School of Computing
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Investigating Performance Overheads Of Virtual Machines And Containers

Xie Yuncheng

Submitted in accordance with the requirements for the degree of
MSc Advanced Computer Science

(2018/2019)

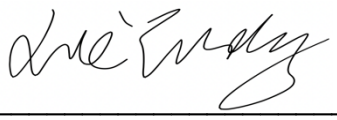
The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Project Report	*Report*	*SSO (03/09/18)*
Printed Report	*Report*	*SSO (03/09/18)*
Software code	*Software codes*	*Supervisor (03/09/18)*

Type of Project: Exploratory Software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) 

Summary

Cloud computing technology has entered a rapid growth period from the initial development, and more and more different types of enterprises have begun to use cloud computing services. At the same time, the core technology of cloud computing is updating constantly. Container virtualization technology has successfully replaced the popularity of virtual machines with its features of lightness, flexibility and rapid deployment.

This project is based on the technology of containers and virtual machines, deploy containers and virtual machines in the experimental environment. Building an automated application to collect data from performance metric tool and benchmark tool, comparing the performance of servers, containers, virtual machines to evaluate relative overhead of containers and virtual machines.

Acknowledgements

First and foremost, I would like to show my deepest gratitude to my supervisor Prof.Xu Jie and Dr.Paul Townend for their patience guidance and affirmation of my project, as well as for the valuable meetings and discussions we have. I am very grateful to my assessor, Dr. Samuel Wilson for his help. He gave valuable comments to my report.

Secondly, I also would like to show my deepest gratitude to Dr. Yang Renyu. He gave me staged guidance and shared valuable experience during software development, that made my project successful.

Finally, I am very thank for my family and friends who were supporting and encouraging me to give the best effort.

Table of Contents

Summary ..... **iii**
Acknowledgements ..... **iv**
Table of Contents ..... **v**
List Of Figures ..... **viii**
List Of Tables ..... **ix**
Chapter 1 Introduction **1**
1.1 Context 1
1.2 Aim 2
1.3 Objective 2
1.4 Motivation 2
1.5 Problem Statement 3
1.6 Deliverables 3
1.7 Project Management And Methodology 4
1.7.1 Version Control 4
1.7.2 Software implement management 4
1.7.3 Report Management 6
1.7.4 Methodology 6
1.8 Risk Assessment 7
1.9 Report Structure 8
Chapter 2 Background Research **9**
2.1 Overview 9
2.2 Cloud Computing 9
2.3 Defination Of Performance Overheads 11
2.4 Virtualization Technology 12
2.4.1 Hardware Virtualization 13
2.4.2 Virtual Machine 15
2.4.3 Full Virtualization 16
2.4.4 Para Virtualization 17
2.4.5 Operating System Virtualization 18
2.5 Binary Translation 19
2.6 Container 20
2.7 Virtual Machine VS Container 22

Chapter 3 Project Design **26**
3.1 Overview 26
3.2 Plans 26
3.3. Architecture 28
3.3.1 External Architecture 29
3.3.2 Internal Architecture 30
3.4 Tool 28
3.4.1 Python 31
3.4.2 Qemu 31
3.4.3 Docker 32
3.4.4 Sysbench 33
3.4.5 Sysstat 33
Chapter 4 Project implementation **35**
4.1 Overview 35
4.2 Environment 35
4.3 Automated Testing Application 37
4.3.1 Packages 37
4.3.2 Development 38
4.3.3 Test Parameter 40
Chapter 5 Results And Evaluation **42**
5.1 Overview 42
5.2 Methodology Evaluation 42
5.3 Performance Evaluation 42
5.4 Overhead Evaluation 44
Chapter 6 Conclusion **48**
6.1 Overview 48
6.2 Project Accomplishment 48
6.3 Project Limitation 49
6.4 Future Work 49
6.5 Personal Relection 50
List of References **52**
Appendix A External Materials **55**

List Of Figures

Figure 1.1 Project Version Control....4
Figure 1.2 Report Management....4
Figure 1.3 Software Implement Management....5
Figure 1.4 Waterfall Methodology....6
Figure 1.5 Agile Methodology....7
Figure 2.2.1 Type of Cloud Computing....11
Figure 2.4.1 Virtualization....12
Figure 2.4.1.1 Hardware Virtualization....14
Figure 2.4.2.1 Bare-Metal Hypervisor....16
Figure 2.4.2.2 Hosted Hypervisor....16
Figure 2.4.3.1 Full Virtualization....17
Figure 2.4.4.1 Para Virtualization....18
Figure 2.4.5.1 Operating System Virtualization....19
Figure 2.5.1 Binary Translation Approach....20
Figure 2.6.1 Relationship between Cgroup and Namespace....21
Figure 2.7.1 Architecture of VM and Container....23
Figure 3.2.1 Running Performance Metric Tool Background....27
Figure 3.2.2 Running Performance Metric Tool Real Time....28
Figure 3.3.2.1 External Architecture....29
Figure 3.3.2.2 Native Internal Architecture....30
Figure 3.4.3.1 Comparison LXC and Docker....32
Figure 3.4.4.1 Sysbench....33
Figure 3.4.5.1 Pidstat....34
Figure 4.3.2.1 Project UML Figure....39
Figure 5.3.1 Set of performance metric figure....43
Figure 5.4.1 CPU Overhead....44
Figure 5.4.2 A Set Of Overhead for each test....44
Figure 5.4.3 CPU eps and latency....45
Figure 5.4.4 Memory Usage....46
Figure 5.4.5 File I/O fsyncs....46
Figure 5.4.6 Threads eps....47

