An Evaluation and Analysis of Computing Virtualization on Raspberry Pi Using Virtual Machines

Vincent Melval Richards

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2020

Vincent Melval Richards

An Evaluation and Analysis of Computing Virtualization on Raspberry Pi Using Virtual Machines

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Flávio de Oliveira Silva

Uberlândia
2020

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

R518e
2020

Richards, Vincent Melval, 1980-
An evaluation and analysis of computing virtualization on Raspberry Pi using virtual machines [recurso eletrônico] / Vincent Melval Richards. - 2020.

Orientador: Flávio de Oliveira Silva.
Dissertação (mestrado) - Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2022.5006
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Silva, Flávio de Oliveira, (Orient.). II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

André Carlos Francisco
Bibliotecário - CRB-6/3408

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 38/2020, PPGCO				
Data:	16 de dezembro de 2020	Hora de início:	09:29	Hora de encerramento:	10:48
Matrícula do Discente:	11812CCP001				
Nome do Discente:	Vincent Melval Richards				
Título do Trabalho:	An Evaluation and Analysis of Computing Virtualization on Raspberry Pi Using Virtual Machines				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Pedro Frosi Rosa - FACOM/UFU; Eduardo Coelho Cerqueira - UFPA e Flávio de Oliveira Silva - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Eduardo Coelho Cerqueira - Belém/PA; Pedro Frosi Rosa e Flávio de Oliveira Silva - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Flávio de Oliveira Silva, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor(a) presidente concedeu a palavra, pela ordem sucessivamente, aos(às) examinadores(as), que passaram a arguir o(a) candidato(a). Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o(a) candidato(a):

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Referência: Processo nº 23117.075312/2020-17 SEI nº 2458951

This dissertation is dedicated to my parents and friends, who encouraged me during the critical phases of this study.

Acknowledgments

Firstly, I would like to express my gratitude to my advisor Prof. Dr. Flávio de Oliveira Silva, for his guidance, knowledge, and motivation. Besides my advisor, I would like to thank my friend and teammate, Rodrigo Moreira, for his continuous work and efforts to make this project a reality. Last but not least, I would like to thank my family and friends for their support throughout the writing of this dissertation.

"The energy of the mind is the essence of life."
(Aristotle)

Abstract

The Internet of Things (IoT) application can offer tremendous opportunities for organizations and institutions to grow and explore innovative ideas. However, several IoT use cases rely on Edge Computing capabilities and the virtualization capacity on edge. The use of virtualization on edge is a crucial requirement in realizing the efficient implementation of IoT. Furthermore, the management and orchestration of computing, storage, and networking resources in the Edge Cloud pose several challenges. Network Function Virtualization (NFV) envisages several Virtual Network Functions (VNF) deployed across the infrastructure. In this scenario, there are issues related to Single Board Computers (SBC) capacity to support compute virtualization based on Virtual Machines (VM). In this work, we evaluate and analyze computing virtualization on Raspberry Pi versions 3 and 4 using VM. Based on the experimental, we show the feasibility of using low-cost devices such as RPi as the edge computing hardware.

Keywords: Cloud Computing, Edge Computing,NFV, Virtualization, IoT..

List of Figures

Figure 1 – Proposed Solution Build-Blocks. 38
Figure 2 – Enabled KVM Virtualization 41
Figure 3 – Experimental Scenario Sequence Diagram. 46
Figure 4 – RPi3 Instances Total Boot Times. 48
Figure 5 – Rpi3 Temperature Measurement. 49
Figure 6 – RPi3 Resource Allocation over Time. 50
Figure 7 – Rpi4 Instances Boot Times. 51
Figure 8 – Rpi4 Temperature Measurement 52
Figure 9 – Rpi4 Resources usage 53
Figure 10 – Comparison of Instances Boot Times 55
Figure 11 – Comparison of Temperature Usage 56
Figure 12 – Comparison of Resources Usage 58

List of Tables

Table 1 – Comparing Single Board Computer (SBC)s CPU, RAM and Cost . . . 24
Table 2 – SBCs Storage Type and Cost 24
Table 3 – Raspberry-Pi4 vs Pi3 25
Table 4 – Full Virtualization, Para-Virtualization, and Hardware-Assisted Virtualization 29
Table 5 – Comparison of Related Works 35

Acronyms list

5G	Fifth Generation Mobile Networks
AR	Augmented Reality
ETSI	European Telecommunications Standards Institute
IoT	Internet of Things
MANO	Management and Orchestration
MEC	Mobile Edge Computing
MR	Mixed Reality
NFV	Networking Function Virtualization
NFVI	Network Functions Virtualization Infrastructure
RPi	Raspberry Pi
RPi3	Raspberry Pi Version 3

RPi4	Raspberry Pi version 4
SBC	Single Board Computer
URLLC	Ultra-Reliable and Low-Latency Communications
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VMs	Virtual Machines
VNF	Virtual Network Function
VNFs	Virtual Network Functions
VR	Virtual Reality

Contents

1 INTRODUCTION . . . 19

1.1 Motivation . . . 20

1.2 Objectives and Research Challenges . . . 20

1.3 Hypothesis . . . 21

1.4 Contributions . . . 21

1.5 Research Method . . . 22

1.6 General Scheme of Dissertation . . . 22

2 LITERATURE REVIEW . . . 23

2.1 Openstack . . . 23

2.2 Single Board Computers . . . 23

2.3 Compute Virtualization . . . 28

2.3.1 VM Based Virtualization . . . 28

2.3.2 Container Based Virtualization . . . 30

2.4 Networking Function Virtualization (NFV) . . . 31

2.4.1 Main Concepts . . . 31

2.4.2 Management and Orchestration . . . 31

2.5 Related Work . . . 32

3 VIRTUAL MACHINE ON THE EDGE . . . 37

3.1 Virtual Functions on the Edge . . . 37

3.2 Architectural Design . . . 38

3.3 Virtual Machines on Raspberry Pi . . . 39

4 EXPERIMENTAL EVALUATION . . . 45

4.1 Testbed Description . . . 45

4.1.1 RPi3 Testbed Description . . . 45

4.1.2 RPi4 Testbed Description . . . 46

4.2 Evaluation Methodology . . . 47
4.3 Discussion . . . 47
4.3.1 Discussion about RPi3 . . . 48
4.3.2 Discussion about RPi4 . . . 51
4.3.3 Discussion about RPi3 x RPi4 . . . 54

5 CONCLUSIONS . . . 61
5.1 Main Contributions . . . 62
5.2 Technical Contributions . . . 62
5.3 Publications . . . 62
5.4 Future Work . . . 63

BIBLIOGRAPHY . . . 65

Chapter 1

Introduction

The Internet of Things (IoT) is a key enabler for a generation of services and applications. IoT's growth and popularity along with its related areas such as cloud computing, machine learning, and big data will transform information society. However, the advent of big data, despite its benefits, presents a long-term challenge of how to load the vast amount of data generated. The long-term accumulation of streaming data or video data (such as Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (MR)) with IoT hardware devices (KRISTIANI et al., 2019) could be addressed by the sharing of a larger platform with IoT platform as one form of solution.

Cloud Computing is of vital importance to businesses and individuals and is serving its purpose well (ZHANG; RAVISHANKAR, 2019). However, the cloud data centers are far away, and the distance that data need to traverse the Internet to end-users is also relatively long. Here arises the problem of latency (Zhang et al., 2018). Therefore numerous proposals have been established, such as (Gouareb; Friderikos; Aghvami, 2018), that aim to mitigate the effect of structural latency through optimization techniques that the conventional service implementation model imposes.

Latency can be acceptable for some applications that can tolerate delays; however applications such as Virtual Reality, Real-time Processing, and Augmented Reality will function poorly or even fail when faced with latency issues (Shi et al., 2016).

Furthermore, the Fifth Generation Mobile Networks (5G) presents another pressing need to address the latency challenges that Cloud Computing faces. Therefore Edge Computing is presented as a solution to resolving the latency challenges. Edge computing can achieve this by bringing processing closer to end-users at the Edge of the network, and if the internet connection fails, the processing continues (HU et al., 2015).

IoT scenarios rely on sensors, computing devices, and gateways deployed on the edge, near the user. The next generation of services and applications will use the concept of Networking Function Virtualization (NFV) (LI; CHEN, 2015) where different components will be deployed across the infrastructure on the cloud and also on the edge (Parvez et al., 2018).

1.1 Motivation

The use of virtual machine virtualization on the edge can be considered a relevant step to address some issues that container virtualization may not be able to guarantee.

Therefore a first case in point is the NFV European Telecommunications Standards Institute (ETSI) Architecture that uses a single entity, called Management and Orchestration (MANO). MANO must be able to communicate with the VIM and be able to deploy VNFs across the whole infrastructure.

Furthermore considering that Virtual Network Function (VNF) can also run on Virtual Machines. One requirement of NFV is the capacity to deploy Virtual Machine (VM) not only on the core of the network but everywhere and also in the edge.

Therefore considering that on the edge can have single board computers with low computing capacity, such as Raspberry Pi, the main motivation of the work is to verify the ability of this type of hardware to allow VMs running of them.

Despite that virtual machines are consider as heavy and resource consuming compare to containers but there is still a need for them base on this reference (TAO QI XIA; LI, 2019). Here the author noted that the use of VMs are considered better than container in some situations such as when the user can not afford to have a compromised host and any event that can result in the host crashing. As anyone of these situations can result in a high cost for the user. Consequently, with such a possibility existing a VM is considered the best option because when a container crashes or has been compromised, the containers in problem may affect the entire host machine.

Therefore this and other issues serve as an important motivation to conduct an investigation of the capacity of the Raspberry pi to handle computer virtualization based on VMs. Despite that current works exclusively focus on the use of containers on SBC edge devices such as the Raspberry Pi (RPi), it is necessary to provide and option to container virtualization on SBC devices on the edge. This is so especially for high cost operations that can not risk host compromise in the event of container failure and its ripple effect on the entire system.

1.2 Objectives and Research Challenges

The main goal of this study is to investigate the capacity of the SBC namely Raspberry Pi Version 3 (RPi3) and Raspberry Pi version 4 (RPi4) to handle computing virtualization based on Virtual Machine using an experimental approach. A few secondary objectives are also set:

- ❏ To investigate the state-of-art regarding computing virtualization based on virtual using SBCs.

- ❏ To research and verify the operating system and software components to enable Virtual Machine on Raspberry Pi versions 3 and 4.
- ❏ To use real SBC to perform several tests using well-known metrics such as boot time, CPU, memory and swap disk usage to verify hardware capacity and compare the results.

1.3 Hypothesis

The infrastructure on edge based on SBCs such as RPi3, and RPi4 can support compute virtualization based on VMs and can enable the deployment of VNFs in this type of hardware in a seamless way that it is done in bare metal servers.

1.4 Contributions

I will be seeking to provide an implementation approach of MEC that uses low cost raspberry pi as compute node on the edge. Moreover my contribution to the process of enabling virtualization at the edge of the network for future service provision includes the following: Enabling multiple virtual machines to be launch and subsequently be used at the edge of the network. Provide an alternative to the available options currently being used at the edge of the network on resource constraint devices such as the raspberry pi. Demonstrate the feasibility and possibility of utilizing RPi at the edge as a low-cost and low energy consumption device. Therefore contributing to the rapid development of Mobile Edge Computing (MEC) and implementation of 5G network. Our work, along with related research, could serve as a motivation for other researchers and investors to get involved in the development process of Edge Computing, which is a facilitator for the implementation of MEC and 5G computing.

1.5 Research Method

Creation of Reference Base - The research will be carried out on the main topics, with focus on the implementation of edge computing using raspberry pi 3 and to a lesser extent raspberry pi 4. Virtualization will be configured on the Rpis for the implementation of openstack compute to be used with virtual machines.

To find out the capability of the RPi3 to support traditional VMs, our experiments look at standard system resources usage parameters such as CPU, RAM, and disk swap usage. Also, we investigate the processor temperature.

The percentage of CPU performance is one such metric that was collected for analysis. Memory response was also necessary as it provides relevant information regarding the amount and types of programs that the system will be able to accommodate. The use of swap was also necessary to buffer RAM after all the system memory has been allocated.

The temperature, which is also related and relevant to the CPU performance, was also recorded for analysis. Finally, the virtual machines boot times were collected to compare the boot time when only one virtual machine is running as opposed to progressively running multiple virtual machines simultaneously.

During the experiments, we used the *psutil* tool (RODOLA, 2016) to collect the data about resource usage on the RPis. We executed each test 15 times to avoid statistical bias and calculated average values.

1.6 General Scheme of Dissertation

The remaining of this works is organized as follows: Chapter 2 presents an overview of different concepts and technologies in the context of this research and also presents the state-of-the-art in this area. Chapter 3 presents the rationale, the choices and solutions regarding this work. Chapter 4 shows the experimental evaluation and describes the testbed, the evaluation approach used and a discussion about the results. Finally, in Chapter 5, details the main contributions of this work and in addition points out some suggestions for the evolution of the work in future investigations.

Chapter 2

Background and Literature Review

This chapter presents fundamental concepts that are in the context of the work, such as SBCs, compute virtualization techniques, and NFV main concepts. Furthermore, the chapter presents a related work section where the literature about this area is described and analyzed.

2.1 Openstack

OpenStack is defined a set of software tools for building and managing cloud computing platforms for public and private clouds as well as it is a collection of open source software projects which provides an Infrastructure-as-a-Service (IaaS) solution through a set of interrelated services. It provides series of interrelated projects delivering various components for a cloud infrastructure solution as well as controls large pools of storage, compute and networking resources throughout a datacenter that are all managed through a dashboard known as Horizon. It gives administrators control while empowering their users to provision resources through a web interface. Openstack was founded by NASA and Rackspace Hosting and have rapidly grown to be a global software community of developers collaborating on a standard and massively scalable open-source cloud operating system. It was launched in July 2010 by NASA and Rackspace (KUMAR et al., 2014).

2.2 Single Board Computers

According to (JOHNSTON MIHAELA APETROAIE-CRISTEA; COX, 2016), SBC is considered as a small, complete computer consisting of sockets for external display, network and expansion capabilities via USB and PCB headers for external circuit boards or sensors.

In our research the main focus will be on the ARM architecture which according to Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation is becoming increasingly widespread due primarily to its low-

Table 1 – Comparing SBCs CPU, RAM and Cost

SBCs	CPU	Memory	Cost(US|$)
Raspberry Pi2 B	Quad Core @900MHz ARMv7	1GB	35
Raspberry Pi3 B	Quad Core @1.2GHz ARMv8	1GB	35
Odroid C1+	Quad Core @1.5GHz ARMv7	1 GB	37
Odroid C2	Quad Core @2GHz ARMv8	2GB	40
Odroid XU4	Quad Core @2/1.4GHz ARMv7	2GB	74
BeagleBone	ARM Cortex A8	512MB	45
Alix 3D2	AMD LX800	256MB	103

Table 2 – SBCs Storage Type and Cost

SBCs	Storage	Cost(US$)
ODROID-C1+(S805)	MicroSD Card Slot/eMMC module socket	35
Raspberry Pi3 B+	MicroSD Card Slot 4GB/eMMC on-board flash storage	35
Beaglebone Black	SPI Flash onboard, MicroSD, 8-bit SDIO interface	50
Udoo Neo	Quad Core @2/1.4GHz ARMv7	50-65
Odroid XU4	MicroSD, eMMC	74

power characteristics, low cost, and its use in smartphones, tablets, and other devices. It was further noted that however despite these great feature along with the modern ARM multi-core processors they are able to compete with general purpose CPUs. Furthermore some versions of the ARM processors are 32-bit with the use of more powerful 64-bit devices now growing. Although the focus will be on ARM SBC devices, we will briefly look on AMD based SBC.

