Figure 35 - Table of Linux Kernel Compile Times on RPi 4

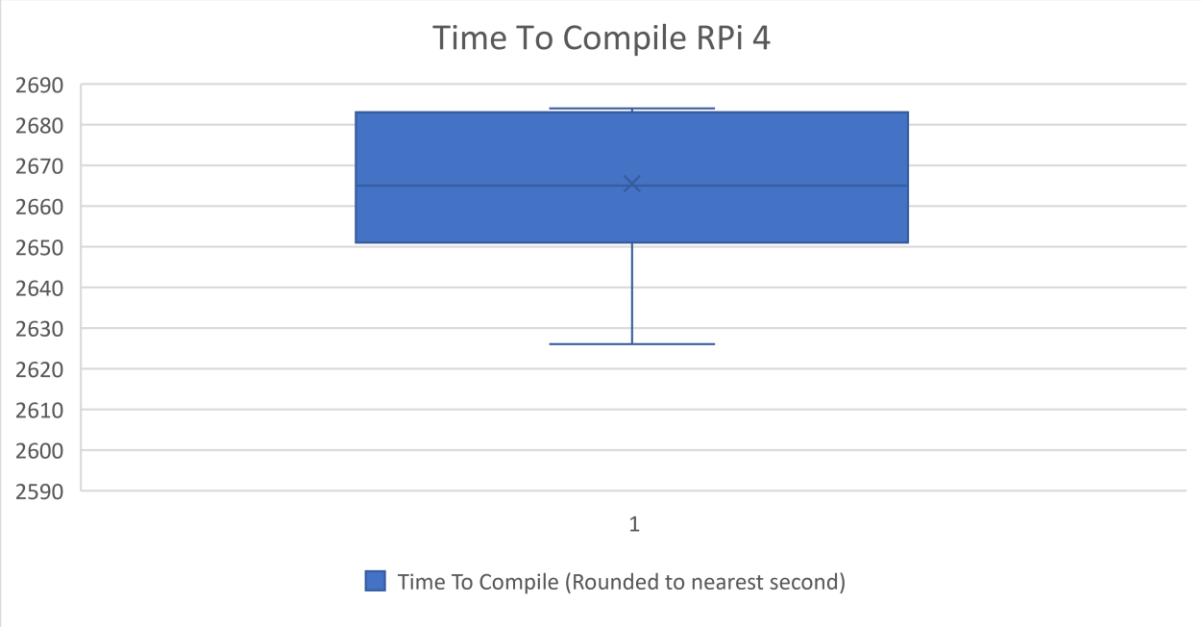

Figure 36 - Boxplot Graph of RPi 4 Time to Compile Linux Kernel

Networking – iperf

```
------------------------------------------------------------
Client connecting to 192.168.1.219, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.126 port 39836 connected with 192.168.1.219 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3] 0.0000-10.0053 sec  1.07 GBytes   921 Mbits/sec
```

Figure 37 - Result of Iperf Test on RPi 4

UCS C220-M3S

Results

Timed Linux Kernel Compile

```
  MOTHERBOARD:          Cisco s UCSC-C220-M3S
    BIOS Version:       C220M3.3.0.4e.0.1106191007
    Chipset:            Intel Xeon E5/Core
    Network:            Intel I350 Gigabit Connection

  MEMORY:               16 x 8 GB DDR3-1600MT/s M393B1K70DH0-YK0

  DISK:                 299GB MR9271-8i + 598GB MR9271-8i
    File-System:        ext4
    Mount Options:      errors=remount-ro relatime rw
    Disk Scheduler:     MQ-DEADLINE

  OPERATING SYSTEM:     Debian GNU/Linux 11
    Kernel:             5.10.0-8-amd64 (x86_64)
    Compiler:           GCC 10.2.1 20210110
    Security:           KPTI + usercopy/swapgs barriers and __user pointer sanitization + Full generic retpoline IBPB: conditional
IBRS_FW STIBP: conditional RSB filling Protection

    Would you like to save these test results (Y/n): y
    Enter a name to save these results under: linux_kernal_build_03-03-22
    Enter a unique name to describe this test run / configuration: initial run

If desired, enter a new description below to better describe this result set / system configuration under test.
Press ENTER to proceed without changes.

Current Description: 2 x Intel Xeon E5-2643 0 testing with a Cisco s UCSC-C220-M3S (C220M3.3.0.4e.0.1106191007 BIOS) and Matrox
s MGA G200e [Pilot] on Debian GNU/Linux 11 via the Phoronix Test Suite.

New Description:


Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 25 Minutes [00:17 UTC Mar 4]
        Running Pre-Test Script @ 23:52:54^[S
        Started Run 1 @ 23:53:16
        The test quit with a non-zero exit status.
        Running Interim Test Script @ 23:53:19
        Started Run 2 @ 23:53:23
        The test quit with a non-zero exit status.
        Running Interim Test Script @ 23:53:26_
```

Figure 38 - A Failed attempt to benchmark compiling the Linux Kernel on the UCS

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 1 Hour, 21 Minutes [14:48 UTC]
        Running Pre-Test Script @ 13:28:08
        Started Run 1 @ 13:28:28
        Running Interim Test Script @ 13:31:17
        Started Run 2 @ 13:31:21
        Running Interim Test Script @ 13:34:10
        Started Run 3 @ 13:34:14
        Running Post-Test Script @ 13:37:02

    Build: defconfig:
        168.68031287193
        167.06452798843
        167.37055397034

    Average: 167.71 Seconds
    Deviation: 0.51%
```

Figure 39 - First Run of defconfig Linux Kernel Compile on UCS

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 1 Hour, 7 Minutes [15:19 UTC]
        Running Pre-Test Script @ 14:13:10
        Started Run 1 @ 14:13:31
        Running Interim Test Script @ 14:16:19
        Started Run 2 @ 14:16:23
        Running Interim Test Script @ 14:19:11
        Started Run 3 @ 14:19:15
        Running Post-Test Script @ 14:22:03

    Build: defconfig:
        167.33668804169
        166.87628698349
        167.28896999359

    Average: 167.17 Seconds
    Deviation: 0.15%
```

Figure 40 - Second Run of defconfig Linux Kernel Compile on UCS

```
Timed Linux Kernel Compilation 5.16:
    pts/build-linux-kernel-1.13.0 [Build: defconfig]
    Test 1 of 1
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 57 Minutes [15:25 UTC]
        Running Pre-Test Script @ 14:28:34
        Started Run 1 @ 14:28:55
        Running Interim Test Script @ 14:31:43
        Started Run 2 @ 14:31:48
        Running Interim Test Script @ 14:34:36
        Started Run 3 @ 14:34:40
        Running Post-Test Script @ 14:37:28

    Build: defconfig:
        167.66078305244
        167.13102579117
        166.88427186012

    Average: 167.23 Seconds
    Deviation: 0.24%
```

Figure 41 - Third Run of defconfig Linux Kernel Compile on UCS

https://openbenchmarking.org/result/2203079-FO-SECONDRUN37

https://openbenchmarking.org/result/2203090-FO-THIRDRUND40

Run	Time To Compile
1	167
2	167
3	167
4	168
5	167
6	167
7	169
8	167
9	167
Average	167
Standard Deviation	0.71

Figure 42 - Table of Linux Kernel Compile Times on UCS C220-M3S

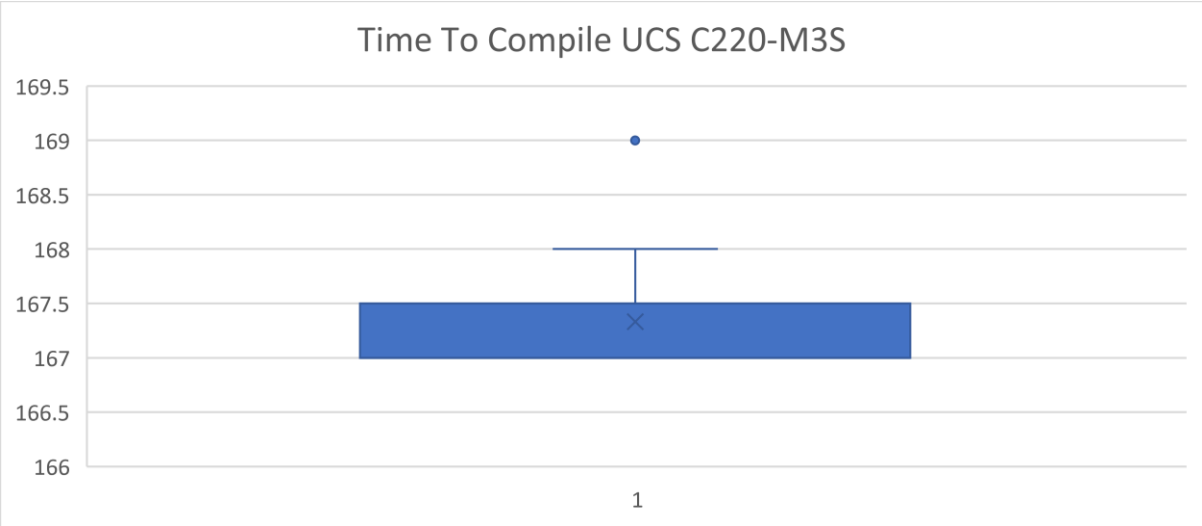

Figure 43 - Boxplot Graph of UCS C220-M3S Time to Compile Linux Kernel

Networking – iperf

```
root@rhys-debian:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.201 port 5001 connected with 192.168.1.202 port 59832
[ ID] Interval       Transfer     Bandwidth
[  4] 0.0000-10.0205 sec  1.09 GBytes   934 Mbits/sec
```

Figure 44 - Result of Iperf Test on UCS C220-M3S

Evaluation and Analysis

Networking – iperf

The table below shows the network performance of the systems analysed:

System	Transfer	Bandwidth
RPi 4	1.07GB	921Mbps
RPi 3	113MB	94.1Mbps
UCS C220-M3S	1.09GB	934Mbps

Figure 45 - Table Outlining Iperf Performance of Systems

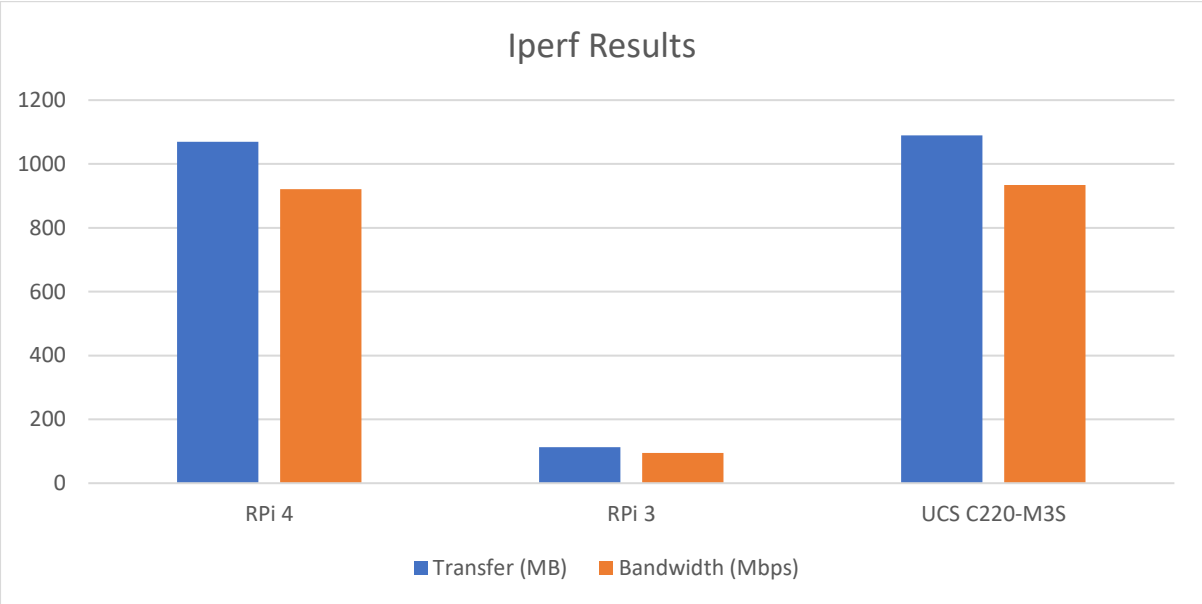

Figure 46 – Chart to Show Iperf Results

Figure 47 - Status of RPi 3 Link When iperf Testing Performed

The results shown above for the RPi 3 will be due to the 10/100 NIC included on the RPi 3 vs the speeds seen on the RPi 4 and UCS C220-M3S which both have 10/100/1000 NIC's. Further to this the speeds seen on both the RPi 4 and UCS C220-M3S are both similar reaching near the theoretical limit of a gigabit NIC. However, the RPi lacks PCIe slots that can allow for an additional 10Gigabit NIC to be added to the system which would drastically improve the performance of the UCS C220-M3S over the RPi 4.

Timed Linux Kernel Compile

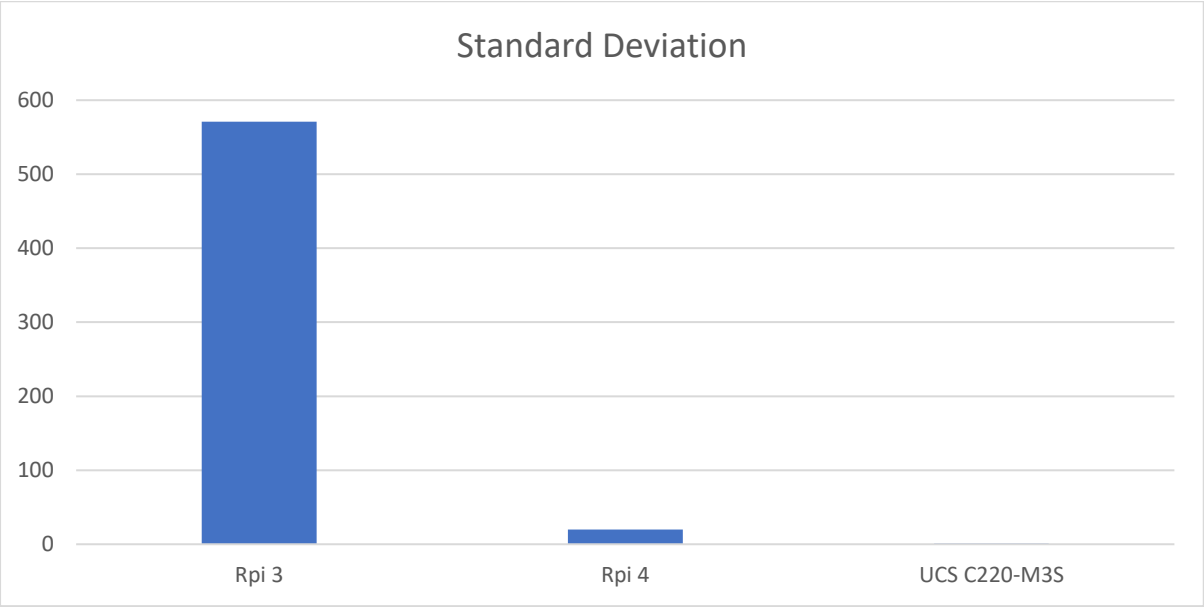

Figure 48 - Bar Chart Highlighting Standard Deviation Between Systems

The box plot charts for each system (Figure 36, Figure 29 and Figure 43) alongside the standard deviation chart above highlight the consistency of each system. The following points can be taken from the data found:

- UCS and RPi 4 perform more consistently
- RPi 3 has up to 10 mins of variation
- RPi 4 was the only system with no outlier

Although visually the RPi 3 looks more consistent in the box plot graph (Figure 29). The RPi 3 was the least consistent of the systems tested, with a standard deviation of 571s, vs the 17.8s and 0.71s standard deviations of the RPi 4 and UCS C220-M3S respectively.

The standard deviation highlights that the RPi 3 is far more inconsistent in its results, therefore resulting in inconsistencies of up to almost 10 minutes. This level of inconsistency in performance if put onto business-critical workloads could cause issues for the business.