List Of Tables

Table 1.2 Risk Assessment 7
Table 2.7.1 Comparison of Virtual Machine and Container 24
Table 4.2.1 Environment 35
Table 4.3.3.1 Test Parameter 40
Table 5.4.1 Overhead Comparison 45

Chapter 1
Introduction

1.1 Context

In the era of big data, cloud computing has become a mainstream computing model in the IT industry today. Cloud computing is an emerging business computing model that distributes computing tasks across resource pools of large numbers of host servers, enabling applications to acquire computing, storage space, and various software services as needed. At the technical architecture level, there are three layers which are IaaS(Infrastructure as a Service), PaaS(Platform as a Service) and SaaS(Software as a Service). SaaS is the most commercial layer, opening up application services to all customers through cloud computing technology, providing a business model which support multi-tenancy and on-demand. The bottom layer of cloud computing is a resource pool of infrastructure equipment.[1] The infrastructure resource pool directly affects the quality of other cloud computing type services, such as the availability and scalability of PaaS and SaaS, which will be significantly affected.

As one of the core technologies of cloud computing, virtualisation technology has mature development and application. The two leading core virtualisation technologies are Virtual Machine and Container. Both are using different virtualisation underlying technologies.

Virtual Machine is one of the virtual areas which is allocated by computer. This virtual area has computer characteristics. Each virtual area does not affect each other, which means Virtual Machine have secure isolation.[2] Virtual Machine plays a significant role in our lives and used by enterprises widely.

Container is a series of processes that are isolated from the rest of the system. All the files needed to run these processes are provided by another image, which means that Container is portable and consistent throughout the development process from development to testing to production. Thus, containers run much faster than relying on traditional environments. Containers are conventional and easy to use.

1.2 Aim

To create script software to achieve automated testing methodologies, which can collect the benchmark data and performance data from Virtual Machine, Container and Native server.

Based on those data, analysis and compare the overheads of Virtual Machine and Container.

1.3 Objective

1) Research Virtual Machine and Container deploy methodologies.

2) Comparing and selecting benchmark tool and performance tool which can use on Virtual Machine and Container.

3) Building and testing script to collect or monitor performance metric of Virtual Machine and Container.

4) Automating run script to implement automated test application.

5) Using data analysis tool to analyse the collected data, compare the overhead of Virtual Machine and Container

1.4 Motivation

The performance of different applications in Virtual Machine and Container may be different. Based on their infrastructure, their performance aspects may have significant differences for different applications or software. And these differences can often cause more costs to applications and even servers. For example, an application runs under Virtual Machine and Container, assuming that memory overhead under Virtual Machine is 20% higher than the memory overhead under Container. This extra 20% memory may cause much workload on Virtual Machine. So it is necessary to test the performance overhead under Virtual Machine and Container in order to deploy the application of user to the appropriate virtual technology.

1.5 Problem Statement

First, need to consider whether the performance under Native server also needs to collect when comparing Virtual Machine and Container performance. Both Virtual Machine and Container based on the environment of Native server, so Native server can use the same performance metric collection method, which can show the difference between Virtual Machine and Container precisely. Whether the performance under Container is similar to Native. Whether the performance under Virtual Machine is the same as Native in some aspects. These problems need to consider.

Second, from the perspective of the user, need to consider which virtual technology is the most suitable for deploying the application of the user. Virtual Machine and Container may have the same performance metric in some aspects. When some applications take up fewer

system resources, they are not much different probably, and the environment in Virtual Machine is safer than Container. At this time, the application is better to deploy on Virtual Machine. So it is a problem that applications are deployed under Virtual Machine and Container to get the maximum of benefits.

Finally, there are more and more technologies related to Virtual Machine and Container, but all this project needs is an environment based on Virtual Machine and Container. Many related technologies or frameworks have optimised Virtual Machine environment and added great extra features. Although these are more convenient and simple for users. However, this may affect the performance comparison between Virtual Machine and Container under origin environment.

1.6 Deliverables

Guideline Files

- o Commands which can deploy Virtual Machine and Container.
- o Benchmark commands.
- o Commands which is configuring the environment of Virtual Machine and Container.
- o Software guide file.

Software Folder

- o Codes of software which including four folder and three files.
- o Data files which collected in host server
- o Figures which generate from collected data

Report(Online and Printed)

1.7 Project Management And Methodology

1.7.1 Version Control

In the preparation of the project, this project is used to prevent data loss, code exceptions and easy modification by the author. Version control is a system that records changes in the content of one or more documents. This system can automatically help us backup every change of the file, and can quickly restore to any backup state. GitHub is an excellent version control system. Most companies and users recognize it. This project is using GitHub to record each version, which code change(Fig. 1.1), and it including all guide files about this project. The following URL is this project GitHub repository address: https://github.com/Reggiecril/Performance