The Table 1 was created from two tables from both article (MORABITO, 2017) and article (KAUP1 STEFAN HACKER2, 2018) as they both compare SBCs. It shows the comparison of popular SBCs features that can guide in the an SBCs selection process. The cost of the devices use to create the needed test-bed or infrastructure is one of the challenges affecting cloud and edge computing deployment. Also very crucial is the type of CPU the device uses. Another very important feature is the available memory, which will directly affect virtualization and the type and amount of application that can be deployed on these systems. In order to select the best hardware for usage in implementing the test-bed Table 2 is added to present more devices options to consider along with the features each device offers.

Table 3 – Raspberry-Pi4 vs Pi3

Features	**Raspberry Pi 4**	**Raspberry Pi 3 B+**
SoC	Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	Cortex-A53 (ARMv8) 64-bi @ 1.4GHz
GPU	Broadcom VideoCore VI	Broadcom Videocore-IV
RAM	1 GB, 2 GB, or 4 GB LPDDR4 SDRAM	1 GB LPDDR2 SDRAM
Display and audio port	2 micro-HDMI 2.0, 3.5mm analogue audio video jack	Full size HDMI 3.5 mm analogue audio video jack
USB	2x USB 3.0 + 2x USB 2.0	4x USB 2.0
Ethernet	Native Gigabit Ethernet	300 Mbps Giga Ethernet
Video Decoder	H.265 4Kp60, H.264 1080p60	H.264 and MPEG-4 1080p30
Power Supply	5V via USB type-C up to 3A and GPIO header up to 3A	5V via micro USB up to2.5A and GPIO header up to 3A
Expansion	40-pin GPIO header	40-pin GPIO header
Wifi	2,4 GHz and 5 GHz 802,11b g n ac wireless LAN	2,4 GHz and 5 GHz 802,11b g n ac wireless LAN
Storage	microSD card	microSD card
Bluetooth	Bluetooth 5,0 BLE	Bluetooth 4,2 BLE

For raspberry Pi4 vs Pi3 there are architectural differences between Rpi3 and Rpi4 as shown in the table above; where Rpi3 is generally build with 1 GB ram for all its models whereas the Rpi4 have a 1 GB ram model, a 2 GB ram model and a 4 GB ram model. The raspberry pi 4 with 4 GB ram will be used in the experiment to complete tests and their results will be used to do a comparison between the Rpi 4 and Rpi 3.

Power Source Options for the Raspberry Pi

The need for a source of energy is paramount for electronic devices to function therefore we will explore the possible options that are available to power the raspberry pi for our research.

There are four categories of power sources that can be applicable to IoT devices though not all maybe efficient or feasible for use with the raspberry pi. The four categories of power sources are as follows:

Wall Power

The wall power is great for stationary applications. It has good stability and the cost for small scale applications is generally affordable. It is also relatively easy to maintain once installed however it has poor mobility.

Batteries

Batteries are the main source of power for applications that need to be mobile. Therefore it is most suitable for mobility, but maintenance can be costly. However an alternative to regular batteries is the use of rechargeable batteries and solar panels.

Energy Harvesting

Another source of powering the raspberry pi that can be consider is Energy harvesting. There are various forms of Energy harvesting, however we can focus on solar energy harvesting. It is not consider to be very stable because it works only when there is light. And even if there is light, the output power varies a lot depending on the light intensity and depending on how you consume the power. So it's not very stable. And solar cells and solar panels are bulky. It is untethered, but mobility is limited by the availability of light. They don't really need any maintenance except for removing accumulated dust(KIM, 2020).

No-power Devices

Finally the other source of power we will analyze is called the No-power devices. The No-power devices will be examined in the use of the of RFID tags, they are extremely unstable because they work only when there's an RFID reader nearby but they are extremely small, cheap, and easy to maintain. Therefore based on the four possible options the most applicable ones for the raspberry pi are wall power, batteries and energy harvest such as solar. According to (KIM, 2020) none of the four sources of energy is best in every aspect. However for our research we will be using wall power or electrical outlet.

In order to give a clear view of the relevance of single board computers its use will be assess from the perspective of cloud computing to edge computing and IoT. An advantage of cloud services is that it assists companies and individuals to reduce the costs of investing in data centers or servers, and its users can flexibly control the use of computing resources (PARK SEULGI KIM, 2018). However as explain by (CAPROLU ROBERTO DI PIETRO, 2019) that cloud computing should not be considered as a one-size-fits-all solution because it tend to central resources. However its centralized approach normally results in an increased separation between user devices and their clouds this in turn will result in significant network latency and jitter. The resulted latency and jitter will affect delay sensitive applications, such as gaming, augmented reality and e-health require low latency and low or no jitter. Here is where edge computing is introduced to address these issues by moving computing and storage away from centralized points. According to (TAO QI XIA; LI, 2019) edge computing is proposed as an extension of cloud computing where it provides hardware sources at the edge of the Internet, it serves client devices with much lower network latency than cloud computing, therefore greatly improving the user experience for delay-sensitive client-remote applications. Furthermore it is now considered as an enabling technology of IoT.

Therefore the next step is to determine how best to implement edge computing infrastructure since the cost for the devices can be very discouraging. In (MATTHEWS, 2018) research they found that the normally cloud systems are implemented with devices such as laptops, portable desktop computers or by means of satellite communication with remote high performance computing (HPC) clusters. They refer to its usage in a battlefield environment and highlight the challenges that were face such as the power consumption and cooling requirements of larger systems can be troublesome, especially in harsh climates. The analysis further mentions other environmental challenges such as dust, high temperatures, and fluctuations in power. In their continued assessment they noted that latency and security requirements can result in delay in the analysis of the information needed for effective command, control and intelligence. The need for single board computers can now be effectively assess. The use of single board computer offer several advantages. Firstly it is a fact that their small size and the fact that they are extremely inexpensiveness compares to laptops or servers and enable high versatility. Secondly, their use of flash storage enables fast access to data without the latency or power consumption of spinning disk storage (MATTHEWS, 2018). Finally it is noted that their System-on-a-Chip (SoC) processors enable data storage capacities and processing capabilities that far surpass microcontrollers and Field Programmable Gate Arrays (FPGAs). Also their ease of reprogramming, and their use of a wide array of ports enable them to be used as standalone computers or mounted on other devices for a variety of applications. Therefore it is more cost effective to implement on the edge and offer a wide array of benefits in terms of mitigating the challenges of using normal computers and servers.

2.3 Compute Virtualization

Virtualization is the process of abstracting the operating system, application, storage and network away from the original hardware or software.

This section describe different compute virtualization techniques.

2.3.1 VM Based Virtualization

A Virtual Machine (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.

Full virtualization is where Virtual machine are created to simulate hardware and allow an unmodified guest OS to be run in isolation. It has two type of Full virtualization namely Software assisted full virtualization and Hardware-assisted full virtualization.

The Software Assisted Full Virtualization is also considered as Binary Translation (BT). It uses binary translation to trap and virtualize the execution of sensitive, non-virtualizable instructions sets. It also allows the emulation of the hardware using the software instruction sets. Types of software assisted (BT) are:

- VMware workstation (32Bit guests);
- Virtual PC;
- VirtualBox (32-bit guests);
- VMware Server

The Hardware-Assisted Full Virtualization (VT) The use of the hardware-assisted full virtualization eliminates the binary translation and it directly interrupts with hardware using the virtualization technology which has been integrated on X86 processors.

The Hardware-assisted – Full virtualization hypervisor type 1 (Bare metal) are as follow:

- VMware ESXi /ESX;
- KVM;
- Hyper-V;
- Xen;

The other category of virtualization is Paravirtualization which works differently from the full virtualization. This type of virtualization It does not need to simulate the hardware for the virtual machines. The Virtual guests are aware that they has been virtualized, unlike the full virtualization (where the guest doesn't know that it has been virtualized)

Table 4 – Full Virtualization, Para-Virtualization, and Hardware-Assisted Virtualization

Parameter	**Full Virtualization**	**Para Virtualization**	**Hardware Assisted Virtualization**
Generation	1st	2nd	3rd
Performance	Good	Better in certain cases	Fair
Used By	VMware, Microsoft, KVM	VMware, Xen	VMware, Xen, Microsoft, Parallels
Guest OS modification	Unmodified	Codified to issue hypercalls	Unmodified
Guest OS hypervisor independent?	Yes	XenLinux runs only on Hypervisor	Yes
Technique	Direct execution	Hypercalls	Exit to root mode on privileged instruction
Compatibility	Excellent	Poor	Excellent

to take advantage of the functions. A major difference between full virtualization and paravirtualization is that in the full virtualization, guests will issue hardware calls but in the paravirtualization, guests will directly communicate with the host (hypervisor) using the drivers.

The types of paravirtualizations are Xen, IBM LPAR, Oracle VM for SPARC(LDOM) and Oracle VM for X86 (OVM).

The other type of virtualization to be considered is Hybrid Virtualization which is the Hardware Virtualized with PV Drivers Here the guest operating systems are unmodified and utilized many VM traps and high CPU overheads which limit the scalability. The hybrid paravirtualizationcan be considered as the combination of Full and Paravirtualization. If there is a bottleneck with full virtualization, especially with I/O and memory intense workloads the virtual machine will use paravirtualization for specific hardware drivers. The types are 1.Oracle VM for x86 2.Xen 3.VMware ESXi. Finally the OS level Virtualization is also known as “containerization”. containerization is where the Host Operating system kernel allows multiple user spaces known as instances. Its usage allow very little or no overhead as its uses the host operating system kernel for execution. Type of os level virtualization are 1. Oracle Solaris zone 2.Linux Lxc 3.Docker 4.AIX WPAR.

The Table 4 shows the difference between Full Virtualization, Para-Virtualization and Hardware-Assisted Virtualization and allow for an informed decision on which is best to use for the application of virtualization method.

Virtualization with KVM and Qemu

Virtualization solutions using hypervisors has been the most widely used for implementing virtualization and achieving isolation. The hypervisors operate at the hardware level, therefore supporting standalone virtual machines that are independent and isolated of the host system. As a result of this isolation of the VM from the underlying host system, it is possible to run a totally different operating system on top of another such as Linux-based vms on Windows and Windows-based VMs on top of Linux (MORABITO; KOMU, 2015). The type of hypervisors are: Type-1: native or bare-metal hypervisors which operate on top of the host's hardware Type-2: hosted hypervisors which operate on top of the host's operating system.

Linux's Kernel-based Virtual Machine (KVM) has characteristics of both type 1 and type 2 hypervisor. It is a virtualization solution for the Linux Kernel. It is used in conjunction with a hardware emulator and virtualizer such as QEMU (MORABITO; KOMU, 2015).

2.3.2 Container Based Virtualization

Linux Containers (CANONICAL, 2020) are based on Kernel Virtualization It has been noted (Full Virtualization vs Para Virtualization vs Hardware-assisted Virtualization) that Several container-based solutions exist such as LinuxVServer, OpenVZ, LXC, Docker,and Rocket. However only lxc, lxd and docker will be assess. LXC is an OS-level virtualization technique. It does not require full isolation of different OSs. Instead, it allows all containers to share an OS kernel. Each container has a virtualized kernel for itself, while the underlying system has only one copy of the kernel (TAO QI XIA; LI, 2019). A newer for is LXD which is a daemon that facilitates a REST API that is used for the management of containers. It runs on top of the LXC and provides a simple command line user experience for easy management of containers that allow it to reduce the complexity (AULIYA; WULANDARI, 2019). However docker is a higher-level platform that combines Linux containers with an overlay filesystem and provides tools for building and packaging applications into portable environments (MORABITO; KOMU, 2015).

Exploring the use of virtual machines on the edge is an interesting option that we have choose despite the argument against its overhead as compare to the use of containers. Their use on the edge can be justify base on the fact that because VMs achieve hardware-level isolation while containers only achieve operating system (OS)-level isolation a grave problem can arise as a result of this. Base on (TAO QI XIA; LI, 2019) research a container can crash or be compromised and as a result the containers in problem may affect the entire host machine containers.

2.4 Networking Function Virtualization (NFV)

2.4.1 Main Concepts

The virtualization solution used is layered and maintains features close to the European Telecommunications Standards Institute (ETSI) NFV framework. Considering a bottom-up approach, we describe the Network Functions Virtualization Infrastructure (NFVI), the hardware management entity, as a biform infrastructure layer, so the virtualization engine is alike for low-cost, high-performance servers. Thus, it becomes possible to offer Infrastructure as a Service (IaaS) in which the service is uniformly deployed over both hardware using the same manifest file (descriptor). Thereby, we make the ability to deploy services across multiple domains hybrid.

2.4.2 Management and Orchestration

Also, next to the NFVI layer comprises the Virtual Infrastructure Manager (VIM), the layer that controls and manages the infrastructure entities.

On top of the bottom layer, we propose an intermediate tier that comprises the service deployment domains. It contains the Core and Edge blocks, which are respectively infrastructures that host services traditionally on high-performance hardware and low-cost infrastructures that can be deployed and enabling verticals of services such as IoT, Smart farms, and others. MANO compliant solutions could be placed here to handle the service management and orchestration of the middle tier. State-of-the-art solutions deal with these deployment domains separately.

The topmost layer that makes up the framework of the proposed solution comprises the multi-form applications that we refer to as Virtualized Everything Functions (VxFs), according to (SILVA et al., 2019). Our proposal as a uniform virtualization layer for both bare-metal and low-cost devices democratizes the deployment domain, so the plethora of applications are broader than state-of-the-art solutions. Also, testbeds of applications like (SILVA et al., 2019), (SALLENT et al., 2012), and (Silva et al., 2018) can add to their facilities our infrastructure model that can add functionality as service descriptor manifest files.

Virtualization is essential for Edge computing provisioning and enables in most use-cases for new kinds of applications (PAOLINO et al., 2015). Therefore, any device that will be used on the Edge must be able to accommodate virtualization even if it is operating system level, such as Linux Containers (LXC) or any other container technology.

However, the use of a virtual machine is needed for some applications in cases where better security, isolation, and network performance (Marques et al., 2018) are preferred, but on most small, low-cost devices enabling or using virtualization is more difficult, especially if one wants to use virtual machines (Barik et al., 2016). The use of virtual machines on the RPi is relatively new and is only enabled through the use of particular operating systems such as OpenSuse, Ubuntu, and Gentoo.