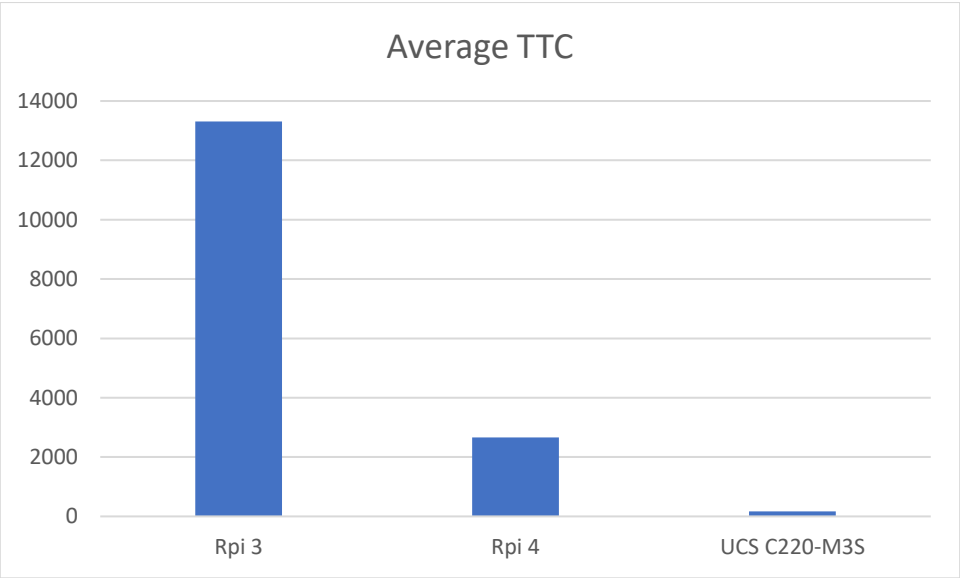

Figure 49 - Bar Chart Highlighting the Average Time to Compile Across Systems

For the average time to compile of each system it can clearly be seen from the above graph that going from left to right each system gets significantly faster than the last. The conclusions below can be drawn:

- RPi 4 takes less time to compile than RPi 3
- UCS takes less time to compile than RPi 4

On average the RPi 4 takes roughly 2 hours 56 minutes less than the RPi 3, the RPi 4 comparatively takes 20% of the time taken by the RPi 3. The difference in this raw processing power can be crucial to key business services. The faster time to compile of the RPi 4, along with the more consistent results for the time to compile, show evidence of more reliable performance from the RPi 4 over the RPi 3. Further to this the UCS C220-M3S performed the compile almost 42 minutes faster on average, the UCS C220-M3S also performed almost 3 hours 40 minutes faster than the RPi 3, the UCS C220-M3S takes 1.3% of the time taken by the RPi 3 and 6.3% of the time taken by the RPi 4. The UCS C220-M3S when performing the kernel compile does however use 50kW to compile the Linux kernel where the RPi 4 uses 13.9kW to perform the same task. Even though the UCS C220-M3S is quicker it uses more energy to provide the same result. This shows how energy efficient the RPi 4 is when compared to the UCS C220-M3S, with the cost to compile being £3.90 on the RPi 4 vs £14 on the UCS C220-M3S.

6 Services Tested

The sections and subsections following will outline different services tested on both the RPi 4 and UCS C220-M3S. Each service will be to the following subsections:

- Approach
- Results
- Evaluation and Analysis
- Conclusion

The approach section will outline how the service will be tested; the following subsection will then be the results, which will contain the results obtained from performing the testing outlined in the

approach. The results will then be analysed under, the evaluation and analysis subsection. The closing subsection will be the conclusion subsection.

DNS

Approach

The performance of the DNS/Pi-Hole service will be tested using the Gibson Research Corporation's DNS benchmark software. This software allows the testing of multiple input DNS servers, this allows for testing of both local DNS servers, ISP DNS servers and public DNS servers such as Google DNS, OpenDNS and Cloudflare DNS.

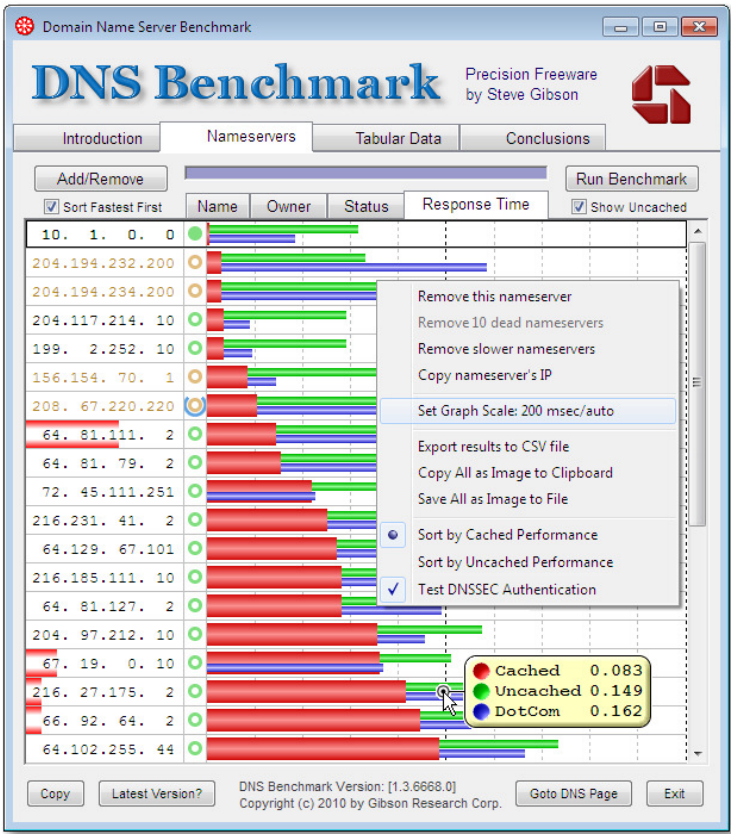

Figure 50 - DNS Benchmark GUI (Gibson Research Corporation, 2018)

The GRC DNS benchmark software provides detailed feedback on the performance of each DNS server tested with statistics such as Cached Name, Uncached Name and DotCom lookup results. It also provides such detail as providing the minimum, maximum, average and reliability results for each of the types of queries.

Results

The following sub sections detail the time to resolve DNS queries for each type of DNS server outlined below:

- RPi 4 – Pi-Hole
- UCS C200-M3S – Pi-Hole
- Virgin Media DNS
- BT DNS
- Google DNS
- OpenDNS

RPi 4 – Pi-Hole

192.168. 1. 3	Min	Avg	Max	Std.Dev	Reliab%
- Cached Name	0.002	0.007	0.015	0.003	100.0
- Uncached Name	0.025	0.067	0.281	0.069	100.0
- DotCom Lookup	0.023	0.036	0.049	0.007	100.0

pi.hole
Local Network Nameserver

Figure 51 - Table of Pi-Hole on RPi 4's Time to Resolve DNS Queries

UCS C220-M3S – Pi-Hole

7. 5. 17.236	Min	Avg	Max	Std.Dev	Reliab%
- Cached Name	0.000	0.000	0.000	0.000	100.0
- Uncached Name	0.004	0.041	0.265	0.069	100.0
- DotCom Lookup	0.004	0.009	0.016	0.004	100.0

··· no official Internet DNS name ···
DNIC-AS-00749, US

Figure 52 - Table of Pi-Hole on UCS C220-M3S Time to Resolve DNS Queries

Virgin Media DNS

194.168. 8.100	Min	Avg	Max	Std.Dev	Reliab%
- Cached Name	0.009	0.010	0.010	0.001	100.0
- Uncached Name	0.024	0.030	0.035	0.007	100.0
- DotCom Lookup	0.021	0.027	0.032	0.007	100.0

cache2.service.virginmedia.net
NTL, GB

Figure 53 - Table of Virgin Media's Time to Resolve DNS Queries

BT DNS

62. 6. 40.162	Min	Avg	Max	Std.Dev	Reliab%
- Cached Name	0.005	0.008	0.021	0.003	100.0
- Uncached Name	0.007	0.042	0.241	0.066	100.0
- DotCom Lookup	0.008	0.011	0.018	0.002	100.0

indnsc71.ukcore.bt.net
BT-UK-AS BTnet UK Regional network, GB

62. 6. 40.178	Min	Avg	Max	Std.Dev	Reliab%
- Cached Name	0.006	0.021	0.230	0.045	100.0
- Uncached Name	0.007	0.041	0.236	0.063	100.0
- DotCom Lookup	0.008	0.010	0.016	0.002	100.0

indnsc76.bt.net
BT-UK-AS BTnet UK Regional network, GB

Figure 54 - Tables of BT Time to Resolve DNS Queries

Google DNS

8. 8. 8. 8	Min	Avg	Max	Std.Dev	Reliab%
- Cached Name	0.013	0.017	0.024	0.003	100.0
- Uncached Name	0.015	0.043	0.242	0.045	100.0
- DotCom Lookup	0.016	0.023	0.039	0.005	100.0

dns.google
GOOGLE, US

Figure 55 - Table of Google Time to Resolve DNS Queries

OpenDNS

208. 67.222.222	Min	Avg	Max	Std.Dev	Reliab%
- Cached Name	0.016	0.019	0.027	0.002	100.0
- Uncached Name	0.017	0.045	0.258	0.048	100.0
- DotCom Lookup	0.017	0.028	0.038	0.006	100.0

dns.opendns.com
OPENDNS, US

Figure 56 - Table of OpenDNS Time to Resolve DNS Queries

Evaluation and Analysis

Pi-Hole is a DNS forwarder, so the results were as expected. The cached DNS entries performing better than ISP DNS servers and non-cached entries performing marginally slower than the ISP DNS servers. Given the differences between the RPi 4 and the UCS C220-M3S, there were only marginal performance differences between the RPi 4 and UCS C220-M3S. These differences are only minor and could be due to the difference in networking configuration between the home set up and Cisco's DMZ. Another consideration is the enterprise grade hardware used in Cisco's DMZ when compared over the enthusiast grade hardware used in home.

Conclusion

Utilising Pi-Hole on the RPi 4 as a DNS forwarder for SMEs, would be beneficial as this allows for the explicit blocking of NSFW sites and known advert domains. The performance of Pi-Hole compared to both BT and Virgin Medias DNS, was slower to resolve the non-cached entries. But faster than both of the ISP's DNS servers for the cached entries. Considering there is slight performance gain/loss of Pi-Hole, for the added benefit of website filtering and telemetry blocking, that can be managed by the company themselves. This is beneficial for business as the ad blocking element to Pi-Hole allows for dedicated content filtering and the RPi 4 which easily handles over 56,000 queries with the CPU sitting below 30% utilisation.

DHCP

Approach

```
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#

# option definitions common to all supported networks...
#option domain-name "example.org";
option domain-name-servers 8.8.8.8, 8.8.4.4;

default-lease-time 600;
max-lease-time 7200;

# The ddns-updates-style parameter controls whether or not the server will
# attempt to do a DNS update when a lease is confirmed. We default to the
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
ddns-update-style none;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
#log-facility local7;

# No service will be given on this subnet, but declaring it helps the
# DHCP server to understand the network topology.

#subnet 10.152.187.0 netmask 255.255.255.0 {
#}
subnet 192.168.1.0 netmask 255.255.255.0 {
}

subnet 192.168.2.0 netmask 255.255.255.0 {
    range 192.168.2.64 192.168.2.126;
    option routers 192.168.2.1; C:\Users\Rhys\OneDrive - Cardiff University\4
- Fourth Year\FYP\Final Report\TTD Results.txt
    default-lease-time 7200;
    max-lease-time 28400;
    option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

Figure 57 - Section of the /etc/dhcp/dhcpd.conf File used by isc-dhcp-server

Although it was initially thought that the time to receive a DHCP lease would be used to test the performance of the DHCP server, the ipchama/dhammer (ipchama, 2021) DHCP stress testing tool

was used and a HTOP output from the VM/RPi taken. The commands below were used on the RPi 4 to initiate the stress test:

$ sudo ./dhammer dhcpv4 --interface eth0 --mac-count 10000 --rps 1000 --maxlife 0 --relay-target-server-ip 192.168.1.125 --relay-source-ip 192.168.1.1

The command below was used for the traditional server set up:

$ sudo ./dhammer dhcpv4 --interface ens0 --mac-count 10000 --rps 1000 --maxlife 0 --relay-target-server-ip 7.5.17.236 --relay-source-ip 7.5.17.254

By utilising the HTOP output the strain the DHCP puts on the hardware for each of the systems can be seen.

Results

RPi 4

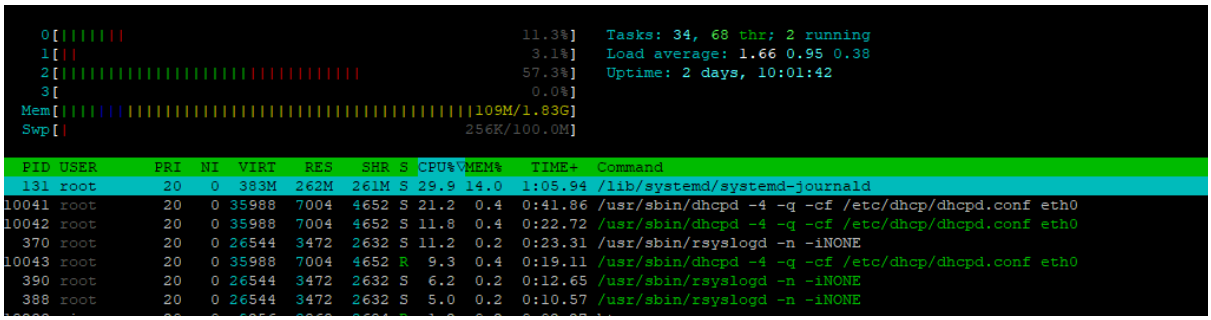

Figure 58 - HTOP Output Running Dhammer DHCP Stress Test on RPi 4

UCS C220-M3S

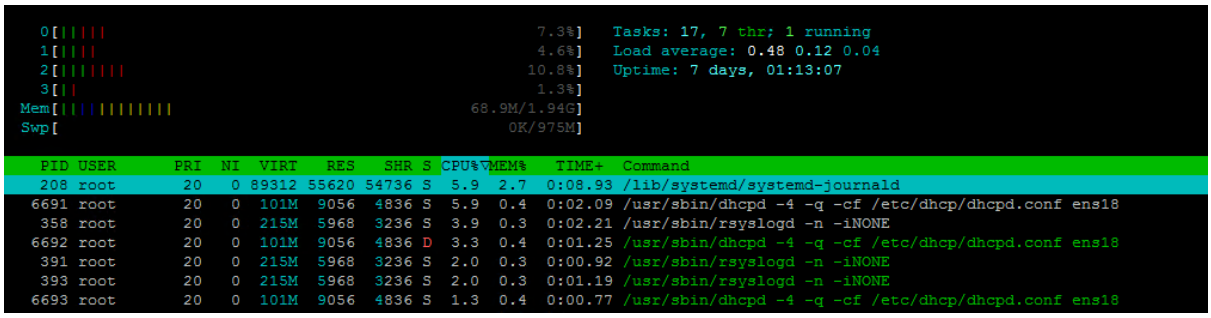

Figure 59 - HTOP Output Running Dhammer DHCP Stress Test on UCS C220-M3S

Evaluation and Analysis

Figure 58 and Figure 59, show that the dhcp service utilises 21.2% CPU on a single thread, on the RPi 4 and 5.9% on the UCS C220-M3S. This is whilst allocating 1000 DHCP leases to virtual clients generated in the dhammer script. This is far more DHCP requests than would be received in an SME, with the European Commission's definition of an SME having less than 250 employees (European Comission, 2022) the maximum DHCP requests assumed to be seen by the network would be 500, assuming 1 for their work laptop/desktop and 1 for their personal/work mobile. With the large quantity of requests being substantially more than a small enterprise would receive in a given second. The low CPU draw on both systems could be run alongside Pi-Hole. Whilst Pi-Hole is running on a container, Pi-Hole & DHCP can be run concurrently on the RPi 4 and use less than 50% CPU to run both services.

Conclusion

Although this service can easily be run on the RPi 4 this is often already handled by the router supplied by the ISP. For this I would not recommend the RPi 4 as this service is already facilitated. The RPi 4 would however be suitable if the business would like to have a glass-isc-dhcp web management for their DHCP, again however this is also often provided with the ISP router.

Webserver

Approach

To test the webserver running on the systems, the technologies listed below will be used:

- Python
- Flask
- Nginx
- Wrk

The webserver will be running using docker compose. The docker compose file will create both, the Flask container hosting the static website details, and the Nginx container that acts as the web proxy for the Flask container. Mapping port 80 on the host to port 8080 on the flask container. This website will be tested over HTTP, provided there is time at the end of the project HTTPS may also be tested using a self-signed certificate or free certificate provided by LetsEncrypt.

Below is a website used for testing the webserver:

Figure 60 - Screenshot of Webserver Main Page

Using wrk to test the webserver the following commands were used:

```
$ wrk -t12 -c50 -d30s --latency http://<IP-OF-HOST>
$ wrk -t12 -c100 -d30s --latency http://<IP-OF-HOST>
$ wrk -t12 -c200 -d30s --latency http://<IP-OF-HOST>
$ wrk -t12 -c400 -d30s --latency http://<IP-OF-HOST>
$ wrk -t12 -c20 -d86400s --latency http://<IP-OF-HOST>
```

The commands above generate 50, 100, 200 and 400 concurrent connections using the -c command. There will also be a test of 20 connections over the duration of a day (24 hour period).

Results

RPi 4

HTOP