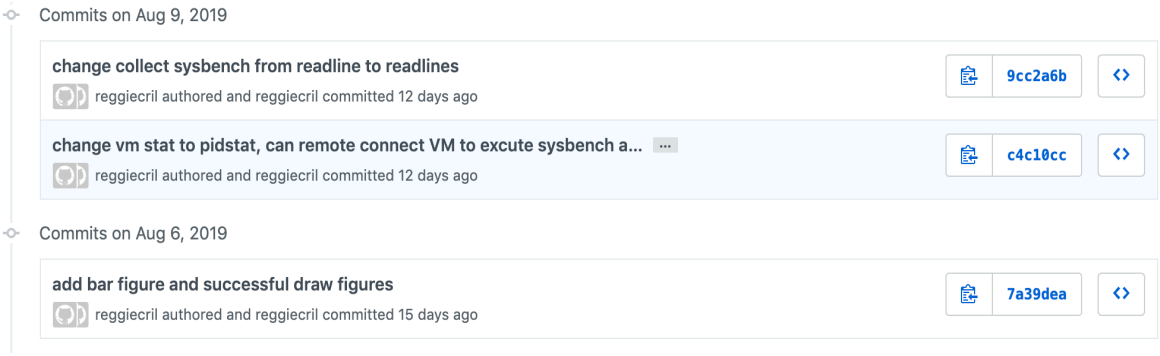


Figure 1.1 Project Version Control

1.7.2 Software implement management

This project management divides into two parts, which are software and report. Software implement management (Figure 1.3) show processes of the implementation of the software. It lists all milestones from preparation to test software. Every task needs to perform step by step, each step has a fixed time, and this time is enough to ensure the fault-tolerant of a task. The whole project will take about 80 days, including problems that may delay, such as software debugging and hardware resources.

1.7.3 Report Management

Report Management (Fig. 1.2) shows objects while doing the project. Those milestones divide into two stages. The first stage is from Understand Project to Scoping and Planning Document, and it is the preparation stage. In this stage, to identify the problem of project, research recent technology about project and identity why this project is necessary. The second stage is project implementation, including software implement management and report writing. This stage is to implement and achieve all deliverables. This stage will take a long time.