In order to build an Edge-Core virtualization platform to host VxFs spanning in different use-case domains, we have separated the architecture into two verticals to make it loosely coupled. The state-of-the-art solutions deal with VxF deployment specifically for each domain, and each virtualization technology: low-cost and high-performance virtualization requires specific managers (Li et al., 2017). For instance, OpenNebula (Edge release) (Milojičić; Llorente; Montero, 2011) deploy services on low-cost hardware; however, uniformity on it does not prevail. That is, services as conceived as LXC on low-cost device services, and for bare-metal follows standard virtualization.

Unlike other Edge virtualization proposals, our solution enables VNF life-cycle management smoothly. VIM does not need to handle low-cost compute infrastructure or bare-metal separately. Somewhat we adapted a uniform virtualization layer for both types of hardware. Thus, the service deployment on top of the low-cost computing does not necessarily have to be container-based once those solutions have network connectivity drawbacks.

2.5 Related Work

Edge computing research and development have reached a high point of interest. It will be the facilitator of 5G and MEC as it seeks to bring processing closer to the user at the Edge of the network and, in so doing, help to resolve the latency problem. However, there still exist hurdles to get over, and the literature describes several works related to this aim.

The authors of MEC-ConPaaS (van Kempen et al., 2017) project stated that deploying a realistic mobile Edge cloud remains a challenge because mobile operators have not yet implemented MEC technologies in production, and this makes it impossible for researchers to use their techniques in actual mobile phone networks.

They also noted that there exist very few open-source platforms that may support experimentation that emulate a mobile Edge cloud, and they recognize the most mature one to be OpenStack++ (van Kempen et al., 2017).

However, MEC-ConPaas researchers claimed that the general OpenStack implementation relies on classical server machines for its deployment. Their proposal is to deploy a experimental MEC testbed that relies on single-board computers such as Raspberry Pis (RPis) and similar devices. Their work relies on ConPaaS (Pierre; Stratan, 2012), which is an open-source run-time management system for elastic applications in public and private cloud environments. They utilized RPi with Linux Containers (LXC) enabled and OpenStack orchestrating the containers.

Therefore they proposed a more natural way of deploying an experimental MEC testbed that relies on single-board computers such as Raspberry Pis (RPis) and similar devices. Their work relies on ConPaaS (Pierre; Stratan, 2012), which is an open-source run-time management system for elastic applications in public and private cloud environments.

MEC-ConPaaS presented an interesting foundation and proof of the concept of using the raspberry pi at the edge, which was further highlighted by (Pahl et al., 2016) who did similar research with ConPaas and the implementation of LXC with three hundred RPis in a cluster.

Also, they explored the various forms of a Platform-as-a-Service (PaaS) for edge computing. However, these relevant foundation researches may be impacted because of the shift from the use of LXC to newer technologies such as Kubernetes and LXD, along with security concerns that it presents. Therefore a newer container technology called LXD based on LXC was presented by (Ahmad; Alowibdi; Ilyas, 2017) who proposed the use of RPi2 Model B and LXD Linux containers.

(KRISTIANI et al., 2019) implements Edge Computing using OpenStack and Kubernetes. Their implementation uses three layers, such as the Cloud side, Edge side, and Device side. The cloud side mainly deals with more complicated operations and data backup. Its main function is to deploy the Kubernetes cluster on an OpenStack platform. Also, the cloud side contains the Virtual Machine (VM) and Kubernetes master side, and it also deals with data backup, complex operations, data visualization, and other applications that do not need quick responses.

A more sophisticated form of edge implementation from the previously mention containers was introduced by (Chang et al., 2014), which applied a computational model called the Edge Cloud. The application here brings the cloud to the edge and uses it as a means of enabling a new type of hybrid application called Edge Apps, which runs over Openstack (Chang et al., 2014).

Another edge app implementation was presented by (SCHILLER et al., 2018), who creates mobile edge apps that are managed as virtual network functions. Their platform also includes an SDN controller to manage traffic by using the control plane to derive states for traffic management. They research Vehicular Fog Computing for Video Crowd Sourcing, where they analyze the availability of vehicular fog nodes based on a real-world traffic dataset. Then explore the serviceability of vehicular fog nodes by evaluating the networking performance of fog-enabled video crowdsourcing over two mainstream access technologies, DSRC and LTE (ZHU et al., 2018).

(BELLAVISTA et al., 2018) implement a solution that uses powerful and low-cost middleboxes deployed at the edges of the network. It serves as enablers for their Human-driven Edge Computing (HEC) as a new model to ease the provisioning and to extend the coverage of traditional MEC solutions. Their approach uses different devices from the RPi but is also relevant as middleboxes allow specialized services to be used at the edge of the network. However, some of these services can be enabled on the RPi to reduce cost of implementation.

(TONG; LI; GAO, 2016) apply hierarchical architecture for the edge cloud to enable aggregation of the peak loads across different tiers of cloud servers. They use this as a means of maximizing the number of mobile workloads being served. The need for load balancing is essential and can also be implemented on the RPi which have multipurpose capabilities which are similar in some aspect to the PC.

Another load balancing technique is proposed by (PUTHAL et al., 2018) whose technique is used to authenticate the EDCs and to discover less loaded Edge Data Centers (EDC) for task allocation. They state that it is more efficient than other existing approaches in finding less loaded EDC for task allocation and that it strengthens security by authenticating the destination EDCs.

In the work (LEI et al., 2018), the authors' solution allows mobile Edge applications to be provided by chaining the service functions with the assistance of Edge computing techniques and virtualized resources. They also build a testbed to evaluate their Edge caching architecture for proof of concept and tested it with typical caching scenarios. The research is not base on the RPi; however, the chaining of function can be tested on the RPi.

In conclusion of the review of related work, the use of the RPi with other devices is assessed in (AKRIVOPOULOS et al., 2018) who uses RPIs and Zotac and also an off-the-shelf Intel-based edge-based server box. The software that they deployed on the devices is bundled using a collection of Docker containers. They use these to create an IoT-based platform for real-time monitoring and management of educational buildings on a national scale. They design the system to process sensor data on the Edge devices of the network.

Unlike the presented related works there are some noted differences with the virtualization that we used on the edge and the one of the devices we used on the edge. Our use of a RPi4 on the edge for comparison with the use of a RPi3 on the edge. The use of virtual machines on the raspberry pi on the edge was not reflected in any of the related works that were published. However in our research we explore the prospect of using virtual machines on sbcs on the edge on the RPi 3 and RPi4 to assess the performance of the virtual machines on each devices.

Table 5 – Comparison of Related Works

The Previous Solutions	**Type of Virtualization**	**Edge device**	**Metrics**
(van Kempen et al., 2017)	Lxc	Rpi3	storage, CPU, memory, network, power consumption
(Ahmad; Alowibdi; Ilyas, 2017)	Lxd	RPi2	Launch time
(KRISTIANI et al., 2019)	Kubernetes, dockers	Rpi3	Boot time
(Chang et al., 2014)	Lxc	sensors, laptops, computers	Edge App
(SCHILLER et al., 2018)	Lxc,kvm	cloud server	latency,SDN,VNF
(BELLAVISTA et al., 2018)	VM, dockers	middleboxes	Connectivity, Stability, Load Balancing
(PUTHAL et al., 2018)	Edge datacenters (EDCs)	wireless sensor nodes and smart devices	algorithm
(LEI et al., 2018)	VMs	Intel-computer,radio	SFC in edge caching
(AKRIVOPOULOS et al., 2018)	Docker containers	RPIs, Zotac, VPS	Network-bandwidth, Average-processing, CPU
(ILYAS MUNEEB AHMAD, 2020)	lxd	RPi3	CPU,RAM,I/O,launch time, Zram
(HAJJI, 2016)	Docker	Rpi2	CPU,RAM,Network, Energy
(SHARMA LUCAS CHAUFOURNIER, 2016)	Docker,Kvm vms	Intel,Dell computers	CPU, memory, disk, network
(ZHANG LING LIU; ZHOU, 2018)	Docker, kvm	Intel computers	Ram,Cpu,Boot-time, Scalability

Chapter 3

Virtual Machine on the Edge

To enable the use of NFV it is mandatory to have the capacity of management and orchestration of Virtual Network Functions (VNFs) across the infrastructure.

The new type of services, such as the ones envisaged by 5G that requires Ultra-Reliable and Low-Latency Communications (URLLC) capacity from the network usually also requires computing hardware available on the core and also on the edge of the network. In these usage scenarios where you have different types of computing hardware such as bare metal servers on the core and low cost single board devices on the edge.

In this type of service, the VNFs can be distributed in the these types of computing hardware and it is necessary that the MANO entity can communicate with the VIM and manage the resources in a seamless way in all this hardware.

This chapter presents the rationale necessary to have VNFs distributed on bare metal servers and Raspberry Pi devices. This chapter also describes a solution to enable the creation of Virtual Machines (VMs) in low cost devices, in this case, Raspberry Pi devices.

3.1 Virtual Functions on the Edge

The goal of creating a solution that uses virtual machines on raspberry pi is to add to the options of virtualization on the edge devices as currently most research and test beds use mostly lxc , lxd or Kubernetes on the edge devices. However a traditional cloud environment mostly uses virtual machines on their platforms and some researchers and developers may rather use what they are familiar with in some instance such as virtual machines.

This solution will serve as another option to the available edge virtualization approaches and could possible inspire a combination of other approaches with virtual machine usage on the edge. It is also an inspiration for further research in the uses of traditional virtual machines on the edge or a combination of virtual machines and containers. Furthermore it will be an additional enabler for the implementation of 5G technology with will use edge computing also.

3.2 Architectural Design

To handle these drawbacks we bring a uniform solution as described in Figure 1. Our proposal enables $x86\_64$ virtualization compatibility on top of low-cost compute resources as it is straightforward on bare-metal compute resources. The boot process, which starts on Nova-API (release 13.1.4), receives the requests to a new VxF creation. Afterward, the messages go through the RabbitMQ mechanism to the Nova-Scheduler, which decides which *compute-node* contained in the cluster (Cellv2) will handle the request. Ultimately, the remaining build-block components are Keystone 9.3.0, Glance 12.0.0, and Neutron 8.4.0.

The filter (*ComputeCapabilitiesFilter*) in Nova-Scheduler checks if the deployment request fits adequately on the low-cost *compute-node*, according to VxF flavor, being possible the nova-boot process will create and instantiate the VxF. If a low-cost *compute-node* is unable to handle the deployment request, the VxF will be launched on top of bare-metal compute resources.

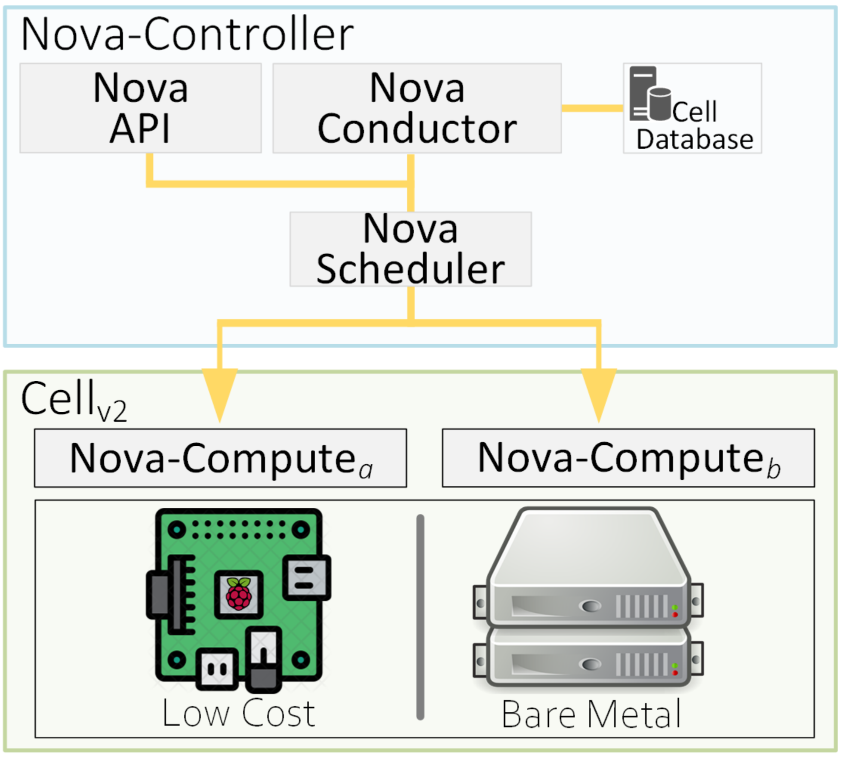


Figure 1 – Proposed Solution Build-Blocks.

Commonly RPi has only one physical network interface, and the basic OpenStack deployment requires two interfaces, we proposed an approach based on Open vSwitch (OvS). Therefore, we created two virtual interfaces *vnet0* and *vnet1*, which we assign different addressing plane. Both interfaces were associated with a *OvS*, which had as trunk port the only physical interface of RPi. The RPi physical interface connects directly to a physical switch where the controller-node and the network gateway are also connected.

Thus, one virtual interface served as a provider interface for the Neutron plugin, working as L2 Agent, and the other allowed the compute-node to exchange messages with the controller-node services such as AMQP and databases.

Our solution for enabling Edge computing involves the use of the small and low-cost RPi running Ubuntu 16 *armhf*-capable. However, its limited memory, in comparison to the bare-metal poses restrictions on the programs that can be installed and executed on it. Therefore, since we are installing OpenStack on the RPi, only the Nova-Compute can be installed on it and not the Nova-API or Nova-Cert as these freezes the RPi as soon as they start running. However, we build it with Nova-Compute and Neutron for the network. Therefore, the RPi memory capacity was able to allow these to run smoothly.

As long as OpenStack requires virtualization, we enable such technology for the Edge with this approach. To this end, we implement OpenStack Nova-Compute to utilize virtual machine by connecting a localized OpenStack controller to our RPi OpenStack *compute-node*. The RPi configuration enables the booting, stopping, accessing, and deleting virtual machines from the RPi.

Therefore our solution can place all these added functionalities at the Edge of the network closer to the user. The use of low-cost devices will also benefit the users and businesses, along with the benefit of low latency will be achieved when the services and functionality are placed at the Edge of the network closer to the user.

Our proposal investigates use of standard virtual machines on low-cost hardware. The goal is to use the same virtual machine that usually runs on high-performance hardware in order to verify the suitability and compatibility of the use of a single MANO entity, in our case OSM. Container-based solutions are known (LEON. et al., 2018) on these scenarios; however, the suitability of a uniform proposal for deploying VxFs according to the ETSI framework was not discussed in previous studies.