```
  0[||||||||||||||||||||||||||||||||||||||||||||||||||89.7%]   Tasks: 59, 106 thr; 4 running
  1[||||||||||||||||||||||||||||||||||||||||||||||||||90.9%]   Load average: 3.65 1.13 0.50
  2[||||||||||||||||||||||||||||||||||||||||||||||||||89.7%]   Uptime: 00:38:12
  3[||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]
Mem[||||||||||||||||||||||||||||||||||||||||||||||210M/1.83G]
Swp[                                                0K/100.0M]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
 2411 root       20   0 72896 27020  9920 S 66.7  1.4  0:28.01 /usr/bin/python3 /usr/local/bin/docker-compose up
  661 root       20   0  967M 59196 35648 S 40.8  3.1  1:48.85 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containe
 2772 root       20   0 63960 16460  3992 R 36.9  0.9  0:14.77 uwsgi app.ini
 2768 root       20   0 63960 16460  3992 R 35.6  0.9  0:12.87 uwsgi app.ini
 2769 root       20   0 63960 16428  3960 R 31.1  0.9  0:12.15 uwsgi app.ini
 2771 root       20   0 63960 16388  3924 R 29.8  0.9  0:11.26 uwsgi app.ini
 1077 pi         20   0 14676  5184  4052 S 23.9  0.3  0:09.59 sshd: pi@pts/0
 2742 root       20   0 72896 27020  9920 R 22.0  1.4  0:08.33 /usr/bin/python3 /usr/local/bin/docker-compose up
 2743 root       20   0 72896 27020  9920 S 22.0  1.4  0:08.27 /usr/bin/python3 /usr/local/bin/docker-compose up
 2751 systemd-n  20   0  6920  2884  1568 S 19.4  0.2  0:06.13 nginx: worker process
 2750 systemd-n  20   0  7208  3128  1572 R 17.5  0.2  0:07.18 nginx: worker process
 2752 systemd-n  20   0  6940  2812  1508 S 17.5  0.1  0:05.97 nginx: worker process
```

Figure 61 - HTOP Output from Webserver Handling 400 Requests on RPi 4

Wrk Results

```
pi@rhys-uni-pi-3:~ $ wrk -t12 -c50 -d30s --latency http://192.168.1.125
Running 30s test @ http://192.168.1.125
  12 threads and 50 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    44.92ms    5.15ms 102.82ms   80.66%
    Req/Sec    89.09      9.57   121.00    73.67%
  Latency Distribution
     50%   43.70ms
     75%   46.65ms
     90%   51.22ms
     99%   63.51ms
  32037 requests in 30.03s, 6.26MB read
Requests/sec:   1066.96
Transfer/sec:    213.60KB
```

Figure 62 - Example of CLI Output from Wrk Testing RPi 4

The following results are provided as seen from the Wrk output above.

12 Threads and 50 Connections

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	44.92ms	5.15ms	102.82ms	80.66
Req/Sec	89.09	9.57	121	73.67

Figure 63 - Table of Results For 50 Connections on RPi 4

Latency Distribution

50% 43.70ms

75% 46.65ms

90% 51.22ms

99% 63.51ms

32037 requests in 30.03s, 6.26MB read

Requests/sec: 1066.96

Transfer/sec: 213.60KB

12 Threads and 100 Connections

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	89.99ms	6.88ms	158.59ms	81.53
Req/Sec	88.94	10.12	131	73.32

Figure 64 - Table of Results For 100 Connections on RPi 4

Latency Distribution

50% 88.89ms

75% 92.53ms

90% 97.69ms

99% 112.33ms

31959 requests in 30.03s, 6.25MB read

Requests/sec: 1064.35

Transfer/sec: 213.08KB

12 Threads and 200 Connections

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	264.15	333.49ms	2.00s	85.17
Req/Sec	88.32	23.22	180	73.71

Figure 65 - Table of Results For 200 Connections on RPi 4

Latency Distribution

50% 101.57ms

75% 212.75ms

90% 818.47ms

99% 1.38s

31657 requests in 30.04s, 6.19MB read

Socket errors: connect 0, read 0, write 0, timeout 152

Non-2xx or 3xx responses: 5

Requests/sec: 1053.95

Transfer/sec: 211.01KB

12 Threads and 400 Connections

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	263.25ms	362.33ms	2.00s	86.15
Req/Sec	88.59	30.79	303	67.29

Figure 66 - Table of Results For 400 Connections on RPi 4

Latency Distribution

50% 100.60ms

75% 114.26ms

90% 825.65ms

99% 1.57s

31668 requests in 30.07s, 6.19MB read

Socket errors: connect 0, read 0, write 0, timeout 681

Non-2xx or 3xx responses: 12

Requests/sec: 1053.30

Transfer/sec: 210.91KB

12 Threads and 20 Connections for 24 Hours

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	12.34ms	5.66ms	1.04s	81.99%
Req/Sec	81.35	16.14	303.00	69.16%

Figure 67 - Table of Results for 20 Connections on UCS C220-M3S Over a 24-Hour Period

Running 1440m test @ http://192.168.1.125

Latency Distribution

50% 10.78ms

75% 14.17ms

90% 19.35ms

99% 33.09ms

84199747 requests in 1440.00m, 16.08GB read

Requests/sec: 974.53

Transfer/sec: 195.09KB

UCS C220-M3S

HTOP

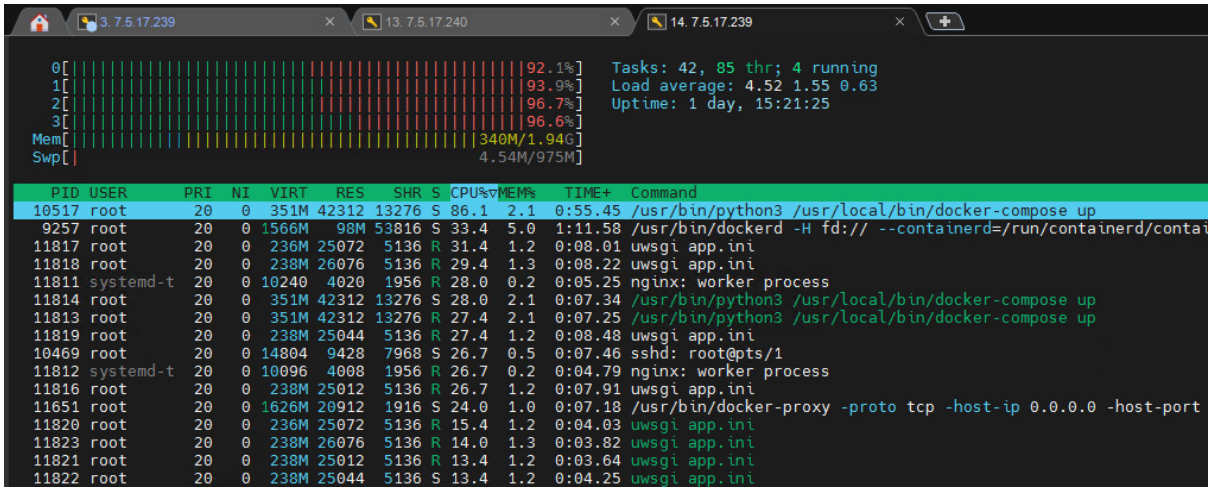

Figure 68 - HTOP Output from Webserver Handling 50 Requests on UCS C220-M3S

Wrk Results