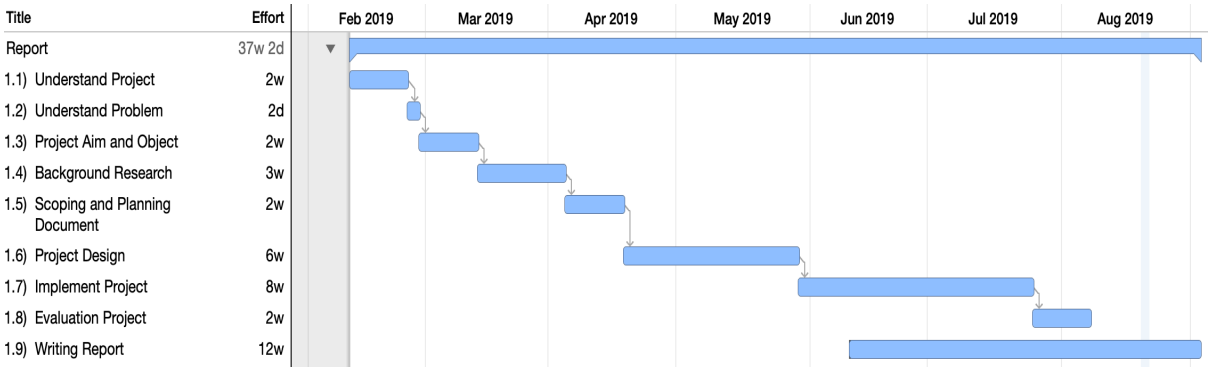


Figure 1.2 Report Management

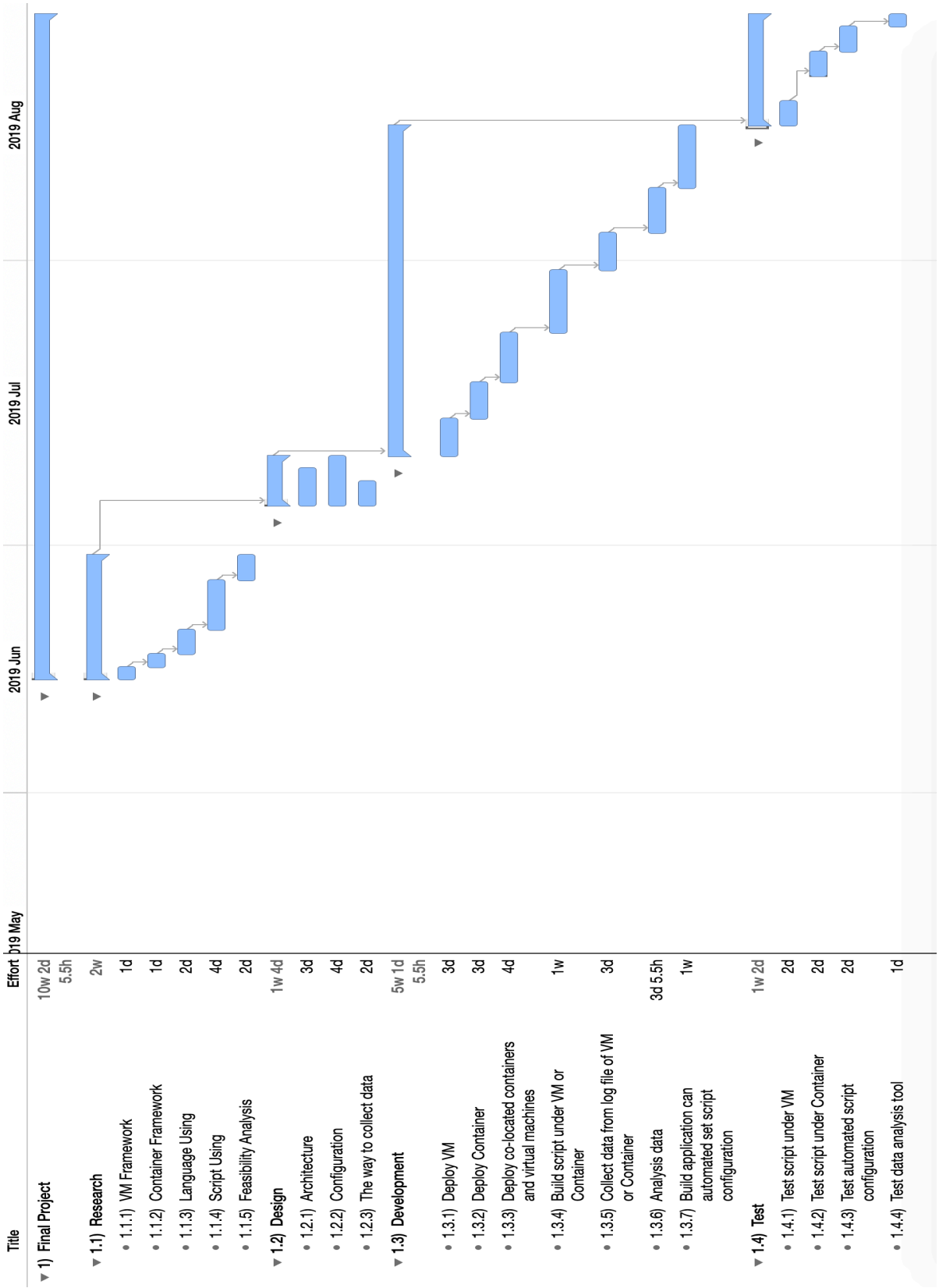


Figure 1.3 Software Implement Management

1.7.4 Methodology

In the initial plan, the methodology of the project is using Waterfall methodology, but this methodology is inefficient during developing. In the waterfall model(Fig. 1.4), the activities of software development are carried out linearly. The current activity collects the results of the previous activity and implements the necessary work.[3] Every stage of development is required to be the best. Especially in the early stage, the quality of the design is related to the cost loss directly. During this project development, each milestone may recursively execute. Because there may be a variety of errors during developing, leading to the current task cannot proceed.

This project uses Agile methodology, and it is the best choice. Agile methodology[4] is a methodology for responding to rapidly changing requirements, using an iterative, step-by-step approach to develop software. (Fig. 1.5) For example, when a software library cannot be installed or does not match the current environment, it may need to be analysed again or compared whether to use this software library for development. If using Waterfall methodology, it must be planned initially to ensure that the library can be installed, but whether it can be installed is unpredictable. Because software needs to be deployed to Native, VM and Container, each platform environment is entirely different. So the installation of the library needs constant testing and debugging before it can be used. Agile methodology meets all the requirements, which can save a lot of development time and output deliverables on time.

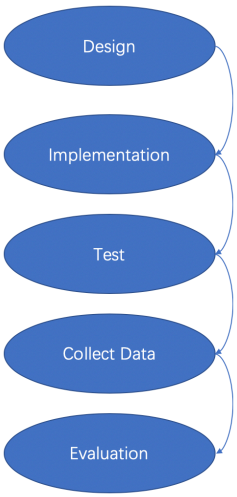


Figure 1.4 Waterfall Methodology

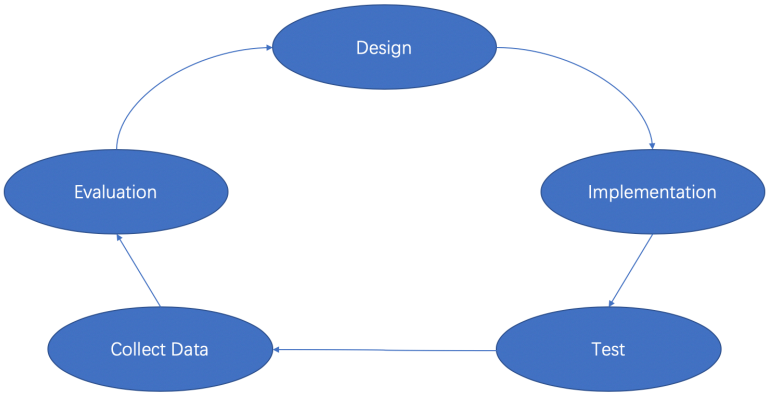


Figure 1.5 Agile Methodology

1.8 Risk Assessment

The following table shows several different risks that probably occur in the project. For example, software bugs. Whether it is simple or complex software, bugs may occur with abnormal parameters, mismatched environments, or even variable errors.

Each risk provides a relative solution to ensure the feasibility of the project.

	RISK	RISK RATING	CONTROL MEASURE NEEDED
1	Software Bugs	4	Expand development time to check software log file, compare with success running log file to fix bugs.
2	Hardware Limitation	3	Check limitation and set script running limitation to protect software success running.
3	Data Loss	4	Use vision control or backup data.
4	Software running out of time	2	Execute another function which is a high priority
5	Physical Reason Ep. Power Failure	3	Backup software before running.

Table 1.2 Risk Assessment

1.9 Report Structure

This report divides into six chapters.

Chapter 1 introduced the problem and background of this project. It also shows the aim and object of this project, given the final deliverables. Finally, it explains all milestones in project management and methodology.