3.3 Virtual Machines on Raspberry Pi

The use of virtual machines on the raspberry pi have presented a range of challenges and issues that needed to be resolved to enable the smooth execution of virtual machines. Firstly the configuration and enabling of virtualization capabilities on SBCs especially arm based SBCs is very challenging since it require a different approach than the traditional computers approach. In the case of traditional computers they normally depend on the use of Intel VT and AMD-V that more specifically the raspberry pi which does not use Intel VT and AMD-V for hardware virtualization enabling. Therefore the first major challenge will be the need to obtain or compile a kernel with KVM userspace enabled along with the required packages and needed configurations that will allow virtual machines to be created and executed without errors or premature failures. Therefore to solve this problem raspberry pi operating systems that have the potential to allow the enabling of KVM will be obtained and configured to enable KVM and the kernel recompiled if necessary. Next the relevant packages needed to prepare the operating system to run virtual machines were prepared, installed, configured and tested for functionality and efficiency. This procedure was done for a number of operating systems and packages to see which allow the optimal performance of the virtual machines.

The following figure illustrates the overview of the underlining virtualization configuration that is needed for the raspberry pi to be capable of using KVM virtualization. The Kernel-based Virtual Machine (KVM) component is an open source Linux kernel module. It has a loadable kernel module namely kvm.ko, that provides the core virtualization infrastructure. It also consisit of a processor specific module such as kvm-intel.ko or kvm-amd.ko. The KVM module at first was only functioning on the x86 hardware platforms that contains the virtualization extensions Intel VT and AMD-V.

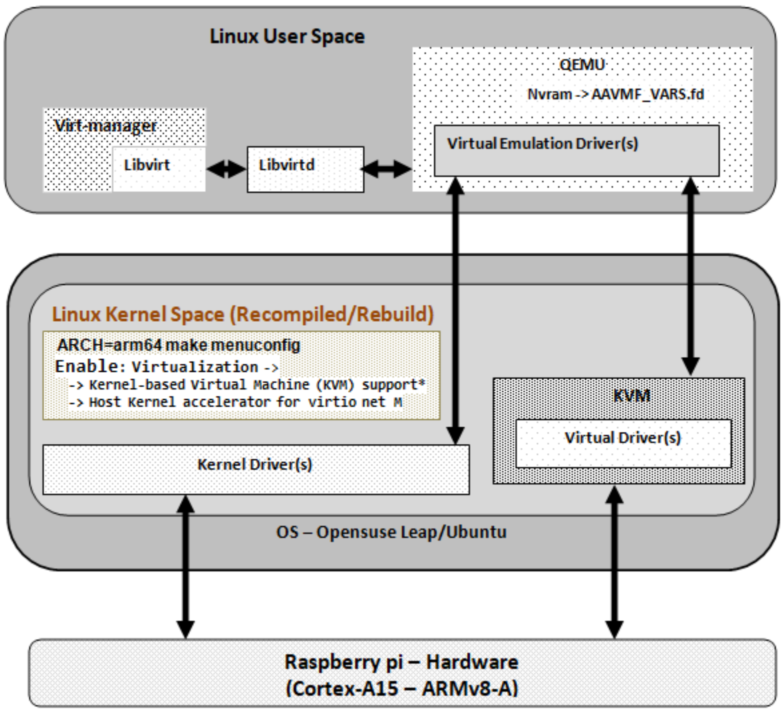

Linux User Space
QEMU
Nvram -> AAVMF_VARS.fd
Virt-manager
Libvirt
Libvirtd
Virtual Emulation Driver(s)
Linux Kernel Space (Recompiled/Rebuild)
ARCH=arm64 make menuconfig
Enable: Virtualization ->
-> Kernel-based Virtual Machine (KVM) support*
-> Host Kernel accelerator for virtio net M
KVM
Virtual Driver(s)
Kernel Driver(s)
OS – Opensuse Leap/Ubuntu
Raspberry pi – Hardware
(Cortex-A15 – ARMv8-A)

Figure 2 – Enabled KVM Virtualization

Virtualization it is now reconfigured to function on the raspberry pi without the need for the virtualization extensions that Intel and AMD computers provide to enable virtualization. The KVM enabled in the kernel is the first and very important stage to install it for usage however because it is a kernel module and resides in the Linux kernel space it can not be used alone to create the virtual machines. It will need the help of another component to make this possible. A program must be present in the Linux user space to interact with KVM in order to provision and manage virtual machines. The main program used here is Quick EMUlator (QEMU). It is a generic, open-source, standalone, software-based, full system emulator and virtualizer that is needed by KVM but it do not need KVM to function on its own however it would perform poorly and slowly if it is not integrated with KVM. Finally, the next important component for the virtualization functionality on the raspberry pi is libvirt. According to (BHOSALE, 2016) it is an abstraction layer library for various hypervisor management APIs written in C. Also it is able to manage a set of virtual machines across different hypervisors.

In addition the operating systems build for raspberry pi generally do not enable virtualization in the operating system kernel for KVM to be used immediately after installing KVM and other virtualization packages. The user or someone other than the original provider of the operating system would normally have to be the one to reconfigure the kernel of the operating system, to enable the virtualization features and then recompile the operating system in order to enable kvm functionality. However customization of the operating system is required now mostly for RPi 3 as some RPi 4 operating system are now being prebuild with KVM virtualization features enabled. Furthermore, even with this feature enable the user will still need to install and configure other relevant packages in order to use KVM with either QEMU or libvirt for creating and executing virtual machines. A final note on the requirements of the operating systems to use KVM. It is adviced that KVM only works on a 64-bit operating system and the 32 bit versions of the operating systems for the Raspberry Pi can not be configured to use KVM. The operating systems that can be use with KVM enabled are ubuntu 18.04, ubuntu 19.04, ubuntu 20.04, Opensuse leap 42,43 and 15, Gentoo and Fedora 33 is the latest to be KVM enabled operating systems. However Ubuntu 16.04 armhf was also used to experiment with nova-compute.

The experiment process involved a lot of trial and error to arrive at the best operating systems and combination of software that would allow efficient and consist creation of virtual machines that are reliable and will function predictable. In the first instance the gentoo kvm configured operating system with qemu installed was tested. Here a gentoo RPi 64 bit image ship with kvm enabled in the kernel was used, other necessary software for creating the virtual machine were a BIOS tianocore aarch64 EFI specific called QEMU-EFI.fd, an iso9660 image that was created with the cloud-init files such as meta-data and user-data. Then these files are included in an instruction to boot the vm with qemu-system-aarch64 to boot a bionic-server-cloudimg-arm64.img virtual machine. However using this method on gentoo works but prove to be less stable for consistent booting of a series of virtual machine on the same host as there was a point that it fail to start the virtual machine that was running perfectly after it was first created. Furthermore more than one virtual machine did not seem possible or trivial to run on the same host.

The next operating system used was ubuntu 18.04 with kvm enabled in the kernel. The supporting software that were installed were qemu-kvm, libvirt-bin, virtinst, virt-manager, bridge-utils, qemu-system-arm and qemu-efi with qemu.conf edited to make nvram equal to AAVMF-VARS.fd. The graphical virtual machine manager was use to create virtual machines which were created efficiently however as the virtual machines creation increase on RPi 4 it was observed that they were unstable. Therefore the next operating system that was tested was openSUSE Leap 42.2 for Rpi3 but was unable to function on RPi4 therefore an equivalent openSUSE leap 15 was used for RPi4.

In order to setup and boot guest vms on openSUSE leap 42.2 the agraf qemu-patched will be needed but may not be needed for openSUSE leap 15 which is an updated versions for RPi4. Also packages such as git, gcc, gcc-c++, zlib-devel, gtk2-devel, libfdt1-devel and make will be needed to make qemu-systemu-aarch64, that will be needed to create virtual machines. Then configurations were done with modprobe and qemu-nbd on the arm64 image. The image will be used to create the virtual machine. The command qemu-nbd will be required to access the arm64 disk image contents in order to copy two files that are necessary for booting Ubuntu Virtual machine. The files are kernel vmlinuz-4.4.0-66-generic-lpae and initrd.img-4.4.0-66-generic-lpae whose series number will be base on the arm image used. The image used to create the virtual machine was xenial-server-cloudimg-arm64-disk1.img. The use of these files and the xenial-server-cloudimg-arm64 was able to create multiple stable virtual machines on both RPi3 and RPi4 which can be used with the same image or multiple copies of the images in different folders but with the required files. It is observed that stable virtual machines could be created until the limit of resources were achieved consistently. Therefore this was selected as the preferred method to use to create stable, efficient and maximum number of virtual machines on RPi3 and RPi4 in order to compare virtual machines function and use of resources on RPi3 and RPi4.

In a different test case with openstack nova a research was carried out with ubuntu 16.04 armhf operating system on the RPi host. A compute node with nova-compute was created where a connection was made and the raspberry pi was able to share the functions of issuing commands to create virtual machines however the scheduler was selecting the amd64 host to run the virtual machines. Here the RPi could start, stop and access the virtual machines but the desired results were to get the scheduler and openstack place the vms on arm therefore that area will be considered for future work.

Finally the use of virt-install could enable the ubuntu arm64 versions especially ubuntu 18.04 and above to create similar amount of virtual machines as using qemu-system-aarch64 on opensuse leap where qemu-system-aarch64 usage make it possible to create vms consistently while the host and the vms remain stable up to the maximum use of resources. However it can also allow more control over virtual machines than the use of qemu-system-aarch64.

Chapter 4

Experimental Evaluation

To evaluate our solution, we propose an experimental scenario in a real testbed environment. This section presents the testbed that realizes the proposed solution, the experiment and the method used for its evaluation.

4.1 Testbed Description

This section presents the tesbed description related to the RPi3 and RPi4.

4.1.1 RPi3 Testbed

The architecture of our experiment includes a RPi3 (Nova-Compute$_a$) and a bare-metal desktop computer (Nova-Compute$_b$) which are together in the network edge. The bare-metal desktop can run services deployed in edge cloud while the RPi3 is the edge computing near the user. Besides that, there is an OpenStack Controller, as presented in Figure 1.

The RPi3 is a Model B+ with a Broadcom BCM2837B0 system on a chip (SoC) with a Cortex-A53 (ARMv8) 64-bit 1.4 GHz processor and 1 gigabyte of LPDDR2 SDRAM. The bare-metal computer and Controller are Core I7 (i7-8550U) processors with 8 gigabytes of RAM. The RPi3 has Ubuntu 16.04 *armhf* server, customized for ARM processors, and publicly available. The software that was installed and configured on the RPi include neutron and nova-compute. However for the final tests and comparison of virtual machines on RPi3 and RPi4, opensuse leap 42.2 was installed on the RFPi3 host.

The bare-metal desktop computer uses a Ubuntu 16.04 64-bit (ARM64) server image. It also has the neutron and nova-compute OpenStack components (Mitaka release).

The controller computer uses a Ubuntu 14.04 64-bit (AMD64) server image and several OpenStack components such as Keystone, Glance, Nova, Placement, Neutron network, and Manila.

All the hardware was interconnected using a local physical network. A bridge network configuration supported the connection of the virtual machines created inside the RPi3, the local network.

4.1.2 RPi4 Testbed

A next testbed was created to test the performance of the raspberry pi 4 performance in similar areas as we did with the raspberry pi 3. Therefore a basic testbed was done just to mainly compare the performance of the raspberry pi 4 and raspberry pi 3 base on our research goals.

Here the raspberry pi 4 was configured with opensuse leap 15 operating system which has virtualization enable in the kernel. It is also a customized operating system for ARM processors, and is also publicly available. Then after kvm is enabled qemu was Download and build with a patched that makes it possible to run virtual machines with kvm.

The RPi4 is a Model B computer with a Broadcom BCM2711 system on a chip (SoC) with a Quad core Cortex-A72 (ARM v8) 64-bit 1.5 GHz processor and 4 gigabytes 0f LPDDR4-3200 SDRAM.

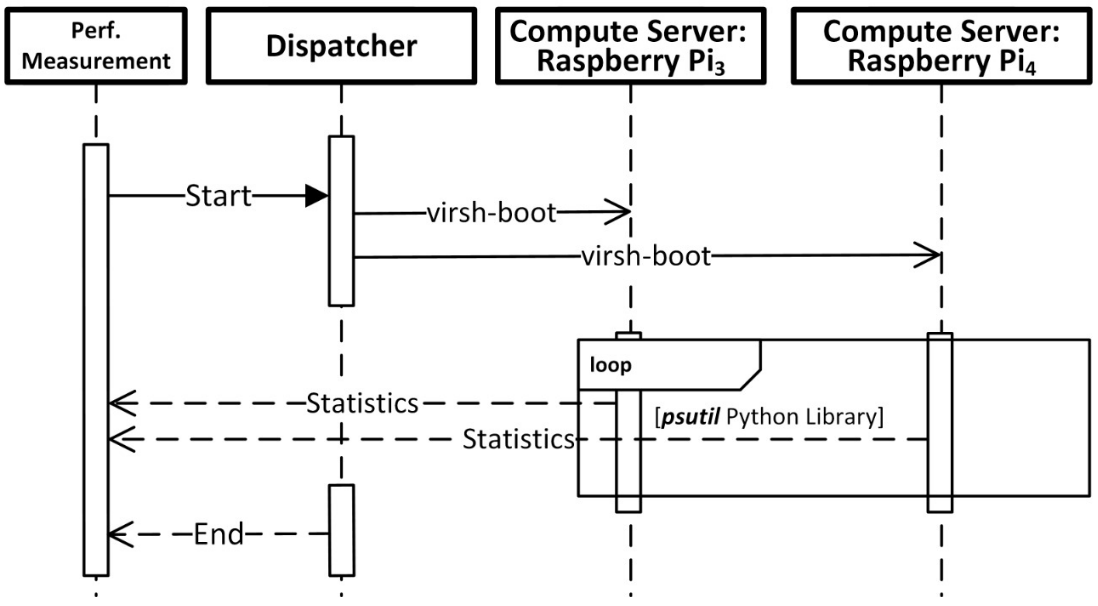


Figure 3 – Experimental Scenario Sequence Diagram.

The overall setup of the experiment is presented in Figure 3. A user starts a script to do the performance measurement. This script creates the VMs on the RPi version 3 and version located in the Edge, indicate as computer servers. All the statistics of the whole process are collected and stores for further analysis.

4.2 Evaluation Methodology

To find out the capability of low cost commodity devices to see how they will behave. In order to do that we choose common low costs devices such as the RPi3 and RPi4. Our experiments will evaluate their capabilities to support traditional VMs and further analyze standard system resources usage parameters such as CPU, RAM, and disk swap usage. Also, we investigate the processor temperature.

The percentage of CPU performance is one such metric that was collected for analysis. Memory response was also necessary as it provides relevant information regarding the amount and types of programs that the system will be able to accommodate as some programs and task require a lot of memory to function efficiently. The use of swap was also necessary to buffer RAM after all the system memory has been allocated.

The temperature, which is also related and relevant to the CPU performance, was also recorded for analyzed. Finally, the virtual machines boot times were collected to compare the boot time when only one virtual machine is running as opposed to progressively running multiple virtual machines simultaneously.

During the experiments, we used the *psutil* tool (RODOLA, 2016) to collect the data about resource usage on the RPis. We executed each test 15 times to avoid statistical bias and calculated average values.