```
root@rhys-wrk:~# wrk -t12 -c50 -d30s --latency http://7.5.17.239
Running 30s test @ http://7.5.17.239
  12 threads and 50 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    18.57ms    5.72ms  87.03ms   79.22%
    Req/Sec   217.10     29.31   484.00     78.06%
  Latency Distribution
     50%   18.36ms
     75%   20.68ms
     90%   24.17ms
     99%   38.63ms
  77866 requests in 30.03s, 15.22MB read
Requests/sec:   2593.03
Transfer/sec:    519.10KB
```

Figure 69 - Example of CLI Output from Wrk Testing UCS C220-M3S

All of the following results are provided as seen from the Wrk output.

12 threads and 50 connections

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	18.57ms	5.72ms	87.03ms	79.22%
Req/Sec	217.10	29.31	484	78.06%

Figure 70 - Table of Results for 50 Connections on UCS C220-M3S

Latency Distribution

50% 18.36ms

75% 20.68ms

90% 24.17ms

99% 38.63ms

77866 requests in 30.03s, 15.22MB read

Requests/sec: 2593.03

Transfer/sec: 519.10KB

12 threads and 100 connections

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	36.63ms	7.90ms	138.48ms	78.38%
Req/Sec	219.2	34.97	474.00	78.42%

Figure 71 - Table of Results for 100 Connections on UCS C220-M3S

Latency Distribution

50% 36.89ms

75% 40.34ms

90% 44.72ms

99% 59.26ms

78625 requests in 30.03s, 15.37MB read

Requests/sec: 2618.10

Transfer/sec: 524.13KB

12 threads and 200 connections

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	188.91ms	291.71ms	1.92s	84.72%
Req/Sec	216.54	53.49	434	69.36%

Figure 72 - Table of Results for 200 Connections on UCS C220-M3S

Latency Distribution

50% 44.21ms

75% 163.09ms

90% 698.50ms

99% 1.10s

77641 requests in 30.02s, 15.18MB read

Socket errors: connect 0, read 0, write 0, timeout 44

Requests/sec: 2586.09

Transfer/sec: 517.72KB

12 threads and 400 connections

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	252.68ms	365.45ms	1.98s	83.17%
Req/Sec	212.67	68.56	696.00	68.60%

Figure 73 - Table of Results for 400 Connections on UCS C220-M3S

Latency Distribution

50% 45.22ms

75% 330.01ms

90% 897.28ms

99% 1.43s

76347 requests in 30.10s, 14.93MB read

Socket errors: connect 0, read 0, write 0, timeout 697

Non-2xx or 3xx responses: 8

Requests/sec: 2536.53

Transfer/sec: 507.83KB

12 Threads and 20 Connections for 24 Hours

Thread Stats	Average	Stdev	Max	+/- Stdev
Latency	5.71ms	3.30ms	1.04s	86.04%

Req/Sec	178.53	40.95	600.00	70.63%

Figure 74 - Table of Results for 20 Connections on UCS C220-M3S Over a 24-Hour Period

Running 1440m test @ http://7.5.17.239

Latency Distribution

50% 5.20ms

75% 6.67ms

90% 8.62ms

99% 17.18ms

184394667 requests in 1440.00m, 35.20GB read

Requests/sec: 2134.20

Transfer/sec: 427.25KB

Evaluation and Analysis

Country Connect Web Requests

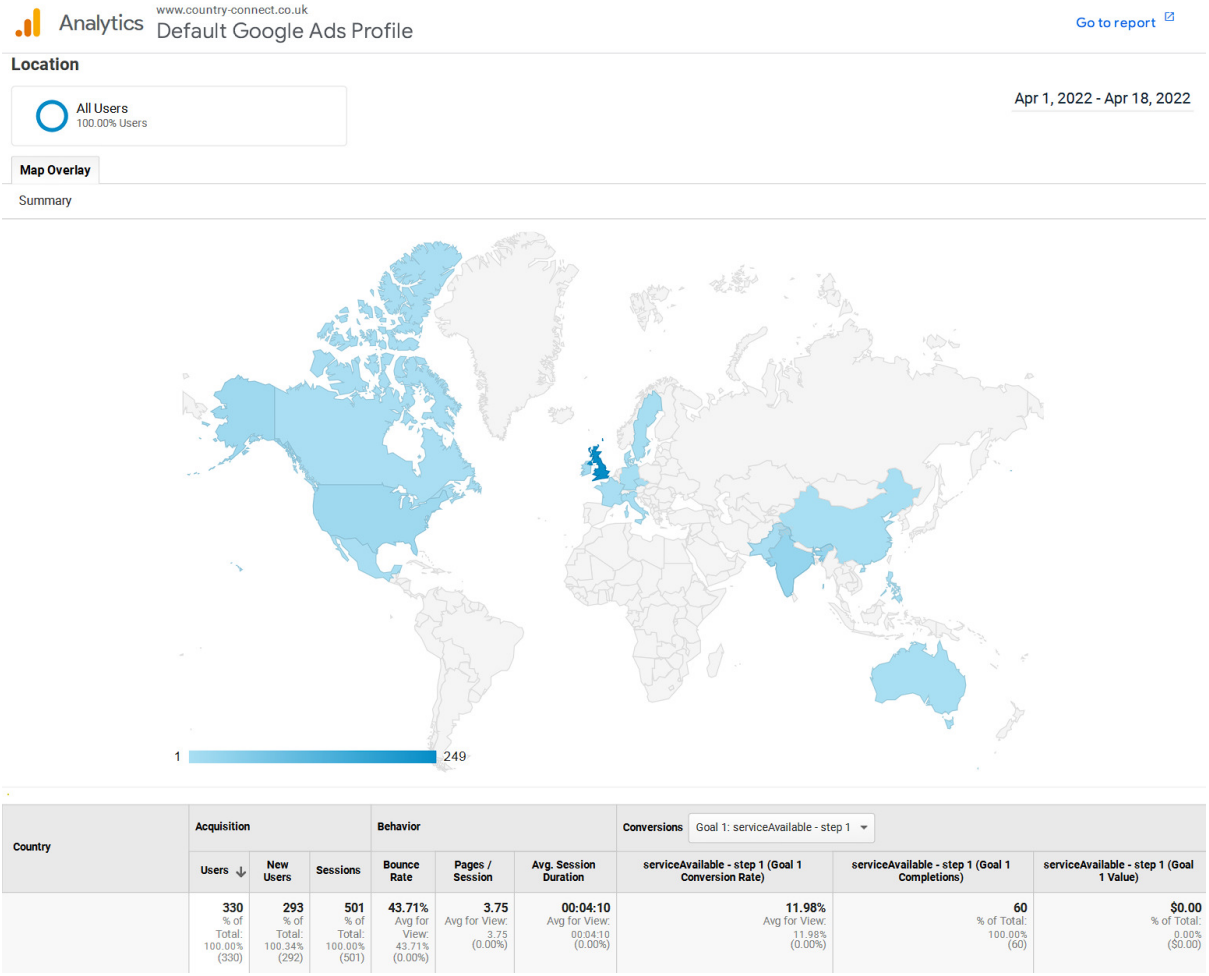

Figure 75 - Web Requests for Country Connect Website

The figure above shows the webserver traffic generated for Country Connect Ltd who is an ISP and MSP based out of south Wales.

RPi 4

The figures below show the results gathered in the webserver testing of the RPi 4:

- Figure 63
- Figure 64
- Figure 65
- Figure 66

The Figures above show detailed results for how many requests were handled per second by the webserver and also metrics for the latency. These results show that the RPi 4 can handle up to 100 concurrent requests with no errors, even jumping to 200 concurrent requests the RPi 4 drops 152 requests over a 30s period. Although this is not ideal that requests time out for a business webserver this will be suitable for most SMEs as the likelihood of them receiving over 100 concurrent requests will likely be small, however once they notice requests hitting over 100 concurrent this could then be migrated to the cloud or even a traditional server.

UCS C220-M3S

The figures below show the results gathered in the webserver testing of the UCS C220-M3S:

- Figure 70
- Figure 71
- Figure 72
- Figure 73

The figures above highlight the performance of the webserver that a VM with the below spec on the UCS C220-M3S:

- 4 vCPU
- 4GB RAM
- 32GB vHDD

From the figures mentioned above it can be seen that for up to 100 concurrent connections 99% of requests were responded to in under 60ms. It can also be seen that over a 30 second period the webserver can respond to/serve ≈77500 requests, this is far more than the average weekly requests seen in the data supplied by Country Connect Ltd.

UCS C220-M3S vs RPi 4

The figures below show that the UCS C220-M3S handled roughly double the requests per second of the RPi 4, 180-220 vs 80-90 requests/sec. The performance difference seen was further confirmed again with the latency of the response latency from the UCS C220-M3S being half of that by the RPi 4. The real-world application of this it means that customers visiting the website will have it load quicker if the website is hosted on the UCS C220-M3S than the RPi 4. However, this would only be noticeable to the end user after 100 or more concurrent requests were made to the website. The RPi 4 could also however handle almost 200 concurrent requests before returning any Non 2xx or 3xx responses where the VM running on the UCS didn't experience these issues until reaching 400 concurrent requests. Across both the systems the standard deviation of the webserver is similar, this shows that the RPi 4 and UCS C220-M3S both perform as consistently as the other system.

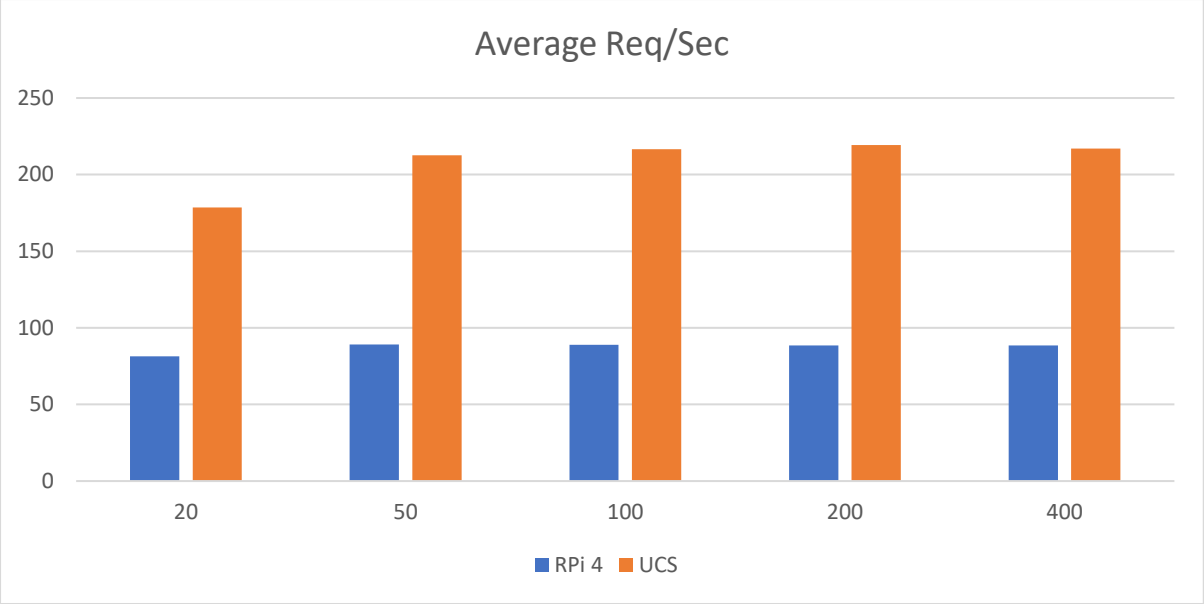

Figure 76 - Average Requests Per Second for Webserver Running on Each System

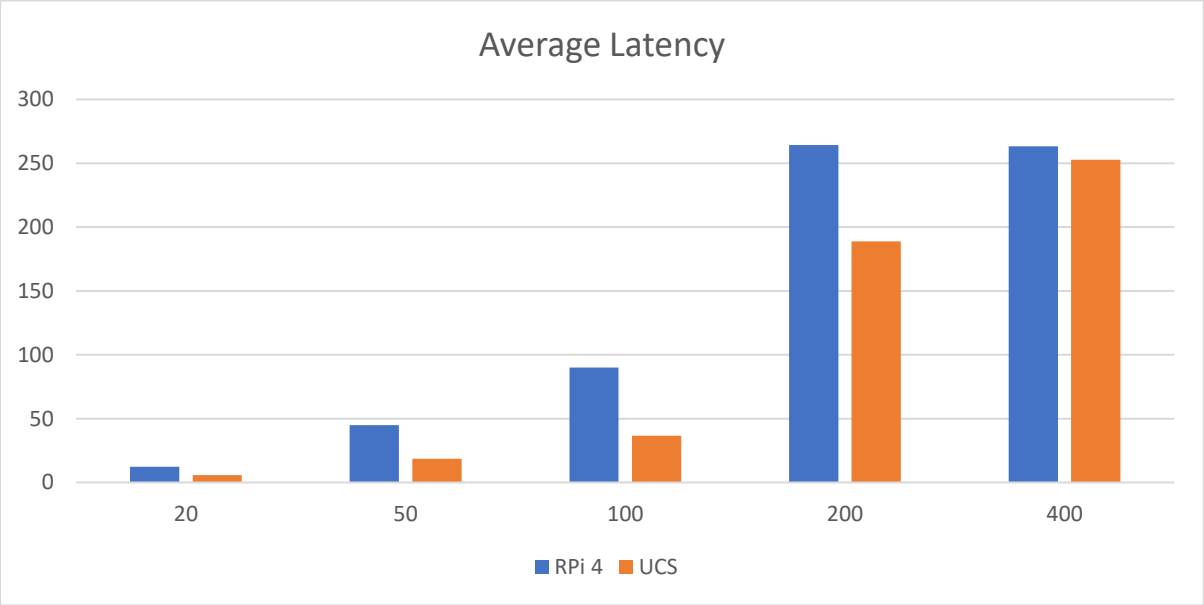

Figure 77 - Average Latency for Webserver Running on Each System

Conclusion

Using the data provided from my current employer Country Connect Ltd, it is clear that a RPi 4 could easily handle the levels of traffic going to a business that serves over 3000+ UK internet customers and provides IT MSP services to several SMEs. Although the hardware of the RPi 4 cannot be upgraded, like adding a more powerful CPU or adding more RAM/storage etc. Utilising technologies such as Docker allows for the easy migration of this service to either the cloud or even to a more traditional rackmount server solution.

Although the RPi 4 does perform worse than the traditional server, when the performance of the RPi 4 is taken into consideration, alongside the power drawn by each system the RPi 4 performs extremely well. In real world application a cluster of RPi 4's could be run for a fraction of the server's cost in both hardware and electricity.

The RPi 4 would be a suitable webserver for any SMEs that get relatively low traffic to their website and that get less than 50-100 concurrent connections at any given time. This for example would be ideal for businesses such as a local Baker, Estate Agent, Butcher etc that will likely never expand out of the local county/town/village but would equally benefit from the customers generated through a website.

AD

Approach

To test the feasibility of running the RPi as an Active Directory Domain Controller (AD DC), a RPi 4 will be set up using Samba (SMB) which is an open-source technology that is compatible with Windows AD. Although SMB is compatible it does not come with the full feature set of a Windows AD DC, this could be limiting as the company grows. For a lot of SMEs, SMB is suitable as they wouldn't use the full feature set of a dedicated Windows AD DC.

The testing will involve configuring the AD DC server to then run 5-20 Windows 10 VMs and link this to the DC and see the HTOP output whilst these VMs are joining the domain.

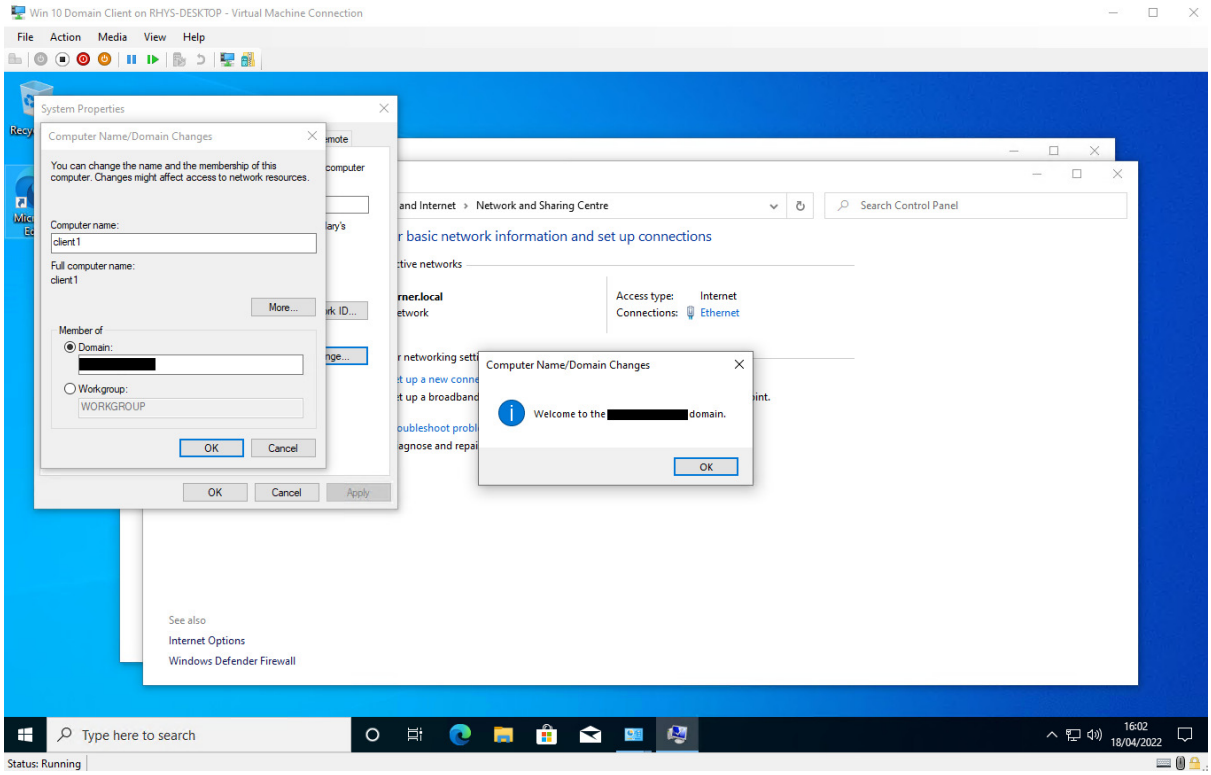

Figure 78 - Successful Connection of Client to AD DC

The statistics used to test the feasibility of this on each system will be as below:

- Winbind
- HTOP Output with 5 Clients connected
- Time to add client to domain

The statistics listed above will be gathered as follows, for Winbind the log level of the server will be increased to include individual requests. This will allow for timestamps that will inform the project of how long authentication takes, when a client queries the AD DC server. Also the time to join a client

to the domain could be recorded on the end user side to see the real time taken, as seen by what the IT team would see when joining desktops to the domain for new starters etc.

Results

Following the outline of how the results will be gathered, this section will outline the HTOP output of the AD DC and the time to process Winbind requests.

RPi 4

HTOP

```
  0[|||                                                3.2%]   Tasks: 103, 57 thr; 1 running
  1[                                                     0.0%]   Load average: 0.15 0.11 0.04
  2[                                                     0.0%]   Uptime: 01:18:33
  3[                                                     0.0%]
Mem[||||||||||||||                                 362M/3.71G]
Swp[                                                 0K/100.0M]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
 1709 pi         20   0  9620  5480  2900 R  3.9  0.1  0:03.79 htop
 1165 root       20   0  140M 32736  8552 S  0.6  0.8  0:06.97 samba: task[drepl] pre-fork master
    1 root       20   0  162M 10056  7448 S  0.0  0.3  0:02.24 /sbin/init
  133 root       20   0   99M 63208 62160 S  0.0  1.6  0:05.09 /lib/systemd/systemd-journald
  164 root       20   0 21252  5804  3660 S  0.0  0.1  0:01.35 /lib/systemd/systemd-udevd
  373 systemd-t  20   0 88092  6100  5364 S  0.0  0.2  0:00.35 /lib/systemd/systemd-timesyncd
  397 systemd-t  20   0 88092  6100  5364 S  0.0  0.2  0:00.01 /lib/systemd/systemd-timesyncd
  400 avahi      20   0  7152  3260  2760 S  0.0  0.1  0:01.96 avahi-daemon: running [dc1.local]
  401 root       20   0  6696  2396  2196 S  0.0  0.1  0:00.01 /usr/sbin/cron -f
```

Figure 79 - HTOP Output from RPi 4 Whilst 1 User Is Logging onto Client Machine

There were also millisecond spikes to 100% on a single thread noticed however these were not prolonged and were extremely brief spikes when adding a client to the domain.

Results Found

Log.winbindd-idmap

[2022/04/18 16:21:58.558371, 4] ../../source3/winbindd/winbindd_dual.c:1658(child_handler)

child daemon request 56

[2022/04/18 16:21:58.581028, 4] ../../source3/winbindd/winbindd_dual.c:1666(child_handler)

Finished processing child request 56

[2022/04/19 13:37:49.501793, 4] ../../source3/winbindd/winbindd_dual.c:1658(child_handler)

child daemon request 56

[2022/04/19 13:37:49.515157, 4] ../../source3/winbindd/winbindd_dual.c:1666(child_handler)

Finished processing child request 56

Log.samba

[2022/04/18 16:21:57.284685, 4] ../../source4/auth/sam.c:203(authsam_account_ok)

authsam_account_ok: Checking SMB password for user client1$@lab.local

[2022/04/19 13:37:47.927762, 4] ../../source4/auth/sam.c:203(authsam_account_ok)

authsam_account_ok: Checking SMB password for user client2$@lab.local

UCS C220-M3S

HTOP Output

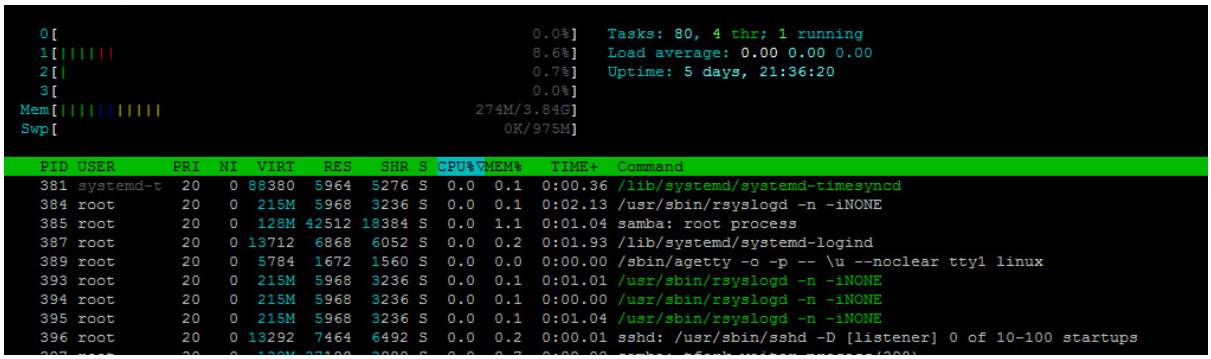

Figure 80 - HTOP Output from UCS AD DC Whilst 1 User Is Logging onto Client Machine

Log.winbindd-idmap

[2022/05/10 20:08:32.987307, 4] ../../source3/winbindd/winbindd_dual.c:1658(child_handler)

child daemon request 56

[2022/05/10 20:08:32.999477, 4] ../../source3/winbindd/winbindd_dual.c:1666(child_handler)

Finished processing child request 56

[2022/05/10 21:02:34.119256, 4] ../../source3/winbindd/winbindd_dual.c:1658(child_handler)

child daemon request 56

[2022/05/10 21:02:34.130786, 4] ../../source3/winbindd/winbindd_dual.c:1666(child_handler)

Finished processing child request 56

Log.samba

[2022/05/10 20:08:32.715109, 4] ../../source4/auth/sam.c:203(authsam_account_ok)

authsam_account_ok: Checking SMB password for user client-1$@lab.local

[2022/05/10 21:02:34.066749, 4] ../../source4/auth/sam.c:203(authsam_account_ok)

authsam_account_ok: Checking SMB password for user client-2$@lab.local

Evaluation and Analysis

In order to make the log files output more measurable the calculations below will be made to ensure proper comparisons across both the systems:

$$Time\ to\ Complete\ =\ Time\ of\ Winbind\ Completion\ -\ Time\ of\ SMB\ Password\ Check$$

$$Time\ Between\ SMB\ Password\ Check\ and\ Winbind\ Starting = Time\ of\ Winbind\ Starting\ -\ Time\ of\ SMB\ Password\ Check$$

$$Time\ to\ Complete\ Winbind\ Request = Time\ of\ Winbind\ Starting\ -\ Time\ of\ Winbind\ Completion$$

TTC = Time between initial password check and Winbind request completing

TBPAW = Time between initial password check and Winbind request starting

TTCWR = Time to complete Winbind request

	TTC	TBPAW	TTCWR
RPi 4	1.441869	1.4238585	0.0180105
UCS C220-M3S	0.1742025	0.1623525	0.01185

Figure 81 - Table of AD Results for Both Systems

It can be seen from the table above that the UCS C220-M3S handles SMB and Winbind requests quicker than the RPi 4. The UCS completed the entire process in almost 1/10th of the time taken by the RPi 4. However, the RPi 4 completed the Winbind request 6.16ms slower than the UCS the largest difference in the TTC was seen in the time between the SMB password check and initiating the Winbind request.

Conclusion

The RPi 4 could be a viable option as an AD server for smaller SMEs, however this service would require more in-depth testing and analysis over a long period of time to see how viable this solution is. The RPi 4 performing AD queries at 10 times the speed of the UCS C220-M3S matches the relative speed difference seen with the timed Linux kernel compile. Further testing of this service on the RPi 4 is required, before a definitive conclusion can be made and a recommendation provided.

NAS

Approach

To test run the NAS, the commands listed below will be used on the Windows machine to generate test files, these files will be 1GB, 10GB, 100GB and 1TB:

```
$ fsutil file createnew E:\1tb.test 1099511627776
$ fsutil file createnew E:\100gb.test 107374182400
$ fsutil file createnew E:\10gb.test 10737418240
$ fsutil file createnew E:\1gb.test 1073741824
```

The time to transfer each of these files will then be recorded and from the results the average transfer speed will be calculated. Both the time to transfer and the average transfer speed can then be used to evaluate the performance of the RPi 4 vs the UCS C220-M3S.

To get an accurate time to download the code excerpt below was used:

```
from datetime import datetime
import ftplib



FTP_HOST = "HOSTNAME OR IP ADDRESS"
FTP_USER = "USERNAME"
FTP_PASS = "PASSWORD"

ftp = ftplib.FTP(FTP_HOST, FTP_USER, FTP_PASS)

LocalFile1GB = 'E:\\1gb.test'

StartTime1GB = datetime.now()

with open(LocalFile1GB, "rb") as file:
    ftp.storbinary(f"STOR /uni/1gb.test", file)

TTC1GB = str(datetime.now() - StartTime1GB)

print("Time to copy 1GB File From FTP: \n" + TTC1GB)
```

Figure 82 - Code Snippet from Script to Upload Files to NAS Using FTP and Python

Everything following the StartTime1GB variable was then repeated for the 10GB, 100GB and 1TB files, with the time to transfer being output prior to starting the next transfer. The following lines differentiated the upload and download process:

```
LocalFile1GB = 'E:\\1gb.test'
with open(LocalFile1GB, "wb") as file:
    ftp.retrbinary(f"RETR /uni/1gb.test", file.write)
```

Figure 83 – Code Snippet for Downloading from NAS

```
LocalFile1GB = 'E:\\1gb.test'
with open(LocalFile1GB, "rb") as file:
    ftp.storbinary(f"STOR /uni/1gb.test", file)
```

Figure 84 - Code Snippet for Uploading to NAS

Using the ftplib module in Python the upload and download can be differentiated by the storbinary and retrbinary functions. The storbinary function was used to upload the file from the Windows 10 client to the NAS using File Transfer Protocol (FTP). The retrbinary function was then used to download the file from the NAS onto the Windows 10 client using FTP.

Results

Following the outline of how the results will be gathered this section will outline both the raw performance of the drives attached to the NAS and the performance of uploading/downloading to/from the RPi 4 and UCS C220-M3S NAS.

Local Drive Results

Windows 10 Client HDD for RPi 4

```
C:\Windows\system32>winsat disk -drive e
Windows System Assessment Tool
> Running: Feature Enumeration ''
> Run Time 00:00:00.00
> Running: Storage Assessment '-drive e -ran -read'
> Run Time 00:00:08.47
> Running: Storage Assessment '-drive e -seq -read'
> Run Time 00:00:04.14
> Running: Storage Assessment '-drive e -seq -write'
> Run Time 00:00:04.19
> Running: Storage Assessment '-drive e -flush -seq'
> Run Time 00:00:04.41
> Running: Storage Assessment '-drive e -flush -ran'
> Run Time 00:00:11.33
> Dshow Video Encode Time                      0.00000 s
> Dshow Video Decode Time                      0.00000 s
> Media Foundation Decode Time                 0.00000 s
> Disk  Random 16.0 Read                       1.95 MB/s          4.2
> Disk  Sequential 64.0 Read                   137.54 MB/s          7.1
> Disk  Sequential 64.0 Write                  196.84 MB/s          7.3
> Average Read Time with Sequential Writes     5.675 ms          5.8
> Latency: 95th Percentile                     44.298 ms          1.9
> Latency: Maximum                             76.917 ms          7.8
> Average Read Time with Random Writes         17.055 ms          2.9
> Total Run Time 00:00:32.72
```

Figure 85 - HDD Performance of Windows 10 Client for RPi 4

RPi 4 NAS HDD

```
pi@raspberrypi:~ $ sudo hdparm -Tt /dev/sdc1

/dev/sdc1:
 Timing cached reads:   1950 MB in  2.00 seconds = 976.09 MB/sec
 Timing buffered disk reads: 382 MB in  3.01 seconds = 126.90 MB/sec
```

Figure 86 - HDD Performance of Hard Drive Attached to RPi 4 NAS

Windows 10 Client HDD for UCS C220-M3S

```
C:\Users\Administrator>winsat disk -drive e
Windows System Assessment Tool
> Running: Feature Enumeration ''
> Run Time 00:00:00.00
> Running: Storage Assessment '-drive e -ran -read'
> Run Time 00:00:01.00
> Running: Storage Assessment '-drive e -seq -read'
> Run Time 00:00:01.98
> Running: Storage Assessment '-drive e -seq -write'
> Run Time 00:00:01.47
> Running: Storage Assessment '-drive e -flush -seq'
> Run Time 00:00:02.20
> Running: Storage Assessment '-drive e -flush -ran'
> Run Time 00:00:01.83
> Disk  Random 16.0 Read                        281.28 MB/s          8.0
> Disk  Sequential 64.0 Read                    1184.04 MB/s          8.6
> Disk  Sequential 64.0 Write                   992.28 MB/s          8.4
> Average Read Time with Sequential Writes      0.877 ms          7.7
> Latency: 95th Percentile                      2.104 ms          7.6
> Latency: Maximum                              7.014 ms          8.3
> Average Read Time with Random Writes          0.733 ms          8.5
```

Figure 87 - HDD Performance of Windows 10 Client for UCS C220-M3S

UCS C220-M3S NAS HDD

```
root@rhys-nas:~# hdparm -Tt /dev/sdb1

/dev/sdb1:
 Timing cached reads:   17074 MB in  1.97 seconds = 8648.19 MB/sec
 Timing buffered disk reads: 622 MB in  3.00 seconds = 207.13 MB/sec
```

Figure 88 - HDD Performance of Hard Drive Attached to UCS C220-M3S NAS

RPi 4

HTOP Output

```
  0[||||||                                          11.6%]   Tasks: 40, 7 thr; 3 running
  1[||||||||||||||||||||||||||||||||||||||||||||||||99.3%]   Load average: 0.77 0.34 0.20
  2[||||||||||                                      17.9%]   Uptime: 06:37:54
  3[||||||||||||                                    20.4%]
Mem[||||||||||||||||||||||||||||||||||||||||||||181M/3.71G]
Swp[                                          0K/100.0M]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
18430 pi         20   0 24848  8756  6044 R 122.  0.2  2:18.94 proftpd: pi - 192.168.1.47: STOR 100gb.test
19014 pi         20   0  8572  4064  2720 R  1.6  0.1  0:00.08 htop
  818 root       20   0  196M 20236 16356 S  0.8  0.5  0:02.08 php-fpm: master process (/etc/php/7.4/fpm/php-fpm.conf)
    1 root       20   0  164M 10612  7324 S  0.0  0.3  0:10.55 /sbin/init
  154 root       20   0 49116 12728 11740 S  0.0  0.3  0:03.31 /lib/systemd/systemd-journald
```

Figure 89- HTOP Output from NAS Transferring 1TB File from Windows 10 Client to RPi 4 NAS

Upload and Download Results

All data in the following section is recorded from transferring files to and from the RPi 4 NAS in seconds and rounded to two decimal places.

From Client to FTP

File/Run	1GB	10GB	100GB	1TB
1	9.68	92.74	1076.14	10874.33
2	11.46	101.59	1069.71	10722.22
3	9.89	93.88	1064.82	10750.65

4	13.34	107.84	1127.29	10747.44
Average	11.09	99.01	1084.49	10773.66

Figure 90 - Table of RPi 4 NAS Transfer Times from Windows 10 Client to NAS

From FTP to Client

File/Run	1GB	10GB	100GB	1TB
1	13.03	128.96	1269.38	13229.13
2	13.49	129.90	1274.07	12934.53
3	13.06	134.46	1268.55	12994.47
4	12.78	124.89	1252.62	12594.46
Average	13.09	129.55	1266.16	12938.15

Figure 91 - Table of RPi 4 NAS Transfer Times from NAS to Windows 10 Client

UCS C220-M3S

HTOP Output

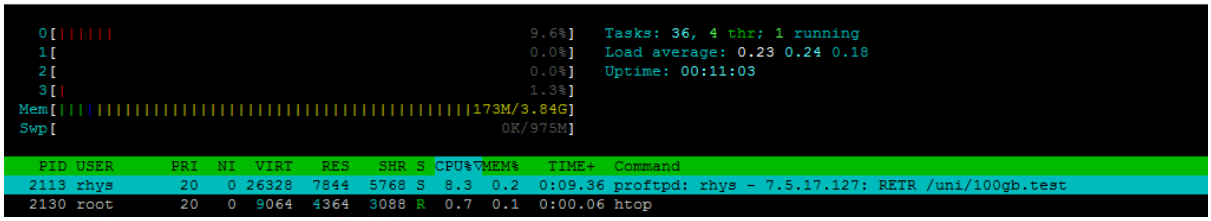

Figure 92 - HTOP Output from NAS Transferring 1TB File from Windows 10 Client to UCS NAS

Upload and Download Results

All data in the following section is recorded from transferring files to and from the UCS C220-M3S NAS in seconds and rounded to two decimal places.

From Client to FTP

File/Run	1GB	10GB	100GB	1TB
1	10.66	103.90	1045.91	10620.98
2	14.22	111.56	1098.62	10684.64
3	14.16	104.81	1032.48	10680.65
4	14.75	104.89	1023.26	10739.41
Average	13.45	106.29	1050.07	10681.42

Figure 93 - Table of UCS C220-M3S NAS Transfer Times from Windows 10 Client to NAS

From FTP to Client

File/Run	1GB	10GB	100GB	1TB
1	9.11	90.93	982.07	9467.59
2	9.30	135.07	1168.54	10646.89
3	9.17	91.23	912.52	10109.77
4	9.12	91.28	939.47	11022.55
Average	9.18	102.13	1000.65	10311.70

Figure 94 - Table of UCS C220-M3S NAS Transfer Times from NAS to Windows 10 Client

Evaluation and Analysis

This section will outline the results of the NAS testing with the Download performance being the copying of the files from the NAS to the Windows 10 client, then the Upload being the copying of the file from the Windows 10 client to the NAS. The figures below highlight the direction of where the file is being transferred in each instance.

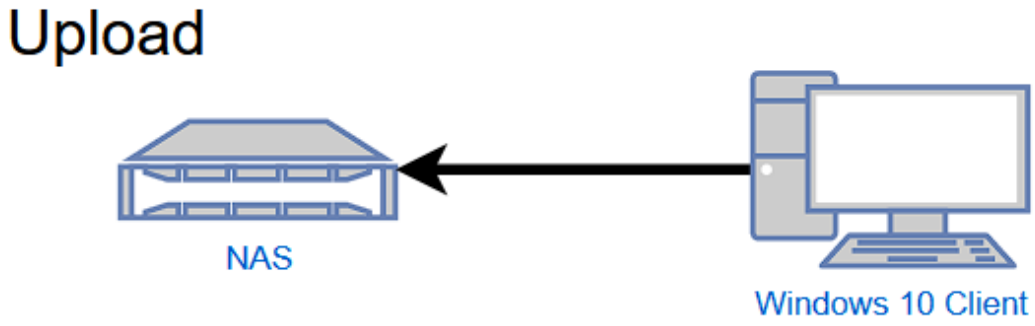

Figure 95 - Diagram of NAS Upload

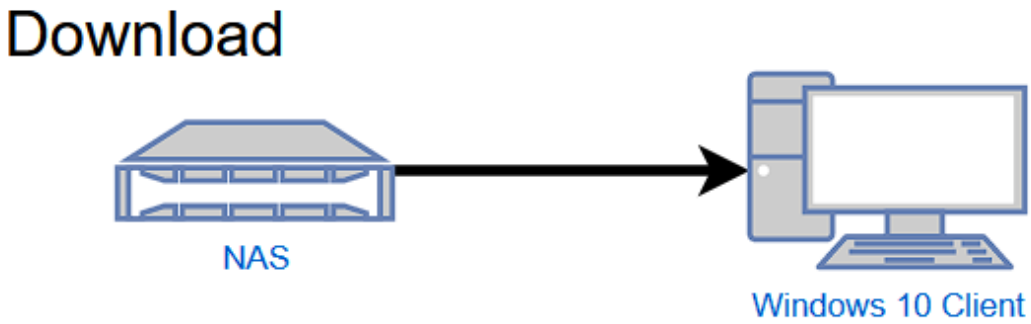

Figure 96 - Diagram of NAS Download

During this test both the time to transfer and the rate of the transfer will be considered. The rate of transfer will be considered when investigating potential bottlenecks in each system. The time to transfer will be used to directly compare each of the systems. The calculation below will be used to output the rate of transfer:

$$Rate\ of\ Transfer\ (MB/s) = \frac{Size\ of\ File\ Transferred\ (MB)}{Time\ to\ Transfer\ (s)}$$

RPi 4

Download

	1GB	10GB	100GB	1TB
Rate of Transfer	78.23	79.04	80.87	81.05

Figure 97 - Table of Rate of Transfer Downloading from RPi 4 NAS

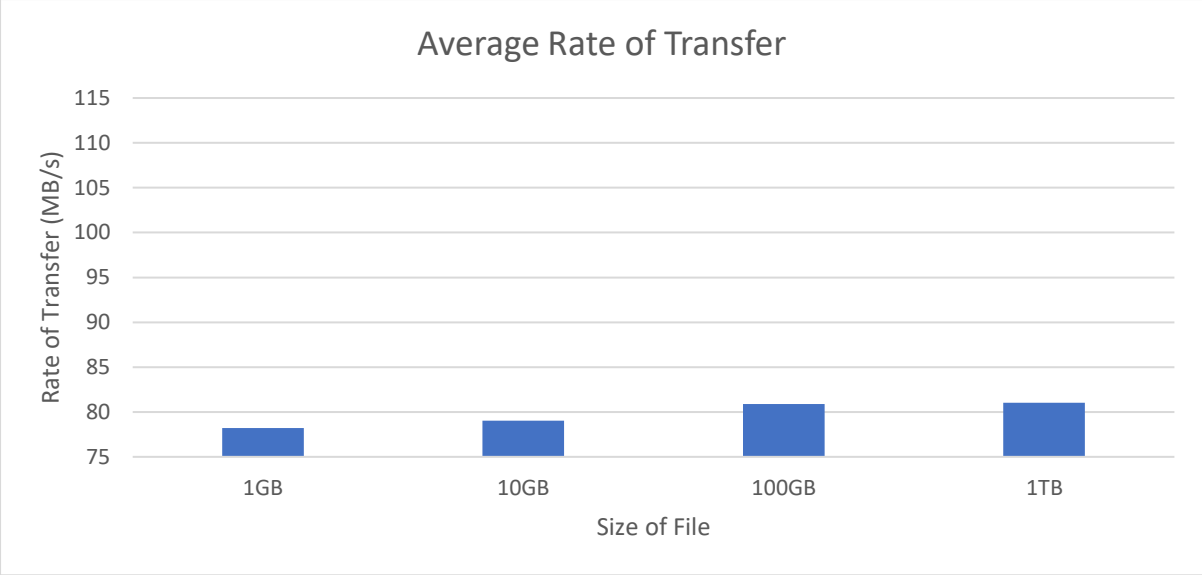

Figure 98 - Chart of Rate of Transfer Downloading File from RPi 4 NAS

Upload

	1GB	10GB	100GB	1TB
Rate of Transfer	92.31	103.42	94.42	97.33

Figure 99 - Table of Rate of Transfer Uploading to RPi 4 NAS

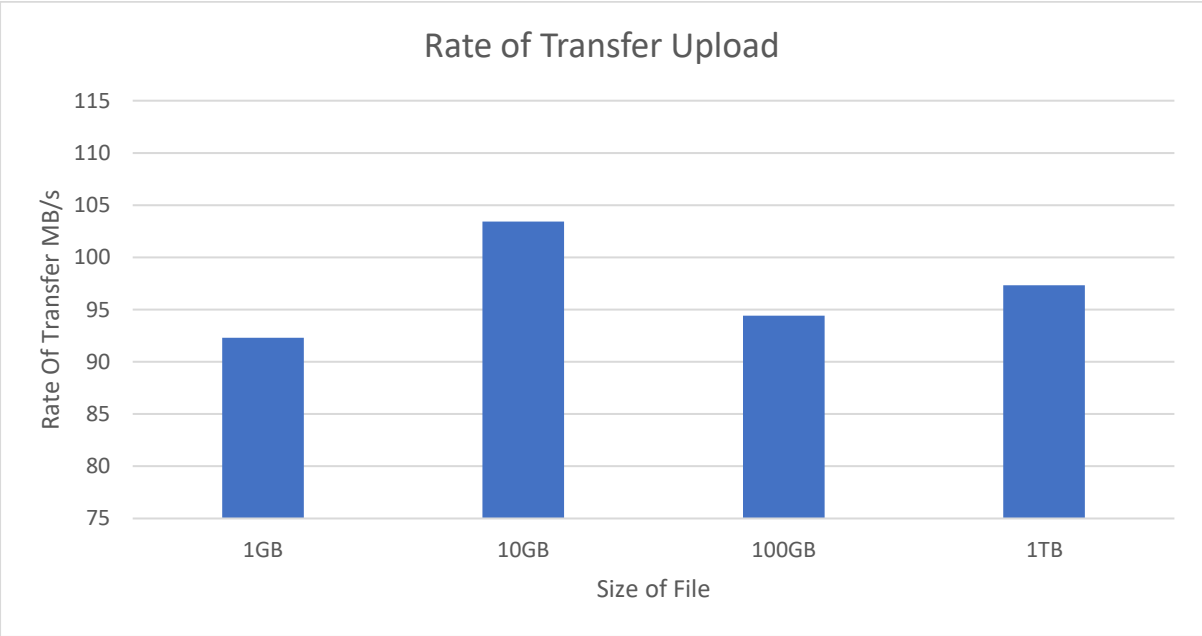

Figure 100 - Chart of Rate of Transfer Uploading File to RPi 4 NAS

UCS C220-M3S

Download

	1GB	10GB	100GB	1TB
Rate of Transfer	111.55	100.26	102.33	101.69

Figure 101 - Table of Rate of Transfer Downloading from UCS C220-M3S NAS

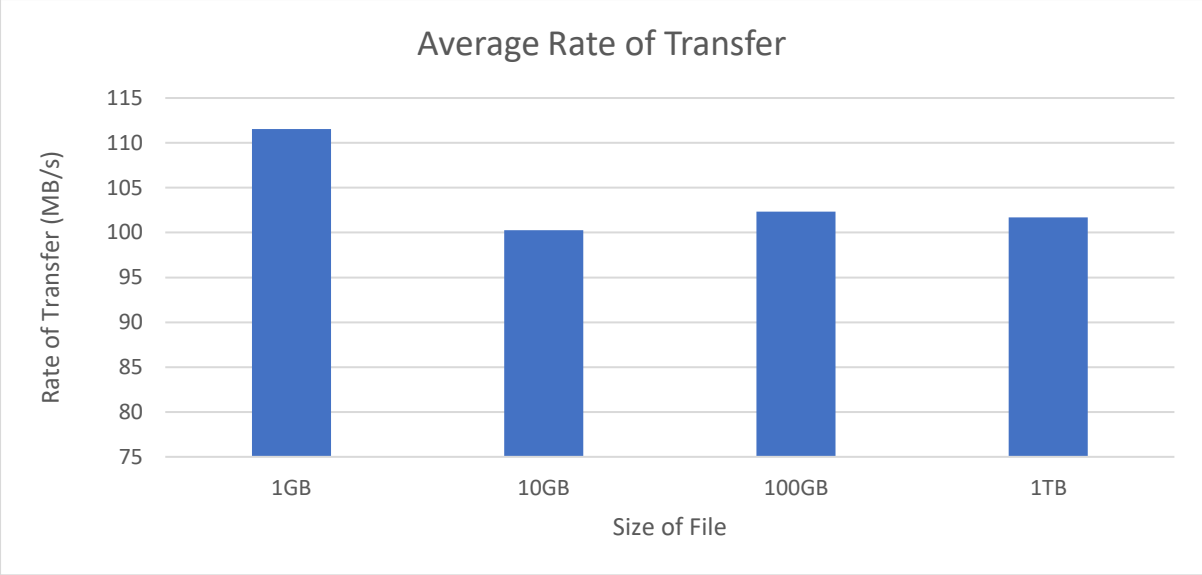

Figure 102 - Chart of Rate of Transfer Downloading from UCS C220-M3S

Upload

	1GB	10GB	100GB	1TB
Rate of Transfer	76.13	96.34	97.52	99.90

Figure 103 - Table of Rate of Transfer Uploading to UCS C220-M3S

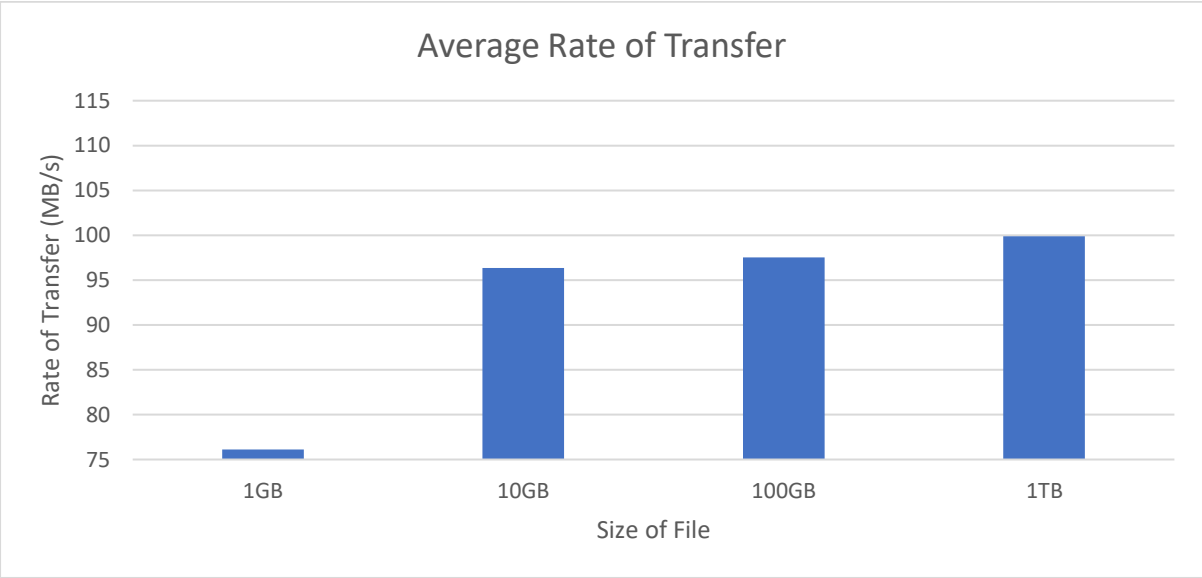

Figure 104 - Chart of Rate of Transfer Uploading File to UCS C220-M3S NAS

The speeds experienced downloading files from the RPi NAS were on average 47.1MB/sec less than the read speed of the drive (Shown in Figure 86). The speeds seen were also only 45.2MB/sec away from fully saturating a Gigabit link, which is the max theoretical speed between the devices tested. The average download speed seen by the RPi 4 is also 24.16MB/sec slower on average than the UCS. For files of 1GB in size the results downloading from the RPi 4 NAS were 3.91s slower on average than the UCS C220-M3S NAS. These differences may also be made worse with the difference in Windows 10 client drive performance. The Windows 10 client for the UCS C220-M3S NAS had a read and write speed of 1184.04 and 992.28MB/sec respectively, this may contribute to some of the

performance differences seen. As well as the download results, the upload results for the RPi 4 were on average 96.87MB/sec which is 40.67MB/sec slower than the read speed of the Windows 10 client's disk. It was also only 28MB/sec slower than full Gigabit transfer speeds. When the speed of uploading to the RPi 4 NAS is compared to the speed of uploading to the UCS C220-M3S NAS, the RPi 4 NAS actually outperforms the UCS C220-M3S. This instance has an average speed to upload to the UCS C220-M3S NAS being 92.47MB/sec. The RPi 4 may get better results when an SSD or faster HDD is used, the potential gains of this may then also be limited by the gigabit NIC on the RPi 4. Equally the UCS C220-M3S may also yield better results with faster drives and/or a 10Gbit PCIe NIC added to the chassis, the 10Gbit NIC will also improve the performance of the iperf results seen in Figure 44.

Conclusions

The speeds seen from both the UCS C220-M3S and RPi 4 were relatively similar to each other with the largest difference being seen in the download speeds from each system. With the downloads from the UCS C220-M3S being 24.16MB/sec faster than the download speed seen from the RPi 4. For a small business, for example, a local estate agents uploading an image or video tour that is a couple of megabytes in size. The employees at the business are not likely to notice the performance difference between the RPi 4 and the UCS C220-M3S. The results found in the testing were also getting close to the theoretical network performance limitations of the systems tested.

7 Power Draw of Systems

Results Found

The top and htop commands are used as this shows the CPU and RAM usage of the systems, top providing a more general overview where htop shows the status of all the cores in the machine. The Linux Kernel compilation was therefore used to find the power draw of these systems as this output the max power draw seen for each system due to creating a test load on the system.

RPi 4

Output of top command whilst running Linux kernel compile.