Chapter 2 show several underlying technologies behind this project. Introduced the background of virtualization technology and built on binary translation. Container is different technology that does not use virtualization technology compared to the differences between Virtual Machine and Container.

Chapter 3 presents the design of the experiments that will be performed and preparation works before project implementation. The overall workflow of the project will be discussed. The choice of frameworks for VM and Container will be explained and discuss how to select benchmark and performance metrics tools.

Chapter 4 explains how to set the environment for native server, container and virtual machine. The detail of workflow of this project will discuss. The test parameter of project will show as table.

Chapter 5 will evaluate this project. Methodology will evaluate in this chapter. There are many figures to evaluate the comparison between virtual machine and container. To evaluate the performance and overhead between container and virtual machine.

Chapter 6 summaries accomplishment of project, discusses the limitation of project and points some future work which project can improve. To make a summary which author learn in this project.

Chapter 2
Background Research

2.1 Overview

Background research based on the analysis of arguments or problems, as well as the steps and results required to design and implement solutions. In this project, the two leading technologies are VM and Container. Before carrying out the project, first is research and investigation background of the two technologies and their underlying technologies. Understand the reason why use the two technologies and compare their strengths and weaknesses. Next is understanding what performance overhead means and what methods can be used to test it.

2.2 Cloud Computing

Cloud computing is the product of the evolution of traditional computing technology and network technology. It includes a series of technologies, such as distributed computing, parallel computing, utility computing, grid computing, virtualization, network storage. Cloud computing integrates IT infrastructure from local to the cloud resource pool and manages these resources. Using virtualization technology to authorize resources to users by pay-on-demand, users can obtain the required services without purchasing expensive hardware or software and do not need to know the specific details of the services. At present, cloud computing is still in a period of rapid development, but there is no complete definition. At present, cloud computing is still in a period of rapid development, but there is no unified definition. Among many definitions of cloud computing, the most widely accepted one comes from the National Institute of Standards and Technology of the United States(NIST). [5] It believes that cloud computing is a ubiquitous business model, which can be accessed a pool of shared computer resources consisting of servers, memory, network devices, applications and services.

More and more people use cloud computing services because cloud computing technology has the following apparent characteristics：

• Cloud service providers centralize computing resources into resource pools for unified management, and the services they provide are inexhaustible and varied for users.

• Users can use heterogeneous client platforms such as mobile phones, tablets, laptops, to rent computing resources according to their needs. Only pay for their used resources, without investing too much money to purchase and maintain infrastructure. [6]

• Cloud service providers can provide flexible services; that is, they can allocate resources to users in real-time when users need them. When users no longer use these resources,

they can dynamically recycle them, which can meet user's needs and avoiding the waste of resources.

• Cloud service providers provide users with secure and reliable services based on redundant backup technology and fast deployment technology. Once a node fails to work, the new node will replace the node to provide the same services for users.

• The resources used by the user are not physical; those resources are from a virtual resource pool. So users do not need to know the physical location and underlying logic of these resources.

There are three main service modes of cloud computing. [7]

• Infrastructure as a Service(IaaS). IaaS provides users with physical resources to run any software, and the user does not need to manage the physical resources and service provider is responsible for resource management. The advantage of IaaS is that it allows users to deploy their applications freely, and reduces much cost of purchase and management.

• Platform as a Service(PaaS). PaaS provides users with the cloud platform environment and programming interface for developing applications, and the user must develop with programming languages and libraries supported by the PaaS service provider. So PaaS is limiting the behavior of the user. Its advantage is that users do not need to manage and control the underlying infrastructure, but can control the deployment of applications and the configuration of the environment so that users can develop their applications more effectively.

• Software as a Service(SaaS). SaaS provides users with applications running on infrastructure, and users can use applications deployed in the cloud through the Internet. Its advantage is that users do not need to install and upgrade applications locally, only need to pay for using specialized applications. In this service, users can use specific services anytime, anywhere without professional knowledge.

These three services of cloud computing can independently provide users with the required types of services, but there are some relationships between them. SaaS services may need the platform and interface provided by PaaS, or directly use the necessary computing resources provided by IaaS