4.3 Discussion

Using the experimental setup and following the evaluation methodology presented in Section 4.2 the solution presented in this work was evaluated.

Here we will present the results and a discussion about the RPi3 and also for the RPi4. Then there will be a comparative analysis about their performance in regards to the time it takes to boot virtual machines and the amount of virtual machines that can run simultaneous on the RPi3 as oppose to the RPi4. Furthermore there will the performance comparison will continue on cpu, disk and ram usage and finally temperature measure.

4.3.1 Discussion about RPi3

The following charts present the results of our experiments. We repeated the experiments to collect the boot times, fifteen rounds.

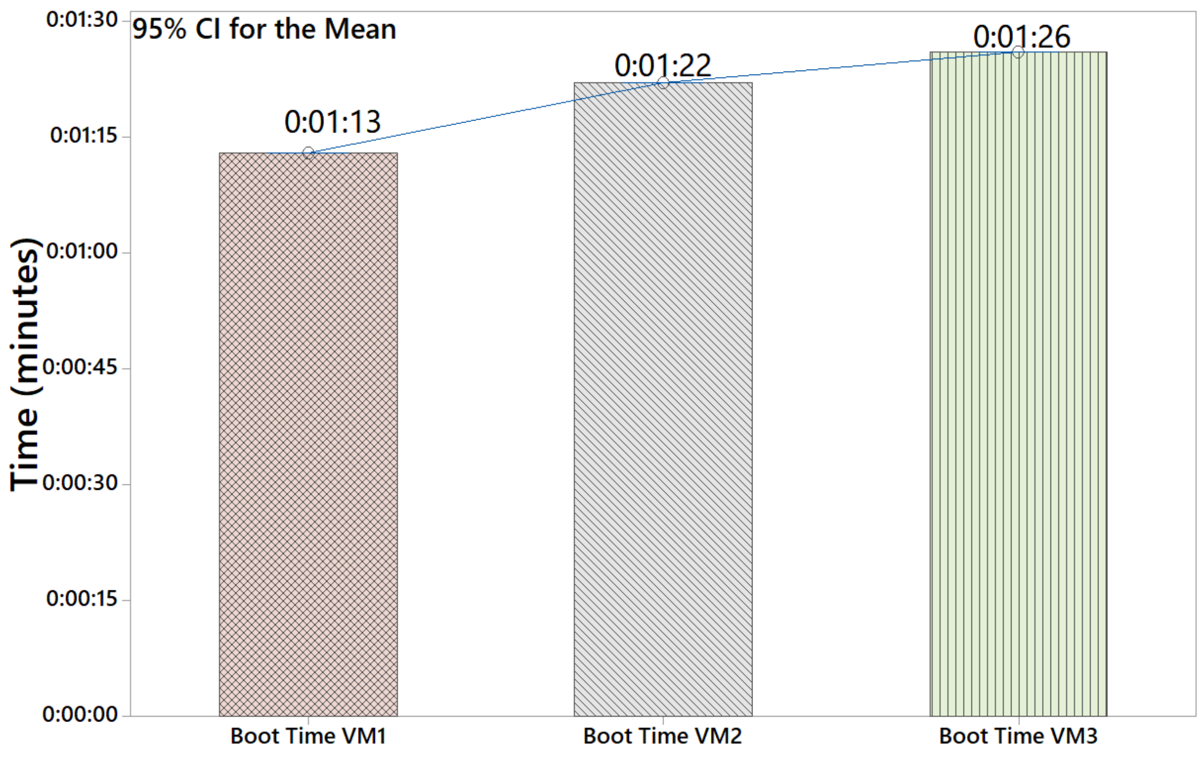


Figure 4 – RPi3 Instances Total Boot Times.

Figure 4 displays the boot times in minutes. The time present is the average time. In each round, we created three VMs since the RPi3 did not support more than this number of instances.

Also, we depict in Figure 5 the temperature rise per experiment round. As we can see, it is possible to notice the increase as the instances are launched. The average temperature, considering the 95% confidence interval, is ≈ 55.29°C.

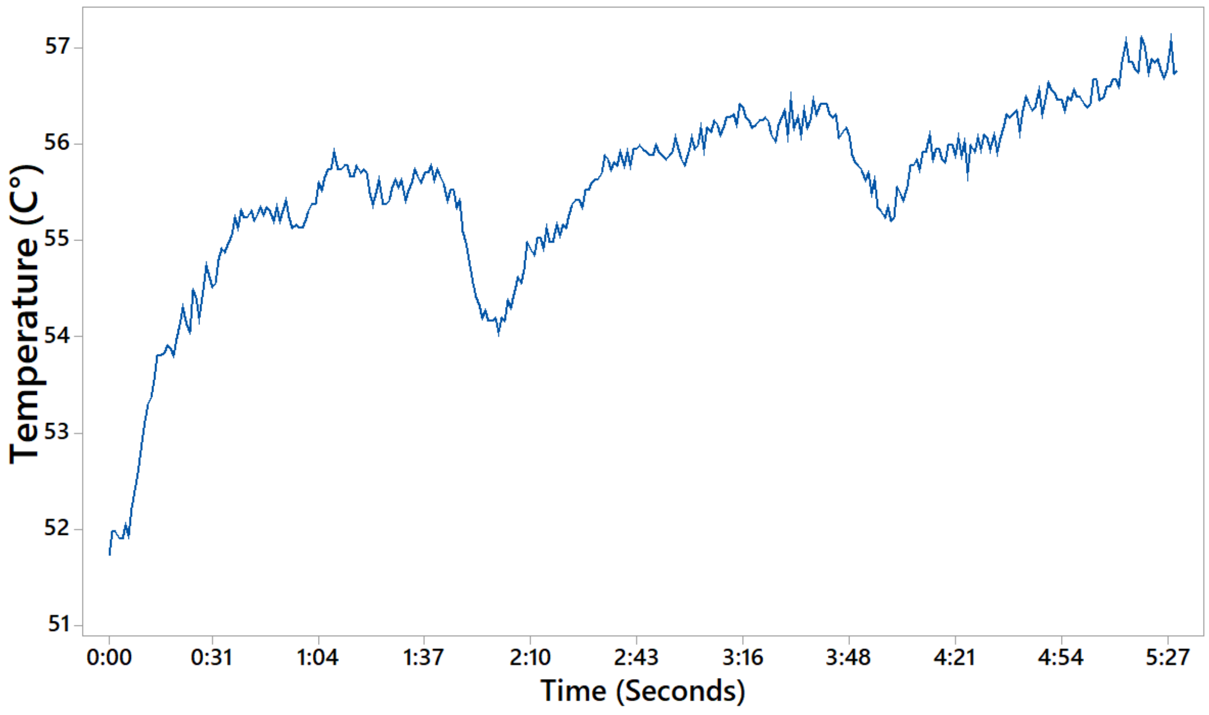


Figure 5 – Rpi3 Temperature Measurement.

Furthermore, the analysis of the temperature provided by Figure 5 shows that it rises during the process of booting each new virtual machine. However, after the booting process of each virtual machine, it falls significantly but not very low. Then risings steadily but sharply.

Figure 6 below presents the % of the usage of CPU, disk swap, and RAM combined in one chart to give an overall view of the system performance. This view makes it possible to compare and assess these related system performance metrics quickly. The graph shows the evolution of each of the above parameters during the experiment. The system resource usage starts the measurement at the beginning of the experiment. It continues getting this information while the three different VM instances are created. The values presented in the charts is the average of the fifteen rounds of the experiment.

The analysis of the CPU performance during the virtual machines booting process per rounds shows that at the start of the booting process of a virtual machine, the base value of the CPU is low then rises sharply and remains high while fluctuating. Then again at the start of the second virtual machine on the RPi, the CPU value falls sharply first then rises again sharply however the base of the low CPU value now shows an increase. This falling and rising behavior during the virtual machine booting process continues with the booting of the third virtual machine along with the low base value rising again significantly.

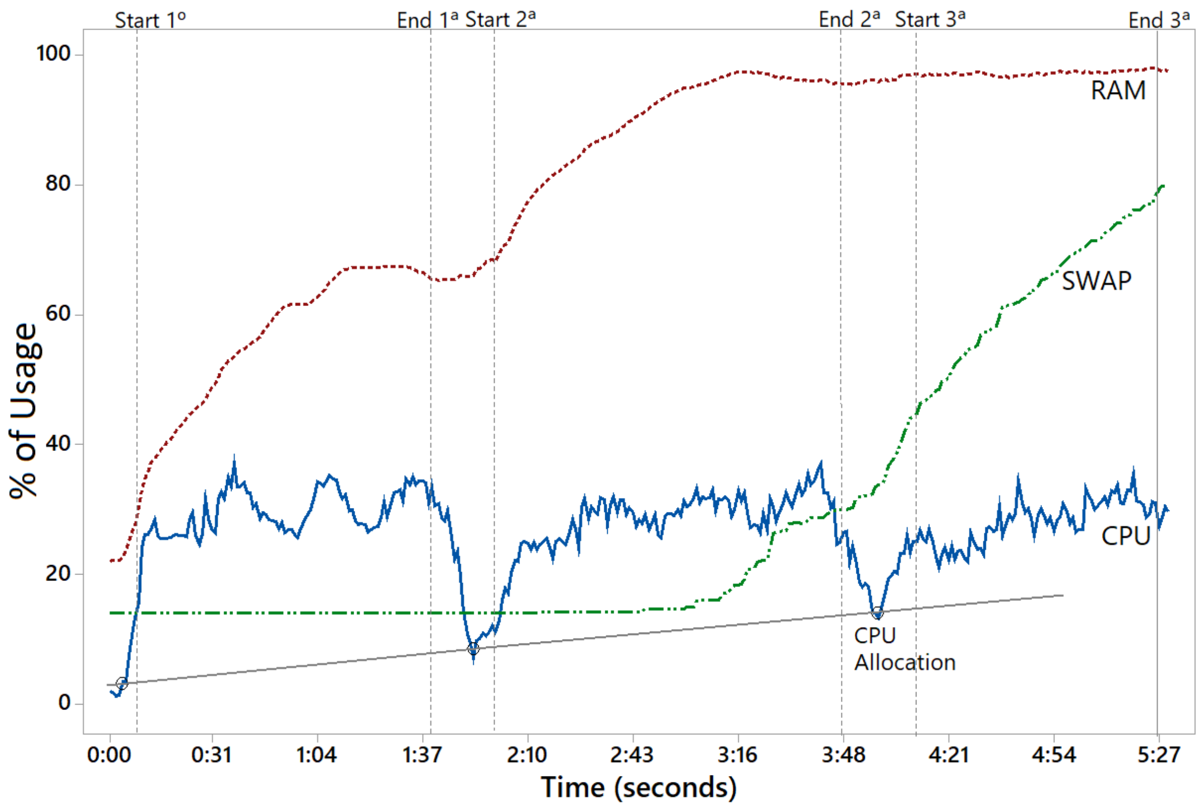


Figure 6 – RPi3 Resource Allocation over Time.

After the creation of a new virtual machine, the RAM allocation, and consumption increases. The memory capacity is the major restricting element in the number of virtual machines that the RPi3 can support. The RPi3 used in the experiment has one gigabyte of RAM. Considering that Ubuntu 16.04 *armhf* image requires 256 RAM per virtual machine, the current limit is three virtual machines.

However, when RAM consumption is near its limit, the swap memory is used as a buffer for processing to prevent the system from crashing. In Figure 6 swap can be observed to be steady before and after one virtual machine is created. Then starts to increase with the creation of the second virtual machine until it peaks with the creation of the third virtual machine.

It can be be seen that the CPU, RAM, swap, and temperature are connected to an extent, and all are influenced by the creating of the virtual machines, especially the last two. All the parameters show an increase during the last two boots, especially with the booting of the third virtual machine.

The MEC-ConPaaS project (van Kempen et al., 2017) states that they use LXC on the nova-compute, which is an RPi also, they state that their first LXC launch image test took 10 minutes to launch them with many configurations they were able to get it to 90 seconds. The boot time of an instance on our Edge infrastructure average less than 2 minutes, which is relatively close to the 90 seconds booting of instances in LXC, which are not as secure as virtual machines.

Finally, we were able to launch images from a MANO miles away. Here we connect the controller to OSM, and this allows us to also start virtual machines on the Edge with virtual network functions as needed or requested. Using this approach then makes it possible to monitor and manage virtual machines on the Edge much more straightforward, using standard platforms common used.

4.3.2 Discussion about RPi4

The Rpi4 CPU performance was similar in some aspect the Rpi3 as the base value of the CPU was low then rises sharply and remains high while fluctuating however the CPU usage peak at 55.6 percent which can be considered as a relatively low usage.

A more detail analysis of the performance of the virtual machines on the raspberry pi 4 will be assess below with related charts.

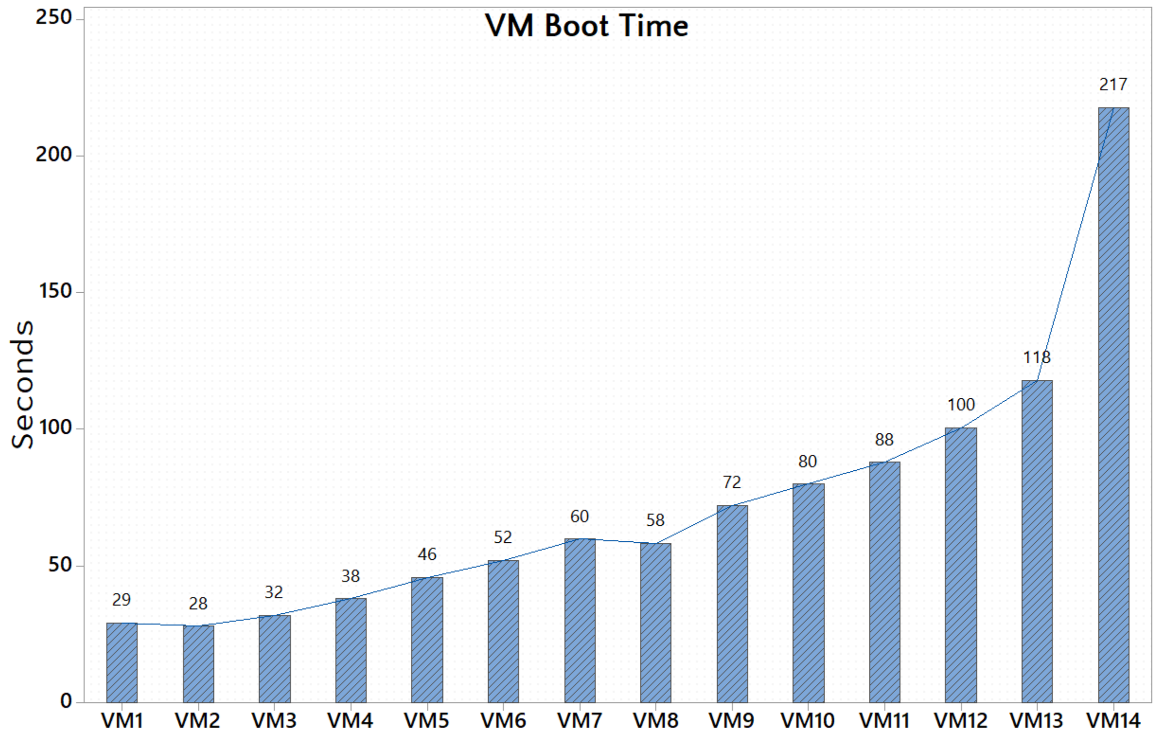


Figure 7 – Rpi4 Instances Boot Times.