```
top - 11:47:54 up 15 min,  2 users,  load average: 3.92, 2.18, 0.94
Tasks: 165 total,   5 running, 160 sleeping,   0 stopped,   0 zombie
%Cpu(s): 94.3 us,  5.5 sy,  0.0 ni,  0.0 id,  0.1 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :   1872.2 total,     79.8 free,    267.8 used,   1524.6 buff/cache
MiB Swap:    100.0 total,     99.2 free,      0.8 used.   1504.6 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
20424 pi        20   0   86824  76240  17408 R  99.7   4.0   0:04.17 ccl
20339 pi        20   0   82068  72572  17324 R  99.3   3.8   0:06.25 ccl
20451 pi        20   0   62340  52964  16996 R  71.4   2.8   0:02.17 ccl
20464 pi        20   0   48064  32476  11036 R  27.6   1.7   0:00.84 ccl
20062 pi        20   0    2584   2036   1416 S   0.7   0.1   0:00.08 make
   12 root      20   0       0      0      0 I   0.3   0.0   0:00.55 rcu_sched
   92 root       0 -20       0      0      0 I   0.3   0.0   0:01.18 kworker/0:1H-mmc_complete
  261 root      20   0       0      0      0 I   0.3   0.0   0:04.29 kworker/u8:2-ext4-rsv-conversion
20425 pi        20   0   11260   3020   2568 R   0.3   0.2   0:00.04 top
    1 root      20   0   33796   8744   6964 S   0.0   0.5   0:03.55 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
    8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
   10 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
   11 root      20   0       0      0      0 S   0.0   0.0   0:00.40 ksoftirqd/0
   13 root      rt   0       0      0      0 S   0.0   0.0   0:00.01 migration/0
```

Figure 105 - Output of top Command from RPi 4 2GB