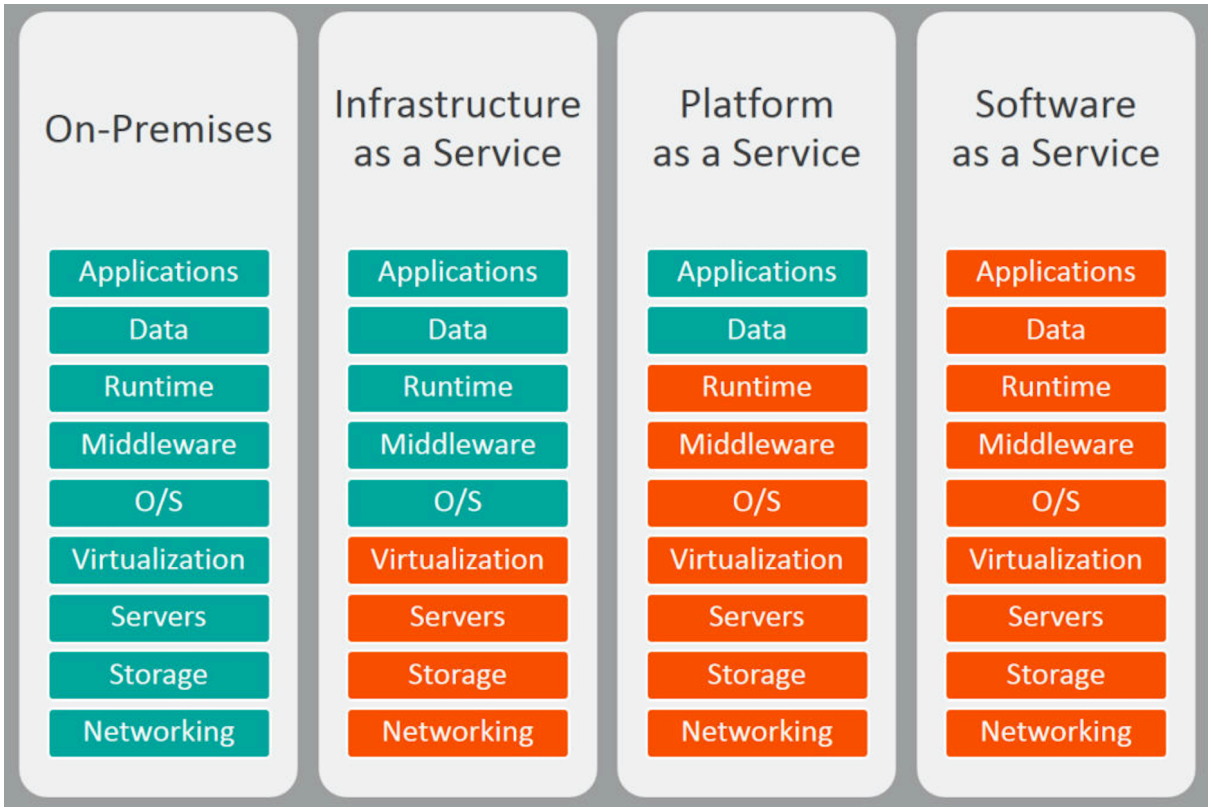


Figure 2.2.1 Type of Cloud Computing[8]

2.3 Definition Of Performance Overhead

Performance overhead is excessive computing time, memory and bandwidth required to perform specific tasks, and it is an additional expense occurring in addition to standard cost. In computer science, it might seem unrelated, but necessary. For example, someone wants to go somewhere so that he may need a car. However, using a car will have a high overhead, so he may want walking. However, if the place is far away, the cost of time will be high, the overhead of using a car is minimal and using a car is the best choice.

Performance overhead is usually used in two cases:

- Hardware
    - o CPU
    - o Memory
    - o Network
    - o IO
- Software
    - o Algorithm(Time complexity, Space complexity)
    - o Data Structure

2.4 Virtualization Technology

Virtualization means abstracts hardware resources. It allows a physical machine to be allocated to multiple virtual instances running concurrently and to share the same physical resources so that the computer operating system can run virtually. (Fig 2.4.1) Virtual machine monitor, also known as Hypervisor, runs on physical resources, controls hardware and creates a guest machine that can run a single operating system.[9]

In the non-virtualized physical architecture, application access to physical layer resources can be controlled by the operating system. After using virtualization technology, multiple operating systems can be virtualized on a physical machine. The application of each operating system is managed by its operating system but requires VMM to control the access of the underlying physical resources.

In the environment of limited resources, the goal of computer science is maximizing the use of resources. Virtualization solves the problem of limited resources very well and improves the utilization of each essential resource at the same time.

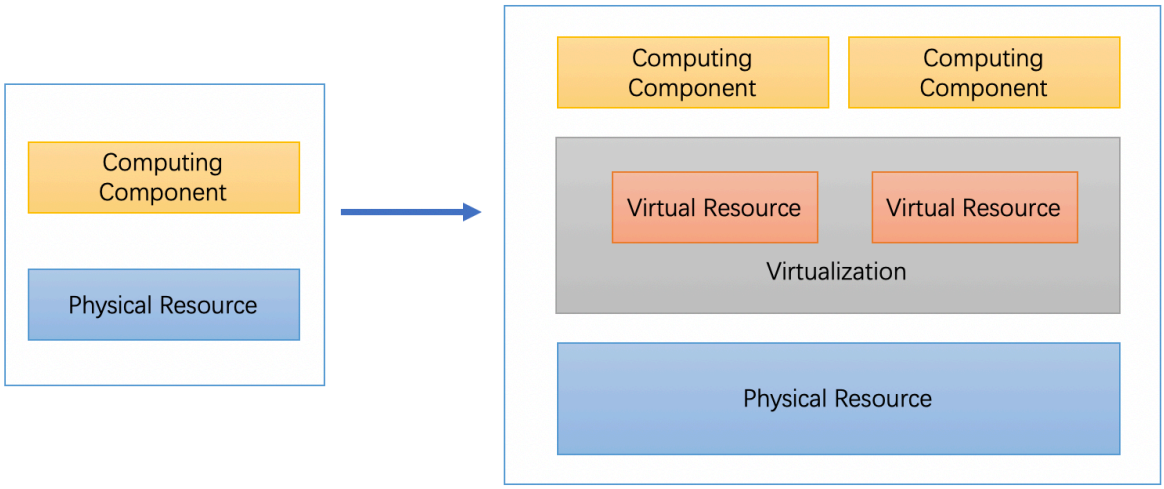


Figure 2.4.1 Virtualization

Virtualization technology has three characteristics.