Figure 7 displays the boot times of the virtual machines on the raspberry pi 4 in minutes. The times are presented on the chart as the average values of the boot time of each virtual machine over the boot rounds. In each round, fourteen virtual machines were create and boot in a consecutive manner on the raspberry pi 4. The decision to stop create virtual machines at the fourteen virtual machine was taken because it was the only inconsistent virtual machine. Inconsistent meaning its boot time can increase significantly sometimes; at times the cloud-init login and other components are failing during the boot process. However these problems as observed are base on internet load and ssh issues. These challenges lead to the decision to stop at the fourteen virtual machine creation.

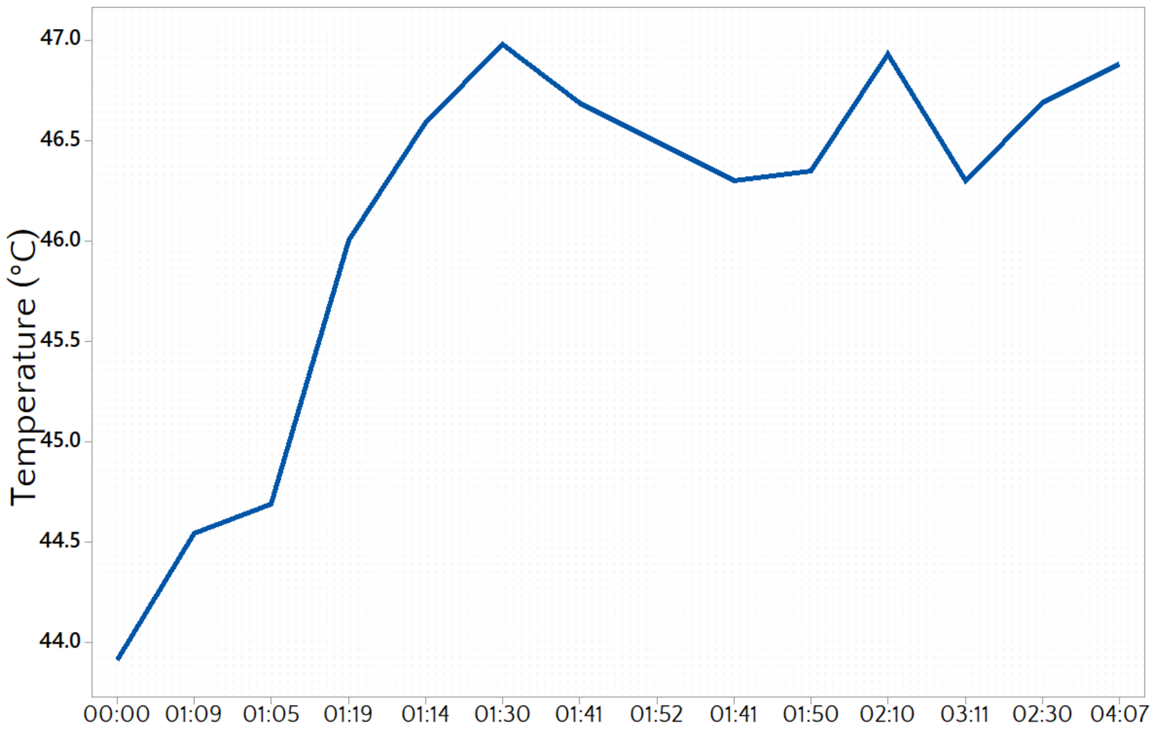


Figure 8 – Rpi4 Temperature Measurement

The temperature on raspberry pi 4 behaves almost similar to raspberry pi 3, the temperature behavior as shown in Figure 8 shows its continuous rising during the virtual machines booting tests on the RPI 4 as each virtual machine is created. It can be observed that after a sharp rise of about a quarter of the virtual machines the temperature falls significantly but not very low for an additional set of virtual machine creation. Then it risings sharply and fall sharply and finally rises again for the last additional created virtual machines.

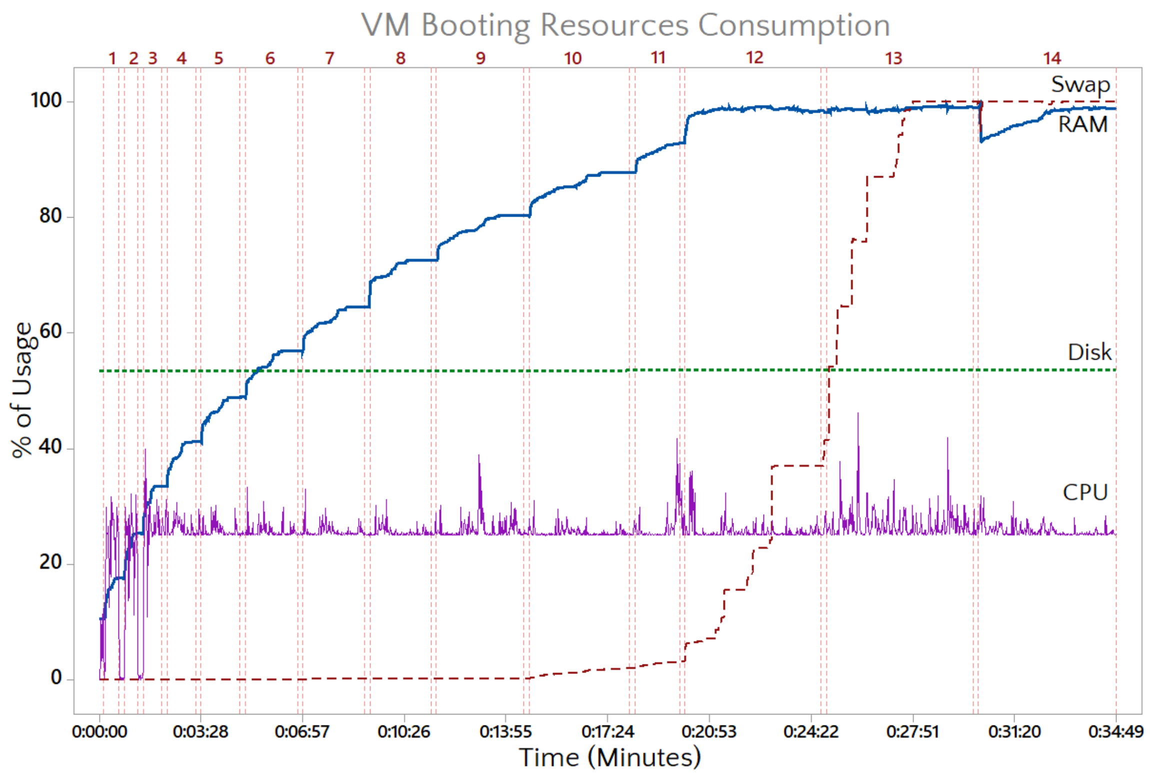


Figure 9 – Rpi4 Resources usage

The Figure 9 illustrates the usage and allocation of system resources progressively as virtual machines are created cumulatively.

After the creation of three virtual machines the raspberry pi 3 consumes 100 percent of ram then at this point the swap memory is used as a buffer for processing to prevent the system from crashing.

The disk usage remains the same from the beginning of the research through to the conclusion of the research. It is so because each virtual machine has its internal storage whose changes can only be assess from within the virtual machine. The next fairly reasonable used resources is the CPU whose usage increases during each boot process significantly and sharply as compare to non boot periods but not and seems not more than ten percent of non-boot periods shows small sharp rise occasionally. The ram memory which is arguable the most important resource in the determination of the number of virtual machine that can be created on the raspberry pi 4 reflect a constant demand and high usage increase as each virtual machine is created. It shows a gradually progressive increase going close to one hundred percent where it peaks at the creation of the 12 virtual machine. Then at one hundred percent with the booting of the fourteen virtual machine it take a sharp decrease but in the high usage area then increase again to one hundred percent usage of memory. The swap memory which complements the ram memory started with no allocation to a very small gradual allocation up to the creation of the 11 virtual machine. Then a noticeable sharp increase starts from the booting of the 12 virtual machine. After which it jump to a one hundred percent usage at the booting of the thirteenth virtual machine and stay steady at one hundred percent usage at the booting of the fourteenth virtual machine.

Here also the maximum usage of ram resources and swap helps to determine the maximum amount of virtual machines that can be created base on available resources especially the amount of ram that is available.

Additionally base on observation while conducting the experiments on the raspberry pi 3 and the raspberry pi 4 an observation is that most of the operating systems used on the raspberry pi 3 can not be installed on the raspberry pi 4. Therefore the significant changes to the Rpi4 result in the redundant of Ubuntu 16 as it along with most operating system built for raspberry pi 3 and 2 either fail to boot or hang during the booting process. However newer versions of Ubuntu function on the Rpi4 as the Ubuntu 18.04 and Ubuntu 19 were able to boot.

The experiments on raspberry pi 4 were done with Ubuntu-18.04.3 preinstalled server arm64 image was customized to meet the requirements of Rpi4 new firmware and included kvm virtualization enabled and opensuse leap 15 which runs ubuntu 16 more efficient. Finally, the virtual machines that were executed on the raspberry pi 4 were create with a bionic-server-cloudimg arm64 image and cirros-0.4.0-aarch64-disk images on Rpi 4 ubuntu-18.04.3. However after recognizing that opensuse leap 15 was more efficient and allow more virtual machines to be created the final tests were done by creating virtual machines with ubuntu 16.04 on opensuse leap 15 on raspberry pi 4.

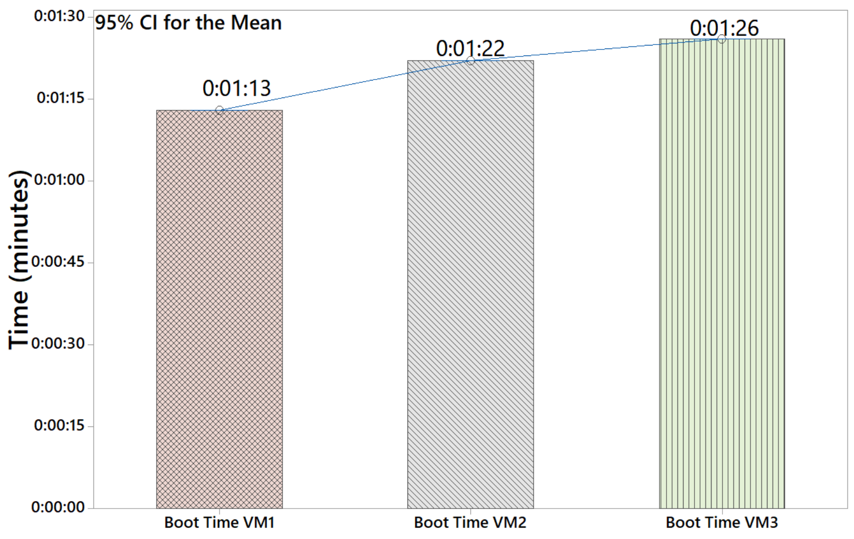
4.3.3 Discussion about RPi3 x RPi4

Here an analysis and comparison of virtual machines running on Rpi3 and Rpi4 will be conducted on boot time, number of virtual machines, ram and cpu metrics and other significant system metrics.

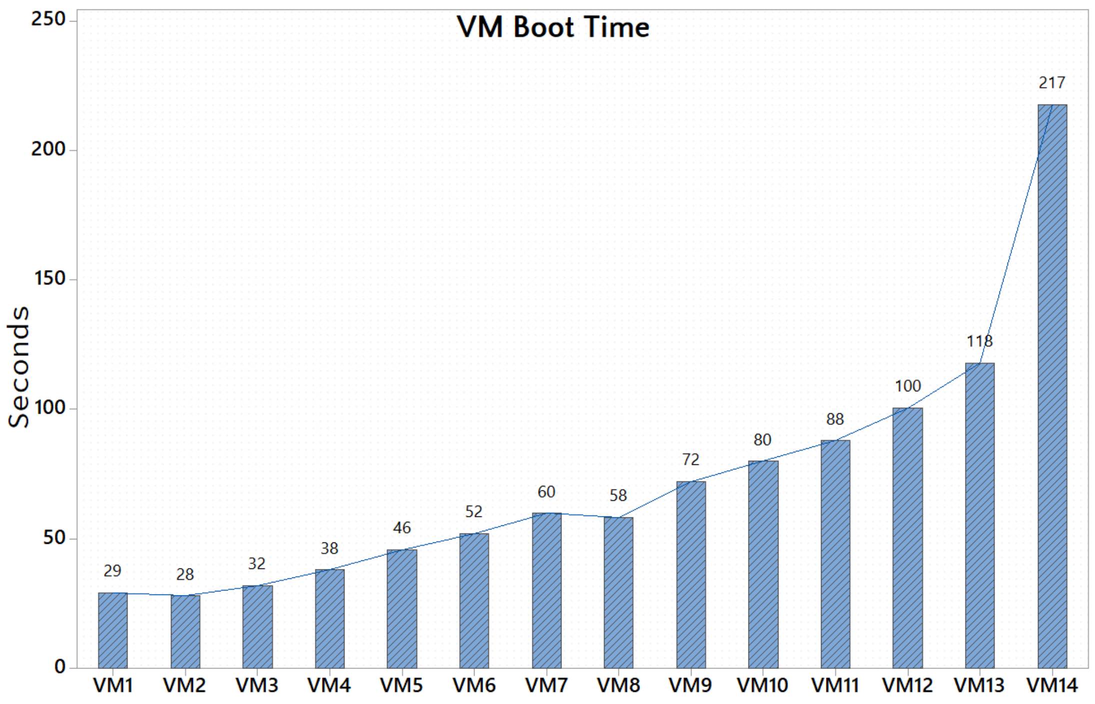
One of the most significant factors in the number of virtual machines that can run on the raspberry pi is the amount of ram that it is designed with as mention before that the memory capacity is the major restricting element in the number of virtual machines that the RPi3 can support. The RPi3 used in the experiment has 1 gigabyte of RAM. On the Rpi3 only 3 virtual machines were able to be created and executed. However on the Rpi4 with 4 GB ram 14 virtual machines with ubuntu 16 operating system were created and executed.

Comparison of RPi 3 and RPi 4 Boot times

The comparisons will be done base on Figure 3 which will be represented by figure 9(a) x Figure 6 which will be represented by figure 9(b).



((a)) RPi3 Instances Total Boot Times.