```
  0[||||||||||||||||||||||||||||||||||||||||||||99.4%]   Tasks: 51, 25 thr; 4 running
  1[||||||||||||||||||||||||||||||||||||||||||||100.0%]   Load average: 3.46 1.46 1.74
  2[||||||||||||||||||||||||||||||||||||||||||||99.3%]   Uptime: 6 days, 05:13:44
  3[||||||||||||||||||||||||||||||||||||||||||||99.3%]
Mem[||||||||||||||||||||||||||||||||||||||143M/3.75G]
Swp[|                                   768K/100.0M]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
28322 pi         20   0 51264 35864 11128 R 52.5  0.9  0:00.80 /usr/lib/gcc/arm-linux-gnueabihf/10/c
28431 pi         20   0 31144 16096 10904 R  6.6  0.4  0:00.10 /usr/lib/gcc/arm-linux-gnueabihf/10/c
19021 pi         20   0  4556  3928  1352 S  2.6  0.1  0:01.85 make -f ./scripts/Makefile.build obj=
17224 pi         20   0  9124  3880  2824 R  0.7  0.1  0:00.72 htop
27207 pi         20   0  2692  2128  1480 S  0.7  0.1  0:00.13 make -f ./scripts/Makefile.build obj=
27804 pi         20   0  2576  2072  1480 S  0.7  0.1  0:00.04 make -f ./scripts/Makefile.build obj=
27886 pi         20   0  2616  1972  1392 D  0.7  0.1  0:00.03 make -f ./scripts/Makefile.build obj=
```

Figure 106 - Output of HTOP Command from RPi 4 4GB

The USB-C to USB-C Multimeter only highlights the Voltage (V) and Current (A) used by the RPi 4 at the time of measurement. This however means that the Power Law must be applied to calculate the watts. To do this the average voltage and current was taken to perform the calculations.

$$P = V \; x \; A$$

Workload	V	A	W
Idle	5.25	0.3	1.575
Linux Compile 1	5.25	0.8	4.2
Linux Compile 2	5.25	0.9	4.725
Linux Compile 3	5.25	0.7	3.675

Figure 107 - Table of Power Draw Readings Gathered from USB-C to USB-C Multimeter

When this data was initially collected it was believed that the figures for power usage from the Multimeter was wrong. Following this a different variety of Multimeter was ordered. This new Multimeter sits between the power brick and outlet on the wall. This confirmed that there was a slight variation to these figures however not as large a deviation as was initially suspected.

Figure 108 - Image of USB-C to USB-C Multimeter Reading Idle power draw of RPi 4

The image above shows the original Multimeter used, this configuration sits between the power brick and power cable for the RPi 4. This was then changed in favour of the configuration below that better matched the traditional server setup and how power draw was measured for this, as the configuration of the server takes the power draw measurements between the plug socket and the wall outlet.

Figure 109 - Multimeter Configuration to Check Accuracy of USB-C to USB-C Multimeter

The table below shows the readings gathered from the outlet Multimeter. The average from the USB-C to USB-C Multimeter is 4.2W vs the average from the outlet Multimeter of 5.2W. This shows a 1W difference, this could be due to inefficiencies with the power brick.

Test	Average Power Draw in Watts
Idle	2.1
Linux Kernel Compile 1	5.2
Linux Kernel Compile 2	5.3
Linux Kernel Compile 3	5.1

Figure 110 - Table of Power Draw Readings Gathered from Outlet Multimeter

UCS C220-M3S

Test	Average Power Draw in Watts
Idle	115
Linux Kernel Compile 1	300
Linux Kernel Compile 2	298
Linux Kernel Compile 3	299

Figure 111 - Table of Power Draw Readings Gathered from Panduit G5 Web GUI

Status

Outlet Name
OUTLET32

Current(A)
1.27

Voltage(V)
235.9

Power(VA)
300

Watts(W)
298

Power Factor
1.00

Figure 112 - Power Draw as Measured from Panduit Web GUI

Sensors

Power Supply | Fan | Temperature | Voltage | Current | LEDs | Storage

Properties

Redundancy Status: **Not Available**

Discrete Sensors

Sensor Name	Status	Reading
PSU1_STATUS	Critical	input lost
PSU2_STATUS	Normal	present
PSU1_PWRGD	Critical	bad
PSU1_AC_OK	Critical	bad
PSU2_PWRGD	Normal	good
PSU2_AC_OK	Normal	good

Threshold Sensors

Sensor Name	Status	Reading (Watts)	Warning Threshold Min	Warning Threshold Max	Critical Threshold Min	Critical Threshold Max
POWER_USAGE	Normal	328	N/A	N/A	N/A	800
PSU1_POUT	Normal	0	N/A	624	N/A	648
PSU2_POUT	Normal	280	N/A	624	N/A	648
PSU1_PIN	Normal	0	N/A	720	N/A	744
PSU2_PIN	Normal	328	N/A	720	N/A	744

Figure 113 - Power Draw as Measured from CIMC

Evaluation and Analysis

Using the tables above (Figure 110 and Figure 111) and the data gathered from both the Multimeter and Panduit G5 PDU, a full comparison of the energy cost to complete a Linux kernel compile. The cost of the idle systems outside business hours (5pm to 9am) and consequently the average cost to run each system over business hours (9am to 5pm).

The graph below (Figure 114) illustrates the power draw of each system in 3 given scenarios bullet pointed below:

- Power draw of the system in an idle state - this is the average power drawn in watts from the outlet.
- Power draw running Linux kernel compile – this is the average power drawn in watts from the outlet.
- Max power draw seen – This is the upper power draw seen from each system during stress tests and benchmarking.

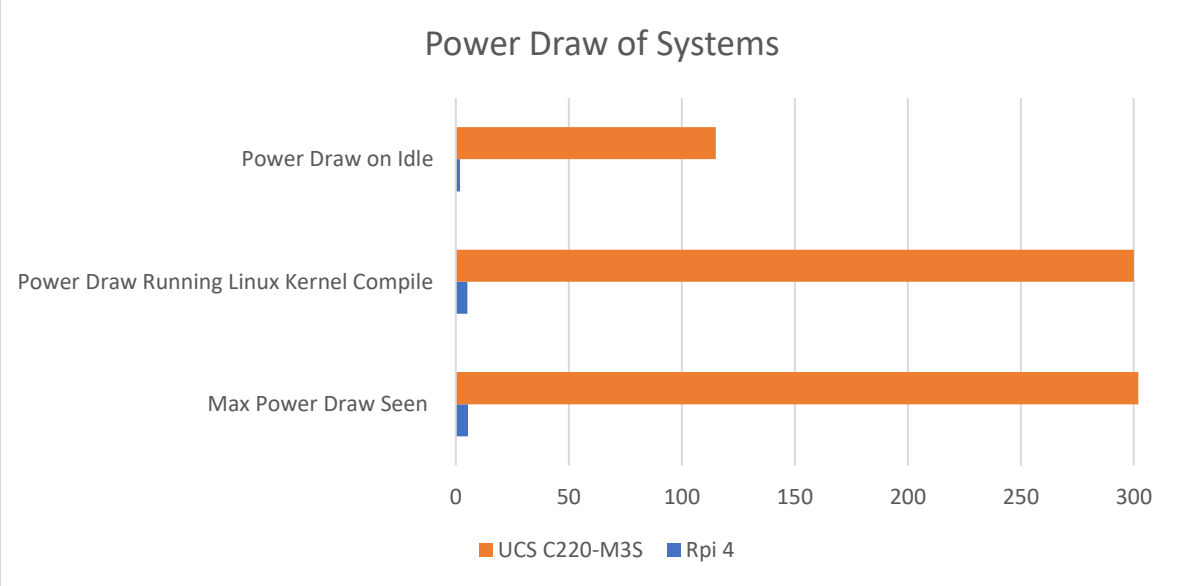

Figure 114 - Graph Illustrating the Power Draw of Systems

Energy Usage of Systems

This section will take the figures for power draw found earlier and apply calculations to ascertain the Cost/kWh usage of each system. To convert the Watt figures found earlier to kWh's the calculation below was used:

$$E_{(kWh)} = \frac{T_{(h)} x\ P_{(W)}}{1000}$$

Energy Cost of Linux Kernel Compile

The calculation below is used to determine the total power draw for full duration of the test. This calculation can be used for both the RPi 4 and for the UCS C220-M3S.

$$\frac{Avg\ Power\ Draw\ x\ (\frac{Time\ To\ Compile}{3600})}{1000} = Total\ Power\ Used\ in\ kWh$$

System	Average Time to Compile	Average Power Draw	Total Power Used (kWh) Rounded to 3sf
RPi 4	2665	5.2	0.00385
UCS C220-M3S	167	299	0.01387

Figure 115 - Table Comparing Average Time to Compile on Systems

Although the UCS C220-M3S compiles in less than 1/10th of the time taken by the RPi4, the RPi 4 is more energy efficient as it takes less kWh to achieve same result. This for services that the RPi 4 can run and perform on par with the

Energy Cost of Idle Time Between Business Hours

The calculation below is used to determine the total power draw for the Idle hours between standard business operating hours. This calculation can be used for both the RPi 4 and for the UCS C220-M3S.

$$\frac{Idle\ Power\ Draw\ x\ Time\ Between\ Business\ Hours}{1000} = Total\ Power\ Used\ in\ kWh$$

System	Idle Power Draw (W)	Idle Time (Hours)	Total Power Used (kWh)
RPi 4	0.0021	16	0.0336
UCS C220-M3S	0.1150	16	1.8400

Figure 116 - Table Comparing Idle Power Draw of Systems

As shown in the table above the RPi 4 uses less energy on idle between 5PM and 9AM the next day.

RPi 4

- Mon 9am – Fri 5PM = 0.1344kWh
- Weekend Fri 5PM – Mon 9am = 0.1344kWh
- Week with idle over weekend = 0.2688kWh
- Assuming power off over weekend = 0.1344kWh

RPi 4	Idle Time Mon - Fri	Idle Time Weekend	No Weekend Power Down	Over 52 Week period	Weekend Power Down	Over 52 Week period
Energy kWh	0.1344	0.1344	0.2688	13.9776	0.1344	6.9888

Figure 117 - Table Outlining Energy Usage of RPi 4 in Idle Hours

UCS

- Mon 9am – Fri 5PM = 7.36kWh
- Weekend Fri 5PM – Mon 9am = 7.36kWh
- Week with idle over weekend = 14.72kWh
- Assuming power off over weekend = 7.36kWh

UCS C220-M3S	Idle Time Mon - Fri	Idle Time Weekend	No Weekend Power Down	Over 52 Week period	Weekend Power Down	Over 52 Week period
Energy kWh	7.36	7.36	14.72	765.44	7.36	382.72

Figure 118 - Table Outlining Energy Usage of UCS C220-M3S in Idle Hours

In total over the space of a 52-week year the business could be saving between 375 and 751 kWh by migrating their core services from a traditional server to the RPi 4.

Energy Cost to Run Systems

The calculation below is used to determine the total power draw for full duration of business hours assuming a lower bound of 50% utilisation and 90% upper bound. This calculation can be used for both the RPi 4 and for the UCS C220-M3S. As this project only covers SME's MNE's will not be considered for this calculation as their power utilisation would be covered on a 24/5 or 24/7 basis dependant on the company.

$$\frac{50\%\ Power\ Draw\ x\ Business\ Hours}{1000} = Total\ Power\ Used\ in\ kWh$$

$$\frac{90\%\ Power\ Draw\ x\ Business\ Hours}{1000} = Total\ Power\ Used\ in\ kWh$$

System	Power Draw (kWh)	Business Hours (Hours Per Week)	Total Power Used (kWh)
RPi 4 – 50%	0.0026	40	0.1040
RPi 4 – 90%	0.0047	40	0.1880
UCS C220-M3S – 50%	0.0575	40	2.3000
UCS C220-M3S – 90%	0.1035	40	4.1400

Figure 119 - Table Outlining Power Usage of Systems in a Working Week

Cost To Run Analysis

To calculate the cost to run each of the systems is extremely situational and cannot be exactly determined as what may be the cost to run for one business could be significantly lower or higher dependant on the usage of each system.

With the current rise in cost of electricity in the UK this could then equate to £105 – 210 in savings to the business per year per server assuming a 28p/kWh electricity cost. This energy saving could buy 2-4 RPi's or even be used to reinvest into other elements of the business.

When the cost to run the RPi 4 is then also compared to, the cost to run GCP lowest specification VM, from Figure 23. The RPi 4 could be purchased, and electricity paid over a year for less than £100 where GCP's smallest VM offering will cost the business $624 USD (roughly £500 as of April 2022) assuming the cost of GCP taken from Figure 23.

E-Waste

Electronic waste (E-Waste) is the waste produced in both the production and disposal of electronic equipment. Over the past decade, technology and electrical equipment have become far more ubiquitous than in prior decades. This can be seen from the 44.7 million metric tonnes (Mt) in E-Waste generated in 2016 an increase of 3.3Mt from 2014, furthermore it was estimated in 2017 that this would increase a further 17% by 2021 (United Nations University, 2017). This project hopes to highlight to businesses that any old tech that they have can be utilised for tasks listed in requirement 4 and that if they invest in technology like a RPi 4 Model B there are other non-business critical services that the RPi 4 can undertake. As the business grows from a small into a medium and even large enterprise consideration should be given to the services run and how efficiently they can be run.

As well as the energy savings from moving some business-critical services to the RPi 4 the business will not need to E-Waste the RPi 4. These could then be used for staff to access the company VDI, or even used for other aspects of the business like a systems monitor for their servers etc.

8 Conclusion

From the analysis of the power draw for the systems, it is seen that the power draw of the RPi 4 is substantially lower than that of the server. Furthermore, the RPi 4 is powerful enough to run most business-critical services for SME's, the services that can be run on the RPi 4 are as follows:

- DNS/Pi-Hole
- NAS
- DHCP
- Webserver – For smaller businesses

Outside of the tools tested in this project, there may be a plethora of tools that may also be implemented across multiple SME's.

9 Example Configurations

Considering the points above referencing the RPi 4's performance of business-critical services, some example configurations will be highlighted below. These will utilise varying levels of redundancy and tools.

Pi-Hole Configuration on 2xRPi 4 Configuration

The example below is a cluster of 2 RPi 4 4GB's running the following configurations hardware could be purchased for £177 including VAT as of April 2022. The two RPi 4's will be broken down as follows:

- Primary DNS Pi-Hole on RPi 1
- Secondary DNS Pi-Hole on RPi 2

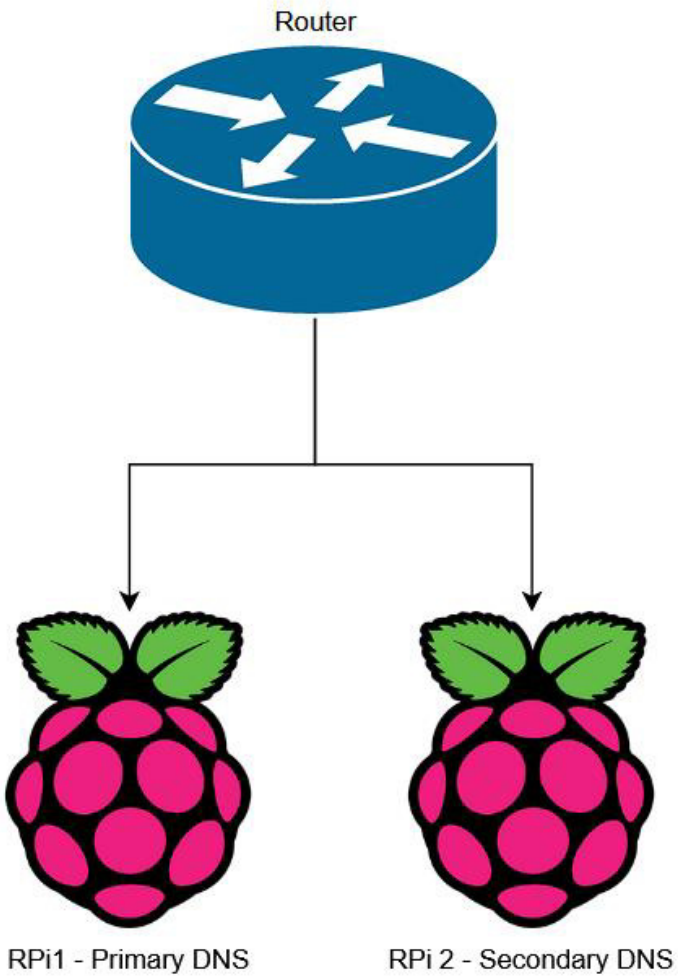

Figure 120 - Example Network Config for SME Using 2xRPi 4 for DNS

The configuration above allows for DNS to be spread across the two RPi's with redundancy at ≈1/6th of the base cost of the UCS C220-M3S utilised in this project. This allows for DNS queries to still be resolved whilst one RPi is undertaking downtime to update or if RPi 1 crashes the business can continue without any noticeable issues that would be seen running a singular Pi-Hole DNS server. As well as the significantly lower base cost of the system the energy used by the above config would be between 4.2W for idle hours and 10.4W at full load, meaning that the configuration above would

run at full load for less than 1/10th of the power usage of the traditional server running at idle. As well as this, the RPi 4's under full load will run at less than 1/28th of one UCS C220-M3S server running under full load. It should also be considered that for this configuration an alternative if energy saving is the main concern is to not run Pi-Hole and just use the routers default DNS or even changing the routers default DNS to point to the likes of Google DNS, OpenDNS or Cloudflare etc.

Redundant Webserver on 3xRPi 4 Configuration

The example below is a cluster of 3 RPi 4 4GB's running the following configurations hardware could be purchased for £265.50 including VAT as of April 2022. The three RPi 4's will be broken down as follows:

- Primary Webserver on RPi 1
- Secondary Webserver on RPi 2
- Load balancer running in a Docker container on RPi 3

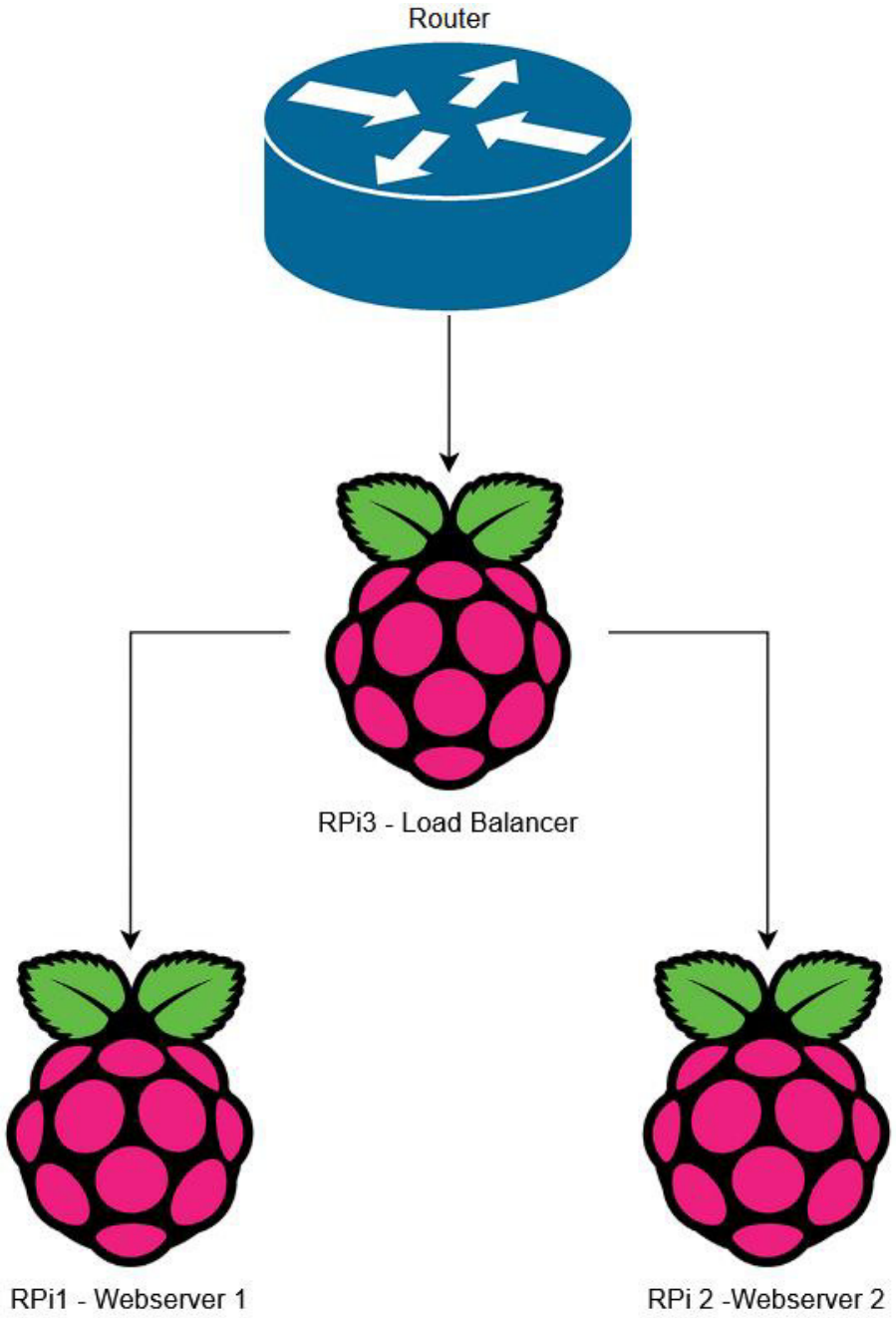

Figure 121 - Example Network Config for SME Using 3xRPi 4 for Company Webserver

The configuration above allows for web requests to be handled across the two RPi's with redundancy at ≈1/4th of the base cost of the UCS C220-M3S utilised in this project. This allows for DNS queries to still be resolved whilst one RPi is undertaking downtime to update or if a RPi crashes customers can continue to visit the company's website. As well as the significantly lower base cost of the system the energy used by the above config would be between 8.4W for idle hours and 20.8W at full load. This means that the configuration above would run at full load for less than 1/5th of the power usage of the traditional server running at idle, additionally, this the RPi 4's under full load will run at less than 1/14th of one UCS C220-M3S server running under full load.

Webserver and Business Critical Services on 4xRPi 4 Configuration