• Integration of resources

The main task of virtualization technology is to integrate resources. In the digit age, the generated data show explosive growth. Organizations need to solve the problem of how to use these data and resources. The research and development of virtualisation technology provide technical support and application platform for resource integration. Especially in recent years, with the popularization of cloud computing technology, centralised resource management has become more and more advanced, which provides conditions for the development and promotion of cloud technology. At present, the utilisation rate of computer hardware resources in major enterprises is still meagre. Virtualisation technology can be

used to concentrate application on a host server based on keeping the original application unchanged. It can significantly improve the utilization of material resources, reduce hardware updating, and save the cost.

• Reduce Energy Consumption

In the digital age, reducing resource consumption is the focus of technological revolution. [10] Virtualization technology is the way to improve resource utilization, and it can also manage energy consumption reasonably. Virtualization technology can simulate different scenarios, to check all kinds of hardware and software in the computer system and find problems in time, to achieve the goal of reducing energy consumption and achieving green development.

• Cross Platforms

Virtualization technology can make the application platform of the host server more transparent. In the digital age, there are more and more data centers, and the application of computer server is more and more complex. When applications run in different platforms, need to consider the problem of different operator systems and the problem of different middleware. However, virtualization technology can effectively solve such problems. Virtualization technology isolates application and hardware platforms and solves the limitations of cross-platform.

2.4.1 Hardware Virtualization

Hardware virtualization technology is a kind of chip virtualisation technology. It refers to the support of virtualisation technology in the CPU at the hardware layer. It implemented by integrating several virtualization commands on the CPU. The original operating system dynamically loaded into the virtual machine by executing the virtualisation extension commands, and the virtual machine monitor added between the original operating system and the physical hardware.[11] The original operating system which runs in the virtual machine is called guest operator system. After host server is booted, before operator system start, initialise virtual machine monitor firstly then initialise each virtual machine. Each virtual machine runs as if it were on hardware so that it can run its client operating system in complete isolation. Figure 2.4.1.1 shows the concept map of hardware virtualization.

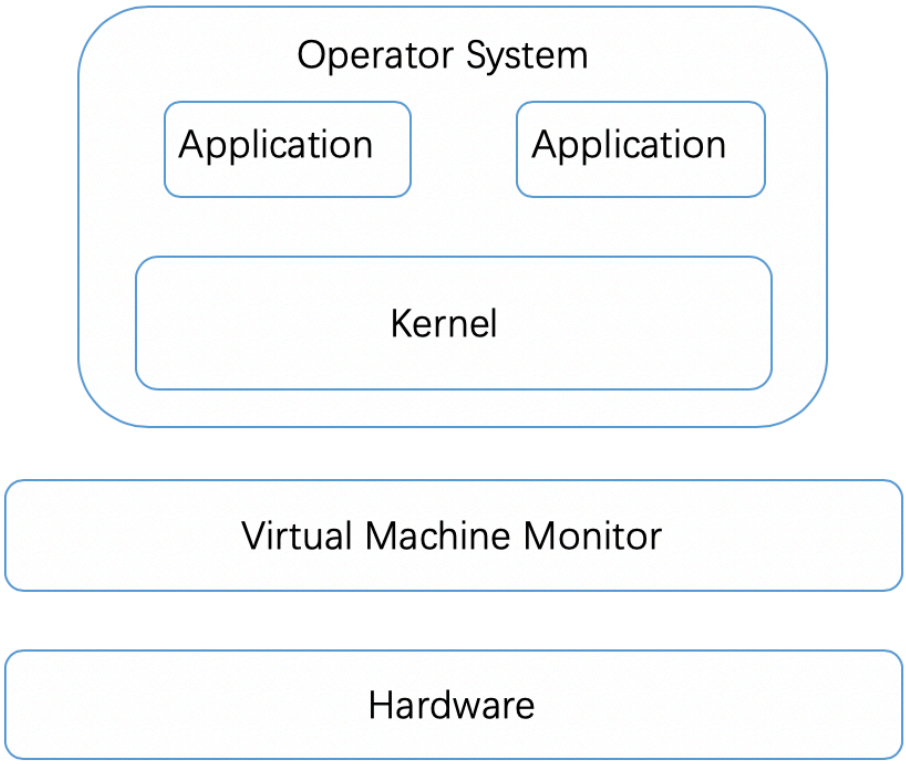


Figure 2.4.1.1 Hardware Virtualization

The virtual machine monitor consists of three parts in physical resources。

• CPU Virtualization

CPU virtualization is the core part of virtual machine monitor, including command simulation, interrupt simulation, injection simulation and symmetric multiprocessor technology simulation.

• Memory Virtualization

Memory virtualization solves the problem of sharing physical memory between virtual machine monitor and client operating system. It isolates the memory of virtual Machine and the memory of the virtual machine monitor. It prevents security vulnerabilities, such as memory operations within a virtual machine affect other virtual machines or virtual machine monitors.[12]

• I / O Virtualization

I/O virtualization is to meet the access requirements of multiple client operating systems to peripheral devices, reusing devices by access interception, device simulation and device sharing.

2.4.2 Virtual Machine

Virtual machine technology is a kind of virtualisation technology. Virtual Machine a complete computer system simulated by software, which has complete hardware system characteristic and running in a completely isolated environment. The basic idea of virtual machine is that abstract hardware of a single computer into several different execution components, so it feels like every independent execution environment runs on its computer.[13] Some of the current virtual machine software is to abstract the physical hardware into an independent component and then to run into the virtual machine. Each virtual machine has its virtual CPU, memory, disk driver, network interface.