((b)) Rpi4 Instances Boot Times

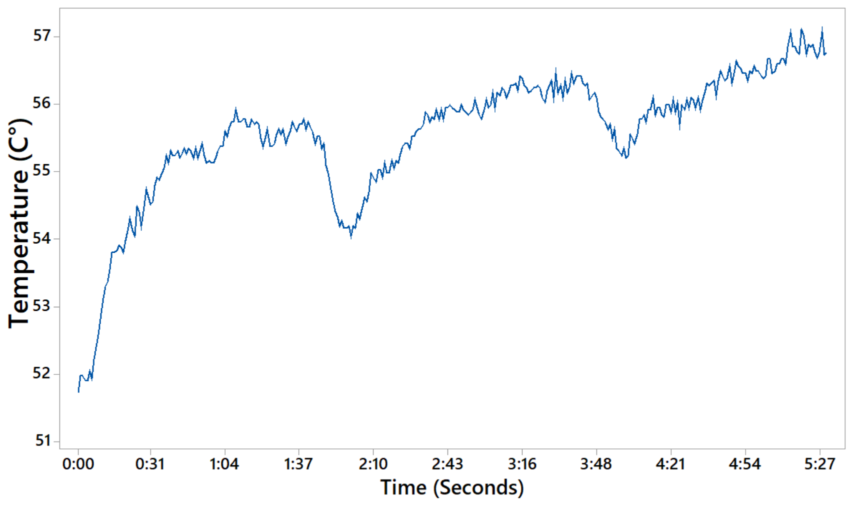
Figure 10 – Comparison of Instances Boot Times

After analyzing the virtual machines booting times, it can be seen from Figure 10(a) that the booting times of each new virtual machine continue to increase for each new virtual machine that is booted. Therefore after the first virtual machine is booted on the RPi3, the boot time for each subsequent virtual machine will increase; however, it will only increase by less than one minute.

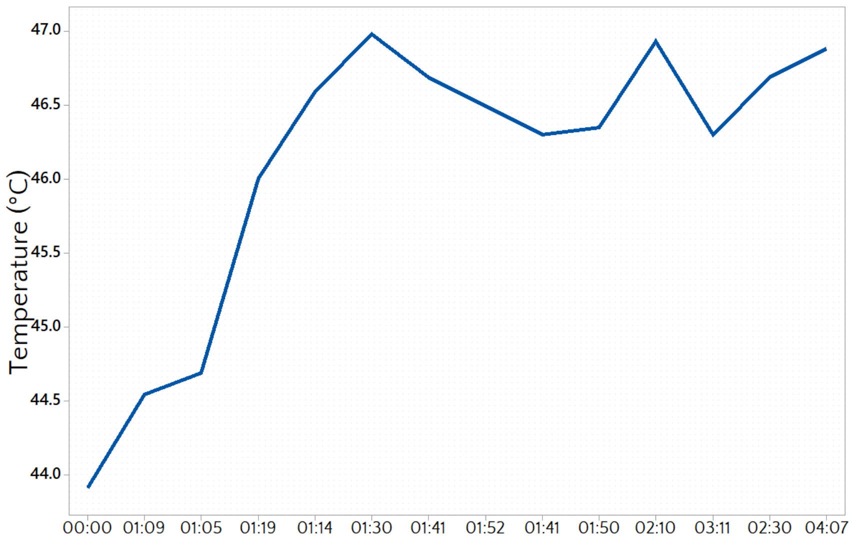
As illustrated on 10(b) the raspberry pi 4 virtual machines average boot times for the first 6 virtual machines are significantly lower than the boot times on raspberry pi 3. However similar to the raspberry pi 3, the average boot times on RPi 4 virtual machines increases gradually as another virtual machine is created. The average time for the fourteen and final virtual machine is drastically higher than all the previous virtual machines. However despite the final virtual machine high boot time average, the RPi 4 performance here is by far more superior in the amount of virtual machines that it allows to be created. Furthermore, another advantage with the RPi 4 is the increase in the average time it takes to create most of the 14 virtual machines.

Figure 4 x Figure 7

Comparison of RPi3 and RPi4 temperature Performance The comparisons will be done base on Figure 4 which will be represented by figure 10(a) x Figure 7 which will be represented by figure 10(b).



((a)) Rpi3 Temperature Measurement



((b)) Rpi4 Temperature Measurement

Figure 11 – Comparison of Temperature Usage

The analysis for the raspberry pi 3 temperature behavior provided by 11(a) display a rise in temperature on the device during the process of booting each new virtual machine. Then after each virtual machine finish booting the temperature tends to fall significantly though not very low. Then risings steadily but sharply. It was observed that the temperature on raspberry pi 4 behaves almost similar to raspberry pi 3, the temperature behavior as shown in 11(b) shows its continuous rising during the virtual machines booting tests on the RPI 4 as each virtual machine is created. It can be observed that after a sharp rise of about a quarter of the virtual machines the temperature falls significantly but not very low for an additional set of virtual machine creation. Then it risings sharply and fall sharply and finally rises again for the last additional created virtual machines.

Comparison of RPi3 and RPi4 CPU Usage and Performance

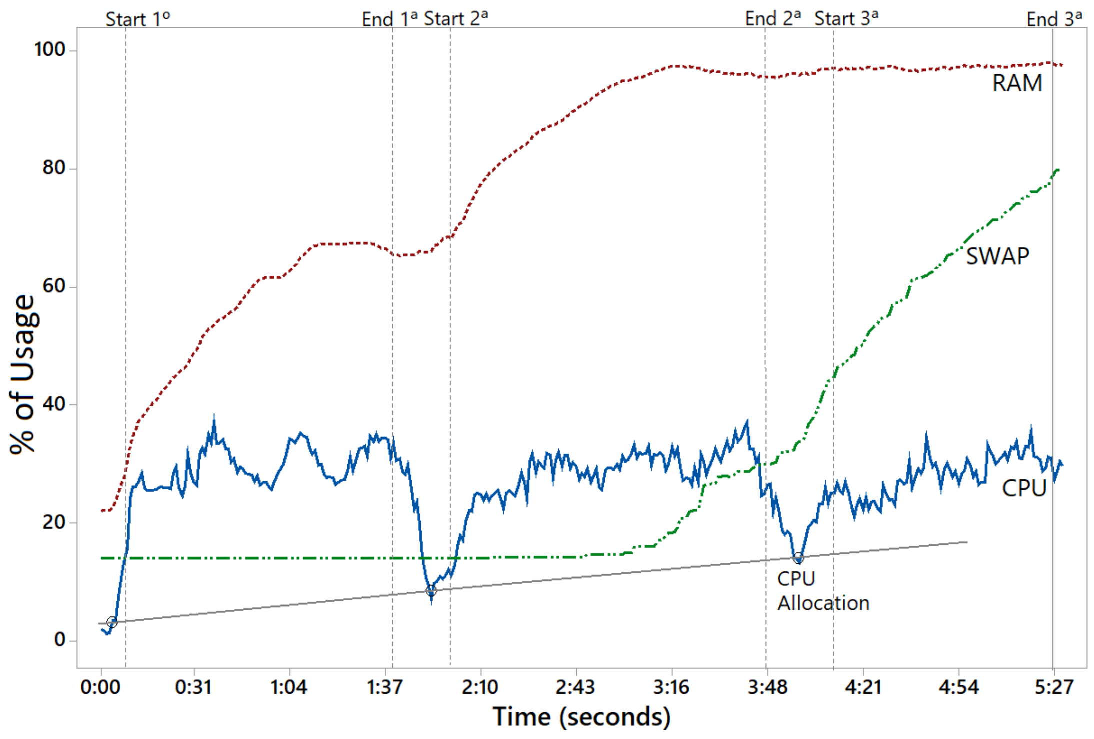
Figure 12 puts together the resource allocation over time for the RPi3 (see figure 12(a)) and for the RPi4 (see figure 12(b)).

In the comparison of 12(a)) and 12(b) the focus will be on the system resources usage and allocation of system resources. They will be assessed progressively as virtual machines are created cumulatively.

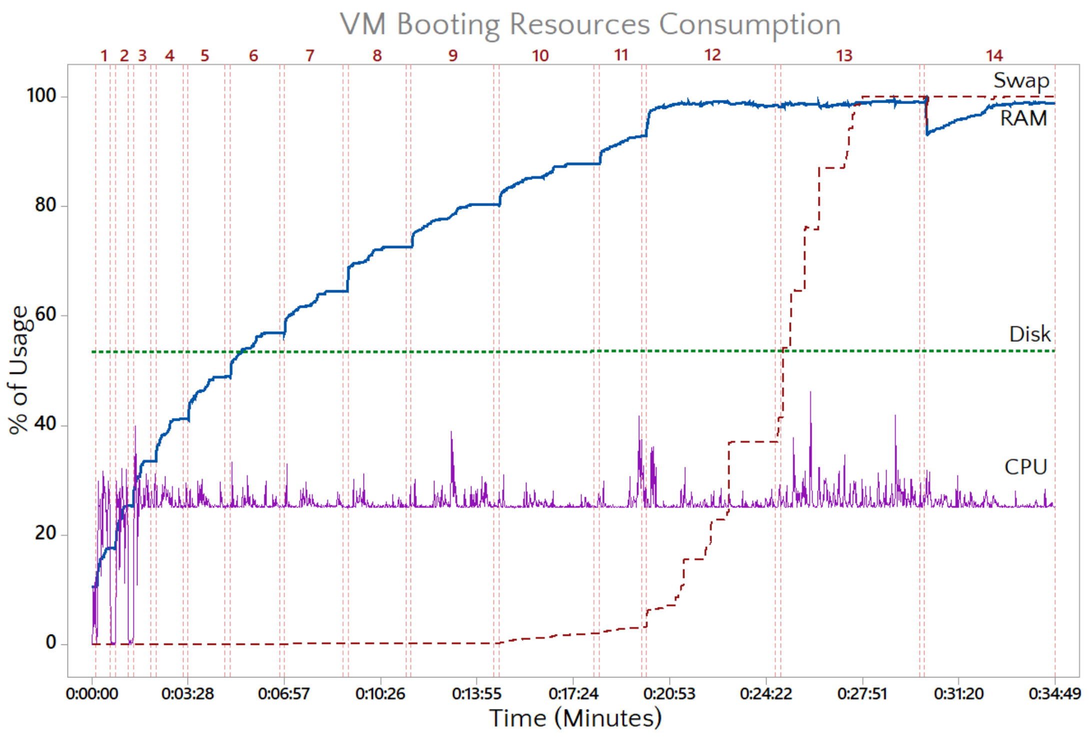
The 12(a)) shows the analysis of the CPU performance during the virtual machines booting process per rounds for the RPi 4. It shows that at the start of the booting process of a virtual machine, the base value of the CPU is low then rises sharply and remains high while fluctuating. Then again at the start of the second virtual machine on the RPi, the CPU value falls sharply first then rises again sharply however the base of the low CPU value now shows an increase. This falling and rising behavior during the virtual machine booting process continues with the booting of the third virtual machine along with the low base value rising again significantly. However for RPi 4, 12(b) shows the CPU utilization. The CPUs usage increases significantly during each boot process. Furthermore when a new virtual machine is created the CPU usage increases sharply as compare to non boot periods but with small sharp rise occasionally.

Comparison of RPi3 and RPi4 RAM Usage and Performance

RPi 3 12(a)) also present the findings on the use of the RAM. It can be seen that after booting each virtual machine, the RAM allocation, and consumption increases. The RAM memory will determine the total virtual machines that can be run simultaneously on the raspberry pi. On the RPi3 the minimal practical amount of memory to allocate to the virtual machines for efficient execution of each virtual machine was 256 RAM per virtual machines. On the RPi 3, 3 virtual machines with 256 RAM were executed gradually which result in the total memory usage of 768 of total memory of 1024 RAM. A fourth virtual machine creation at this time would result in the freezing of the system as the 1 GB ram is fully allocated. However on the RPI 4 with 4 GB ram more than triple the amount of virtual machines was created and function efficiently simultaneously. The test uses the same 256 RAM for each virtual machines. As each virtual machine is created with 256



((a)) RPi3 Resource Allocation over Time.



((b)) RPi4 Resource Allocation over Time.

Figure 12 – Comparison of Resources Usage

RAM, fourteen virtual machines were able to be created for a total of 3584 megabytes of RAM. At 14 virtual machines the boot time increases and some features of the virtual machine takes longer to stabilize therefore from these issues it can be concluded that 14 virtual machine is the maximum virtual machines that can be created on the RPi 4 efficiently. However 14 virtual machines is a lot more than the 3 that RPi 3 RAM allows.

Comparison of RPi3 and RPi4 Swap Memory Usage and Performance

Base on the analysis of the system resource usage for the raspberry pi 3, after the creation of three virtual machines the raspberry pi 3 consumes 100 percent of ram then at this point the swap memory is used as a buffer for processing to prevent the system from crashing. However for the raspberry pi 4 the swap memory steadily increases from the first virtual machine creation to the 11 virtual machine. After which it falls slightly then rises to 100 percentage usage.

Chapter 5

Conclusions

Within this work, we showcased a unique approach to managing and orchestrating resources on servers on the core and low-cost hardware on edge using the same MANO entity, in this case, represented by OSM. We also could explore the constraints associated with using the RPi on edge regarding standard parameters associated execution of virtual machines on these devices.

Previous approaches to deal with state-of-the-art OpenStack implementation on the edge such as OpenStack++ (HA; SATYANARAYANAN, 2015), and MEC-ConPaaS (van Kempen et al., 2017) were initiated to encourage faster Edge computing development with cloudlets. However, OpenStack++ was meant for powerful servers and was not made with consideration of low-cost devices that are prerequisites for widespread adaptation of edge computing.

The reason for this is that the RPi is a low-cost and low-power consumption device (Bekaroo; Santokhee, 2016). Therefore this article shows that the RPi can be used for virtualization on the Edge. More importantly, it demonstrates that virtualization can be enabled with OpenStack using traditional instances on RPi. Furthermore, it has another advantage: the controller resources can be shared with the RPi when a virtual machine is booting from the RPi. This allows the launching of more significant instances and more memory-hungry virtual machines to meet the needs of applications that need more memory than the RPi's limited memory will allow.

According to performance tests, memory is the most degraded resource as low-cost hardware serves virtual machines. This paves the way for new memory allocation formats on low-cost devices.

Our infrastructure proposal based on the ETSI framework enabled us to launch virtual machines uniformly. Namely, MANO does not need to deal with different virtualization technologies for each hardware (high-performance and low-cost). Also, the limitations verified in this work open the way for advances in the virtualization spectrum on single-board computers.

5.1 Main Contributions

Our contribution to the process of enabling virtualization at the edge of the network for future service provision includes the following:

1. Provide an alternative to the available options currently being used at the network's edge on resource constraint devices such as the raspberry pi.

2. Demonstrate the feasibility and possibility of utilizing RPi at the edge as a low-cost and low-energy consumption device. Therefore, it hastened the development of MEC and implemented the 5G network. Our work and related research could motivate other researchers and investors to get involved in the development process of Edge Computing, which is a facilitator for the implementation of MEC and 5G computing.