The example below is a cluster of 4 RPi 4 4GB's running the following configurations hardware could be purchased for £354 including VAT as of April 2022. The four RPi 4's will be broken down as follows:

- Main company webserver on RPi 1 – Webserver
- Main company AD DC server on RPi 2 – AD
- Primary DNS and DHCP server on RPi 3 – DNS and DHCP
- Redundant DNS, AD, DHCP and webserver on RPi 4 – Redundant

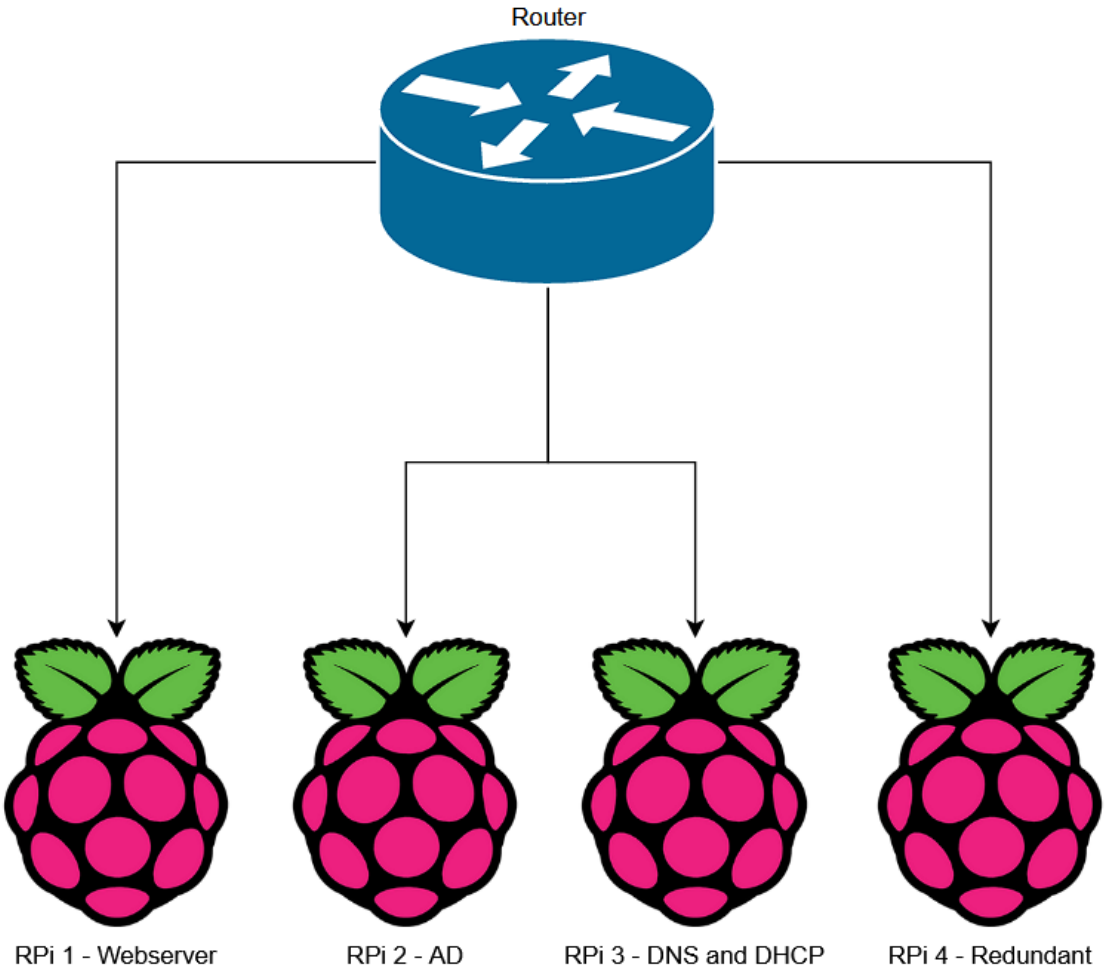

Figure 122 - Example Network Config for SME Using 4xRPi 4

The configuration above allows for key services to be spread across multiple RPi's with redundancy at ≈1/3rd of the base cost of the UCS C220-M3S utilised in this project. It will also provide the SME

with redundancy across all business-critical services. As well as the significantly lower base cost of the system the energy used by the above config would be between 8.4W for idle hours and 20.8W at full load, meaning that the above configuration would run at full load for less than 1/5th of the power usage of the traditional server running at idle. The RPi 4's under full load will run at less than 1/14th of one UCS C220-M3S server running under full load.

10 Future Work

Future work to be considered for the further development of this project includes the items below:

- More in-depth testing of AD
- Testing of more tools
- Deeper Analysis into the stability
- Business use cases of home assistant server
- Research into the business cost associated with downtime
- Potential testing of Win Server on RPi

The tools that could be tested further to those tested within this project, include but are not limited to the following services:

- Systems monitor (Grafana)
- Thin Client to remote into a VDI instance
- VPN Server
- Private Cloud
- Mail server
- NTP server
- Home Assistant or OpenHab

These services once tested would give a broader scope to the services that could be hosted on the RPi 4. It would also add further depth to the types of services, than what has been highlighted within this project.

It was originally intended that a CSP would also be tested for performance metrics, however, due to the time constraints of this project this was not tested but the cost considered.

The likes of Home Assistant or OpenHab, this type of home automation technology may be beneficial to business as they can then generate schedules for IoT devices utilised around the office. These schedules could be utilised to potentially set timings for lights and this alongside wider applications could further save the energy costs to the business.

For practical business use the stability of these systems would need to be considered. As failure rates of the systems will play a key part in the feasibility of whether the RPi 4 could practically be used for business-critical workloads and use cases. A deeper analysis into stability would also further reassure SMEs that their choice to use the RPi 4 to host their services was sound.

As well as an investigation into the stability of services hosted on the RPi 4, businesses would benefit from an investigation into the cost associated with downtime using the RPi 4 vs an established replacement system. These replacement systems are often offered through subscription service plans from vendors such as Cisco, Juniper, Dell etc.

With the recent compatibility released by Microsoft for ARM64 in Windows, this could further down the line include ARM64 support for Windows Server. If this capability is added this could lead to

further testing on the RPi 4. This would be done by testing how the RPi 4 performs running both the desktop and lite versions of Windows server.

Finally, testing of other Single Board Computers (SBC's) such as the Udoo Bolt V8, Asus Tinker Board and Nvidia Jetson etc. Testing these other SBCs could then provide more depth to the analysis, especially with the Udoo Bolt V8 running on x86 CPU architecture therefore removing the need for certain services to support the less widely supported ARM64 CPU architecture.

11 Reflection on Learning

During this project I developed my knowledge of the Linux OS, this has led to me becoming more comfortable with the Linux OS. Whilst l have improved my working knowledge of Linux and the CLI, I understand that there is still lots to learn about the Linux OS. In my learning I found that the granular level of logging in a Linux system allowed me to configure the logging to match what I required from the system. With this knowledge gained of the Linux OS and logging these skills learnt, will be taken forward with me into my career within computing and further developed in the future.

Working on the project was the first practical insight into Proxmox that I had experienced. Although I had my issues with it initially due to not knowing that you can set firewall rules for the datacentre, host and even the VM. Although these detailed firewall settings caused me some confusion in the beginning of the project, I found this newfound knowledge to be helpful in assisting troubleshooting later into the project.

In hindsight I would have taken more time to consider AD and how this could have been better tested. Taking the time to try arranging live deployments of an RPi 4 AD server in business and seeing how this performs for the business.

Challenges Faced

Throughout the project I faced a number of challenges trying to complete it. This included at the beginning of the project when I was trying to work out how AD would be tested and how metrics could be gathered for this service.

I faced a challenge when trying to install some of the services tested, due to blindly following online tutorials on how to set these services up. Some of this led to me breaking the install of RPi OS, upon fixing this issue I began to take the time to think about what each of the commands referenced in the tutorial did and how these commands worked.

Throughout the project once the firewall issues were resolved I had also noticed some weird network dropouts on my VMs running in Proxmox. Unsure whether this was related to my configuration or something on my end, as I could never find any reason as to why these network dropouts were occurring. This did however mean that some of the timeframe of the project was taken trying to resolve this random intermittent networking issue that I could not work out the cause of.

References

Abu Sharkh, M., Jammal, M., Shami, A. & Ouda, A., 2013. *Resource allocation in a network-based cloud computing environment: design challenges,* s.l.: IEEE.

Acácio de Andrade, A., Batista Dietrich, Á., Francisco Blumetti Facó, J. & Reolon Jorge, R., 2020. *Low Cost Solution for Home Brewing and Small Brewing Business Using Raspberry Pi,* s.l.: Springer Nature.

AdGuard, 2022. *Buy a license key | AdGuard.* [Online]
Available at: https://adguard.com/en/license.html
[Accessed 27 April 2022].

Bargain Hardware, 2022. *Refurbished Servers, PCs, Workstations & Parts | Bargain.* [Online]
Available at: https://www.bargainhardware.co.uk
[Accessed 6 February 2022].

Bitwarden, Inc, 2022. *Install and Deploy - Linux | Bitwarden.* [Online]
Available at: https://bitwarden.com/help/install-on-premise-linux/
[Accessed 18 March 2022].

Cisco Systems Inc, 2017. *Cisco UCS C220 M3 Rack Server Data Sheet.* [Online]
Available at: https://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/ucs-c220-m3-rack-server/data_sheet_c78-700626.html
[Accessed 11 February 2022].

Citrix, 2022. *What is VDI? Virtual Desktop Infrastructure.* [Online]
Available at: https://www.citrix.com/en-gb/solutions/vdi-and-daas/what-is-vdi-virtual-desktop-infrastructure.html
[Accessed 11 March 2022].

de la Cruz, J. E. C., Goyzueta, C. A. R. & Cahuana, C. D., 2016. *Intrusion Detection and Prevention System for Production Supervision in Small Businesses Based on Raspberry Pi and Snort,* s.l.: IEEE.

Debian Org, 2021. *Debian -- News -- Updated Debian 11: 11.2 released.* [Online]
Available at: https://www.debian.org/News/2021/20211218
[Accessed 15 March 2022].

European Comission, 2022. *SME definition.* [Online]
Available at: https://ec.europa.eu/growth/smes/sme-definition_en
[Accessed 12 May 2022].

Gibson Research Corporation, 2018. *GRC's | DNS Nameserver Performance Benchmark.* [Online]
Available at: https://www.grc.com/dns/benchmark.htm
[Accessed 10 April 2022].

Imperva, 2022. *What is a Honeypot | Honeynets, Spam Traps & more | Imperva.* [Online]
Available at: https://www.imperva.com/learn/application-security/honeypot-honeynet/
[Accessed 6 February 2022].

Intelligent Servers, 2022. *Intelligent Servers.* [Online]
Available at: https://intelligentservers.co.uk
[Accessed 1 February 2022].

ipchama, 2021. *ipchama/dhammer: DHCP stress tester and benchmark tool.* [Online]
Available at: https://github.com/ipchama/dhammer
[Accessed 9 April 2022].

IT in Stock, 2022. *Cisco UCS C220 M3 UCSC-C220-M3S 2x Quad Core E5-2643 3.30GHz.* [Online]
Available at: https://www.itinstock.com/cisco-ucs-c220-m3-ucsc-c220-m3s-2x-quad-core-e5-2643-330ghz-600gb-24gb-server-48623-p.asp
[Accessed 21 February 2022].

Jackson, C., Gooley, J., Iliesiu, A. & Malegaonkar, A., 2020. *Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide.* s.l.:Cisco Press.

Maulana, H. & Al-Khowarizmi, 2021. *Analyze and Designing Low-Cost Network Monitoring System,* Medan, Indonesia: IOP.

Microsoft Azure, 2022. *Cloud Computing Services | Microsoft Azure.* [Online]
Available at: https://azure.microsoft.com/en-us/
[Accessed 2 February 2022].

Microsoft, 2022. *Active Directory Domain Services Overview.* [Online]
Available at: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview
[Accessed 3 February 2022].

Microsoft, 2022. *STDEV function.* [Online]
Available at: https://support.microsoft.com/en-us/office/stdev-function-51fecaaa-231e-4bbb-9230-33650a72c9b0
[Accessed 03 May 2022].

Microsoft, n.d. [Online].

Miles, C., 2020. *GitHub - Akkadius/glass-isc-dhcp: Glass - ISC DHCP Server Interface.* [Online]
Available at: https://github.com/Akkadius/glass-isc-dhcp
[Accessed 4 February 2022].

Nguye, H.-Q., Loan, T. T. K., Mao, B. D. & Huh, E.-N., 2015. *Low Cost Real-Time System Monitoring,* s.l.: IEEE.

Open Benchmarking Org, 2022. *Timed Linux Kernel Compilation Benchmark - OpenBenchmarking.org.* [Online]
Available at: https://openbenchmarking.org/test/pts/build-linux-kernel-1.13.0
[Accessed 08 March 2022].

PassMark Software, 2022. *ARM Cortex A53 4 Core 1400 MHz vs ARM Cortex A72 4 Core 1500 MHz.* [Online]
Available at: https://www.cpubenchmark.net/compare/ARM-Cortex-A53-4-Core-1400-MHz-vs-ARM-Cortex-A72-4-Core-1500-MHz/4143vs3917
[Accessed 2 March 2022].

Passmark Software, 2022. *ARM Cortex-A72 4 Core 1800 MHz vs Intel Xeon E5-2643 @ 3.30GHz [cpubenchmark.net] by PassMark Software.* [Online]
Available at: ARM Cortex-A72 4 Core 1800 MHz vs Intel Xeon E5-2643 @ 3.30GHz

[cpubenchmark.net] by PassMark Software
[Accessed 12 March 2022].

Pi-Hole, 2022. *Overview of Pi-hole - Pi-hole documentation.* [Online]
Available at: https://docs.pi-hole.net/
[Accessed 2 February 2022].

Portainer, 2022. *Container Management | Kubernetes GUI | Docker Swarm GUI | Portainer.* [Online]
Available at: https://www.portainer.io/
[Accessed 29 March 2022].

Portainer, 2022. *Install Portainer Agent with Docker on Linux - Portainer Documentation.* [Online]
Available at: https://docs.portainer.io/v/ce-2.9/start/install/agent/docker/linux
[Accessed 10 April 2022].

Portainer, 2022. *Install Portainer with Docker on Linux - Portainer Documentation.* [Online]
Available at: https://docs.portainer.io/v/ce-2.9/start/install/server/docker/linux
[Accessed 10 April 2022].

Proxmox, 2022. *Proxmox VE Enterprise Support Subscriptions.* [Online]
Available at: https://www.proxmox.com/en/proxmox-ve/pricing
[Accessed 10 March 2022].

Proxmox, 2022. *Proxmox VE Hardware Requirements.* [Online]
Available at: https://www.proxmox.com/en/proxmox-ve/requirements
[Accessed 20 April 2022].

Raspberry Pi Foundation, 2022. *Operating system images - Raspberry Pi.* [Online]
Available at: https://www.raspberrypi.com/software/operating-systems/
[Accessed 15 March 2022].

Raspberry Pi Foundation, 2022. *Raspberry Pi.* [Online]
Available at: https://www.raspberrypi.com
[Accessed 1 February 2022].

Raspberry Pi Foundation, 2022. *Raspberry Pi 4 Datasheet.* [Online]
Available at: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-mechanical-drawing.pdf
[Accessed 3 February 2022].

Raspberry Pi LTD, 2019. *raspberry-pi-4-datasheet.pdf.* [Online]
Available at: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf
[Accessed 1 March 2022].

Raspberry Pi LTD, n.d. *raspberry-pi-3-b-plus-product-brief.* [Online]
Available at: https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf
[Accessed 1 March 2022].

RedHat, 2020. *What is Infrastructure as Code (IaC).* [Online]
Available at: https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac
[Accessed 5 March 2022].

Stratodesk, 2022. *Stratodesk NoTouch – Thin Client Software.* [Online]
Available at: https://www.stratodesk.com/products/notouch/#1587497476663-a3983ba6-cdf3
[Accessed 1 March 2022].

The Pi Hut, 2022. *The Ultimate Raspberry Pi & Maker Store.* [Online]
Available at: https://www.thepihut.com
[Accessed 1 February 2022].

United Nations University, 2017. *E-waste Rises 8% by Weight in 2 Years as Incomes Rise, Prices Fall - United Nations University.* [Online]
Available at: https://unu.edu/media-relations/releases/ewaste-rises-8-percent-by-weight-in-2-years.html
[Accessed 28 April 2022].

Veritis, n.d. *AWS Vs Azure Vs GCP – The Cloud Platform of Your Choice?.* [Online]
Available at: https://www.veritis.com/blog/aws-vs-azure-vs-gcp-the-cloud-platform-of-your-choice/
[Accessed 13 February 2022].

VMWare, 2022. *What is a hypervisor?.* [Online]
Available at: https://www.vmware.com/topics/glossary/content/hypervisor.html
[Accessed 21 February 2022].

VMWare, 2022. *What is a vSphere Hypervisor? | Free Hypervisor | VMware.* [Online]
Available at: https://www.vmware.com/products/vsphere-hypervisor.html
[Accessed 20 April 2022].

VMWare, 2022. *What is VDI? | Virtual Desktop Infrastructure | VMware Glossary.* [Online]
Available at: https://www.vmware.com/topics/glossary/content/virtual-desktop-infrastructure-vdi.html
[Accessed 11 March 2022].