3. Introduced an ETSI-compliant infrastructure for service deployment at the edge.

5.2 Technical Contributions

The contributions of this work are as follows:

1. We designed and developed the RPi3 and RPi4 testbeds demonstrating the potential usage of Virtual Machine Edge IaaS architecture using open source components.

2. Virtual machines were created until each device's maximum capabilities were achieved. During the creation of each machine, various performance tests were performed on the host devices, such as memory tests, CPU metrics, swap usage, virtual machine boot time, and temperature for the RPi3 and RPi4.

5.3 Publications

During this work, we prepared some research papers. One was submitted and published.

The work called *Enabling the Management and Orchestration of Virtual Networking Functions on the Edge*, published and presented in the *10th International Conference on Cloud Computing and Services Science (CLOSER 2020).* (RICHARDS; MOREIRA; SILVA, 2020).

5.4 Future Work

For future work, we plan the following activities:

- ❏ Implement a solution that integrates the MANO, the VIM and the infrastructure based on RPi with the MANO will have full control of the creation, deployment, and the entire life cycle of the virtual machines;
- ❏ Deploy various applications on the RPi based infrastructure, and we will conduct various investigations and measurements of their performance.

This future work will better understand the supported use cases and provide information about the limitations associated with such scenarios, thus bringing knowledge about the weaknesses and strengths of using low-cost hardware on edge.

Bibliography

Ahmad, M.; Alowibdi, J. S.; Ilyas, M. U. viot: A first step towards a shared, multi-tenantiot infrastructure architecture. In:2017 IEEE International Conference onCommunications Workshops (ICC Workshops). [S.l.: s.n.], 2017. p. 308-313. ISSN2474-9133. https://doi.org/10.1109/ICCW.2017.7962675

AKRIVOPOULOS, O. et al. A Fog Computing-Oriented, Highly Scalable IoTFramework for Monitoring Public Educational Buildings. In:2018 IEEE InternationalConference on Communications (ICC). [S.l.: s.n.], 2018. p. 1-6. https://doi.org/10.1109/ICC.2018.8422489

AULIYA, Y. N. Y. A.; WULANDARI, D. A. R. Performance comparison of docker andlxd with apachebench.Journal of Physics, n. 6, p. 1-6, 2019 https://doi.org/10.1088/1742-6596/1211/1/012042

Barik, R. K. et al. Performance analysis of virtual machines and containers in cloudcomputing. In:2016 International Conference on Computing, Communicationand Automation (ICCCA). [S.l.: s.n.], 2016. p. 1204-1210. ISSN null. https://doi.org/10.1109/CCAA.2016.7813925

Bekaroo, G.; Santokhee, A. Power consumption of the raspberry pi: A comparativeanalysis. In:2016 IEEE International Conference on Emerging Technologiesand Innovative Business Practices for the Transformation of Societies(EmergiTech). [S.l.: s.n.], 2016. p. 361-366. ISSN null. https://doi.org/10.1109/EmergiTech.2016.7737367

BELLAVISTA, P. et al. Human-Enabled Edge Computing: Exploiting the Crowd as aDynamic Extension of Mobile Edge Computing.IEEE Communications Magazine,v. 56, n. 1, p. 145-155, jan. 2018. ISSN 0163-6804. https://doi.org/10.1109/MCOM.2017.1700385

CAPROLU ROBERTO DI PIETRO, F. L. S. R. M. Edge computing perspectives:Architectures, technologies, and open security issues. p. 1-8, 2019 https://doi.org/10.1109/EDGE.2019.00035

Chang, H. et al. Bringing the cloud to the edge. In:2014 IEEE Conference onComputer Communications Workshops (INFOCOM WKSHPS). [S.l.: s.n.],2014. p. 346-351. ISSN null. https://doi.org/10.1109/INFCOMW.2014.6849256

Gouareb, R.; Friderikos, V.; Aghvami, A. Virtual network functions routing andplacement for edge cloud latency minimization.IEEE Journal on Selected Areas inCommunications, v. 36, n. 10, p. 2346-2357, Oct 2018. ISSN 1558-0008. https://doi.org/10.1109/JSAC.2018.2869955

HAJJI, F. P. T. W. Understanding the performance of low power raspberry pi cloud forbig data. p. 1-14, 2016 https://doi.org/10.3390/electronics5020029

ILYAS MUNEEB AHMAD, S. S. M. U. Internet-of-things-infrastructure-as-a-service:The democratization of access to public internet-of-things infrastructure. p. 1-15, 2020. https://doi.org/10.1002/dac.4562

JOHNSTON MIHAELA APETROAIE-CRISTEA, M. S. S. J.; COX, S. J. Applicabilityof commodity, low cost, single board computers for internet of things devicess. p. 1-6,2016. https://doi.org/10.1109/WF-IoT.2016.7845414

KRISTIANI, E. et al. Implementation of an edge computing architecture usingopenstack and kubernetes. In: KIM, K. J.; BAEK, N. (Ed.).Information Scienceand Applications 2018. Singapore: Springer Singapore, 2019. p. 675-685. ISBN978-981-13-1056-0. https://doi.org/10.1007/978-981-13-1056-0_66

LEI, L. et al. Collaborative Edge Caching through Service Function Chaining:Architecture and Challenges.IEEE Wireless Communications, v. 25, n. 3, p. 94-102,jun. 2018. ISSN 1536-1284. https://doi.org/10.1109/MWC.2018.1700321

LI, Y.; CHEN, M. Software-defined network function virtualization: A survey.IEEEAccess, IEEE, v. 3, p. 2542-2553, 2015. https://doi.org/10.1109/ACCESS.2015.2499271

Li, Z. et al. Performance overhead comparison between hypervisor and container basedvirtualization. In:2017 IEEE 31st International Conference on AdvancedInformation Networking and Applications (AINA). [S.l.: s.n.], 2017. p. 955-962.ISSN 1550-445X. https://doi.org/10.1109/AINA.2017.79

Marques, W. d. S. et al. Evaluating container-based virtualization overhead on thegeneral-purpose iot platform. In:2018 IEEE Symposium on Computers andCommunications (ISCC). [S.l.: s.n.], 2018. p. 00008-00013. ISSN 1530-1346 https://doi.org/10.1109/ISCC.2018.8538602

MATTHEWS, S. J. Harnessing single board computers for military data analytics. n. 24,p. 1-24, 2018 https://doi.org/10.1201/9780429445491-4

Milojičić, D.; Llorente, I. M.; Montero, R. S. Opennebula: A cloud management tool.IEEE Internet Computing, v. 15, n. 2, p. 11-14, March 2011. ISSN 1941-0131 https://doi.org/10.1109/MIC.2011.44

MORABITO, J. K. R.; KOMU, M. Hypervisors vs. lightweight virtualization: aperformance comparison. n. 8, p. 1-8, 2015 https://doi.org/10.1109/IC2E.2015.74

MORABITO, R. Virtualization on internet of things edge devices with containertechnologies: A performance evaluation. v. 5, p. 1-16, 2017 https://doi.org/10.1109/ACCESS.2017.2704444

Pahl, C. et al. A container-based edge cloud paas architecture based on raspberry piclusters. In:2016 IEEE 4th International Conference on Future Internet ofThings and Cloud Workshops (FiCloudW). [S.l.: s.n.], 2016. p. 117-124. ISSN null https://doi.org/10.1109/W-FiCloud.2016.36

PARK SEULGI KIM, Y. A. a.-Y. J. D. Lired: A light-weight real-time fault detectionsystem for edge computing using lstm recurrent neural networks. p. 1-15, 2018. https://doi.org/10.3390/s18072110

Parvez, I. et al. A survey on low latency towards 5g: Ran, core network and cachingsolutions.IEEE Communications Surveys Tutorials, v. 20, n. 4, p. 3098-3130,Fourthquarter 2018. ISSN 2373-745X https://doi.org/10.1109/COMST.2018.2841349

Pierre, G.; Stratan, C. Conpaas: A platform for hosting elastic cloud applications.IEEEInternet Computing, v. 16, n. 5, p. 88-92, Sep. 2012. ISSN 1941-0131 https://doi.org/10.1109/MIC.2012.105

PUTHAL, D. et al. Secure and Sustainable Load Balancing of Edge Data Centers inFog Computing.IEEE Communications Magazine, v. 56, n. 5, p. 60-65, maio 2018.ISSN 0163-6804 https://doi.org/10.1109/MCOM.2018.1700795

SALLENT, S. et al. Fibre project: Brazil and europe unite forces and testbeds for theinternet of the future. In: KORAKIS, T.; ZINK, M.; OTT, M. (Ed.).Testbeds andResearch Infrastructure. Development of Networks and Communities. Berlin,Heidelberg: Springer Berlin Heidelberg, 2012. p. 372-372. ISBN 978-3-642-35576-9 https://doi.org/10.1007/978-3-642-35576-9_33

SHARMA LUCAS CHAUFOURNIER, P. S. Y. T. P. Containers and virtual machinesat scale: A comparative study. p. 1-13, 2016. https://doi.org/10.1145/2988336.2988337

Shi, W. et al. Edge computing: Vision and challenges.IEEE Internet of ThingsJournal, v. 3, n. 5, p. 637-646, Oct 2016. ISSN 2372-2541 https://doi.org/10.1109/JIOT.2016.2579198

Silva, F. S. D. et al. Necos project: Towards lightweight slicing of cloud federatedinfrastructures. In:2018 4th IEEE Conference on Network Softwarization andWorkshops (NetSoft). [S.l.: s.n.], 2018. p. 406-414. ISSN null. https://doi.org/10.1109/NETSOFT.2018.8460008

TAO QI XIA, Z. H. C. L. L. M. S. Y. Z.; LI, Q. A survey of virtual machine managementin edge computing.PROCEEDINGS OF THE IEEE, v. 107, n. 8, p. 1-24, 2019. https://doi.org/10.1109/JPROC.2019.2927919

TONG, L.; LI, Y.; GAO, W. A hierarchical edge cloud architecture for mobile computing.In:IEEE INFOCOM 2016 - The 35th Annual IEEE International Conferenceon Computer Communications. [S.l.: s.n.], 2016. p. 1-9. https://doi.org/10.1109/INFOCOM.2016.7524340

van Kempen, A. et al. Mec-conpaas: An experimental single-board based mobileedge cloud. In:2017 5th IEEE International Conference on Mobile CloudComputing, Services, and Engineering (MobileCloud). [S.l.: s.n.], 2017. p. 17-24.ISSN null https://doi.org/10.1109/MobileCloud.2017.17

Zhang, J. et al. Energy-latency tradeoff for energy-aware offloading in mobile edgecomputing networks.IEEE Internet of Things Journal, v. 5, n. 4, p. 2633-2645, Aug2018. ISSN 2372-2541. https://doi.org/10.1109/JIOT.2017.2786343

ZHANG LING LIU, C. P. Q. D. L. W. Q.; ZHOU, W. A comparative study of containersand virtual machines in big data environment. p. 1-8, 2018. https://doi.org/10.1109/CLOUD.2018.00030

ZHU, C. et al. Vehicular Fog Computing for Video Crowdsourcing: Applications,Feasibility, and Challenges.IEEE Communications Magazine, v. 56, n. 10, p. 58-63,out. 2018. ISSN 0163-6804. https://doi.org/10.1109/MCOM.2018.1800116

BHOSALE, V. B. K. Deploying virtualization on ubuntu using libvirt and kvm or qemubare metal hypervisor. p. 1-3, 2016 https://www.academia.edu/30686337/Deploying_Virtualization_on_Ubuntu_using_Libvirt_and_KVM_QEMU_Bare_Metal_Hypervisor

HA, K.; SATYANARAYANAN, M. Openstack++ for cloudlet deployment.School ofComputer Science Carnegie Mellon University Pittsburgh, 2015 https://www.cs.cmu.edu/~satya/docdir/CMU-CS-15-123.pdf

HU, Y. C. et al. Mobile edge computing-a key technology towards 5g.ETSI whitepaper, v. 11, n. 11, p. 1-16, 2015 https://www.etsi.org/images/files/etsiwhitepapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

KAUP1 STEFAN HACKER2, E. M. C. M. D. H. F. The progress of the energy-efficiency of single-board computers. p. 1-8, 2018. https://www.netsys.ovgu.de/netsys_media/publications/NetSys_TR_2018_01.pdf

KIM, Y. Powering the internet of things. 2020. https://nanohub.org/resources/33743/download/2020.06.21-IoT-Power-Kim-DRC2020.pdf

KUMAR, R. et al. Open source solution for cloud computing platform using openstack.IJCSMC, v. 3, n. 5, p. 89 - 98, maio 2014. ISSN 2320-088X. https://www.researchgate.net/publication/263581733_Open_Source_Solution_for_Cloud_Computing_Platform_Using_OpenStack

LEON., D. von et al. A performance exploration of architectural options for amiddleware for decentralised lightweight edge cloud architectures. In: INSTICC.Proceedings of the 3rd International Conference on Internet of Things, BigData and Security - Volume 1: IoTBDS,. [S.l.]: SciTePress, 2018. p. 73-84. ISBN978-989-758-296-7. https://www.scitepress.org/Papers/2018/66774/66774.pdf

PAOLINO, M. et al. T-kvm: A trusted architecture for kvm arm v7 and v8 virtualmachines securing virtual machines by means of kvm, trustzone, tee and selinux. In:InProc. of the Sixth International Conference on Cloud Computing, GRIDs,and Virtualization. [S.l.: s.n.], 2015
https://www.thinkmind.org/download.php?articleid=cloud_computing_2015_2_30_20114

RODOLA, G.Psutil package: a cross-platform library for retrieving information on running processes and system utilization. 2016. https://pypi.python.org/pypi/psutil

CANONICAL.Linux Containers. 2020. Disponível em: https://linuxcontainers.org

SCHILLER, E. et al. CDS-MEC: NFV/SDN-based Application Management for MEC in5g Systems.Computer Networks, v. 135, p. 96 – 107, 2018. ISSN 1389-1286. Disponívelem:
http://www.sciencedirect.com/science/article/pii/S138912861830080X

RICHARDS, V.; MOREIRA, R.; SILVA, F. Enabling the Management and Orchestrationof Virtual Networking Functions on the Edge. In: . [s.n.], 2020. p. 338–346. ISBN978-989-758-424-4. Disponível e. Disponível em:
https://www.scitepress.org/PublicationsDetail.aspx?ID=ZNyWFADuQ3A=&t=1

SILVA, A. P. et al. 5ginfire: An end-to-end open5g vertical network function ecosystem.Ad Hoc Networks, v. 93, p. 101895, 2019. ISSN 1570-8705. Disponível em:
http://www.sciencedirect.com/science/article/pii/S1570870518309387

ZHANG, G.; RAVISHANKAR, M. Exploring vendor capabilities in the cloudenvironment: A case study of alibaba cloud computing.Information Management,v. 56, n. 3, p. 343 – 355, 2019. ISSN 0378-7206. Disponível em:
http://www.sciencedirect.com/science/article/pii/S0378720617311199