Hypervisor plays an essential role in virtual machines. It is an intermediate software layer running between the physical server and the operating system. It allows multiple operating systems and applications to share underlying physical hardware. It can access all physical devices and virtual machines on the server, also known as virtual machine monitor. The hypervisor is the core of all virtualisation technologies. [14]

When the server starts up and executes Hypervisor, it allocates an appropriate amount of memory, CPU, network and disk to each virtual machine, and loads the client operating system of all virtual machines.

The hypervisor has two types.

The first type is bare-metal Hypervisor, which runs directly on the host server to control host hardware and manage the guest operating system. (Fig 2.4.2.1) For example, VMware 5.5 and KVM.

• Features

- o Hardware support is required
- o Virtual Machine Monitor is Main Operating System
- o High operating efficiency

Second type is Hosted Hypervisor, which runs on traditional operating systems like application. (Fig 2.4.2.2) For example, Xen 3.0 and Virtual PC 2004. [15]

• Features

- o Virtual Machine Monitor runs as an application in the main operating system environment
- o Operating efficiency is generally lower than bare-metal hypervisor.

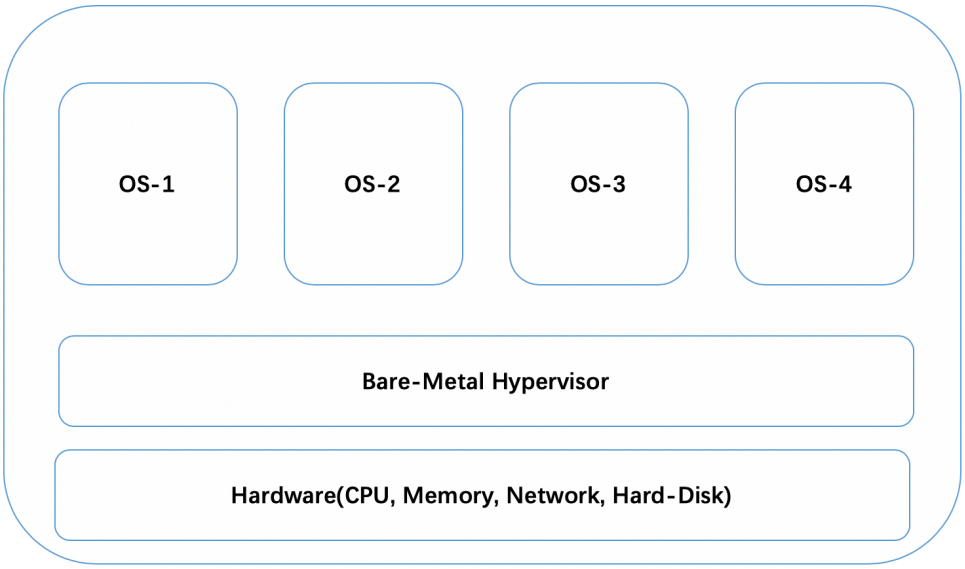


Figure 2.4.2.1 Bare-Metal Hypervisor

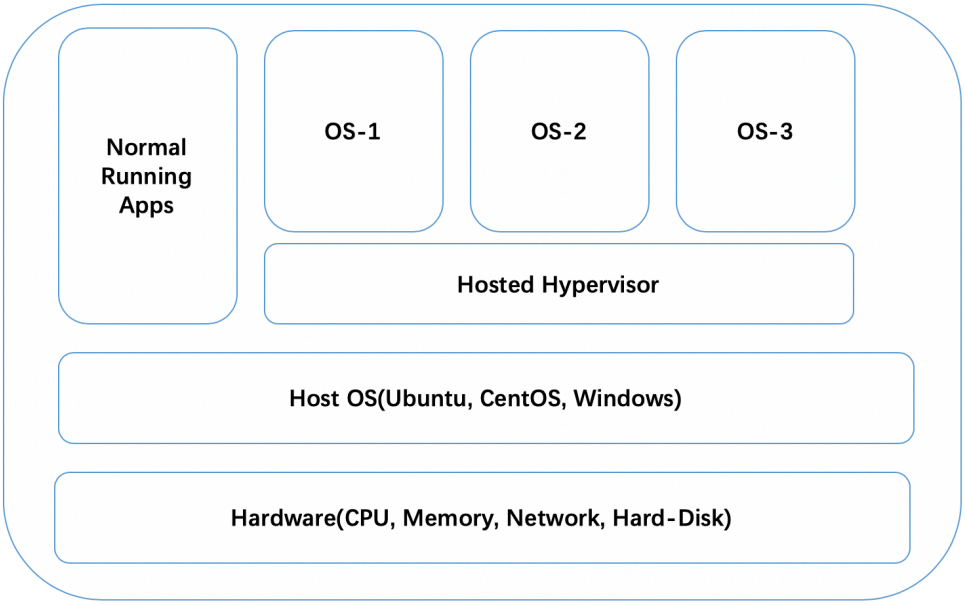


Figure 2.4.2.2 Hosted Hypervisor

2.4.3 Full Virtualization

Full virtualisation adds a software layer between the underlying hardware of the host server and VM, that is virtual machine monitor or Hypervisor. At this time, the virtual machine monitor acts as the host operating system to manage different virtual machines. It hides the physical characteristics of a specific computing platform and provides users with an abstract, unified and simulated computing environment. On virtual machine platform, multiple virtual machines can be simulated, and its successes to run different types of operating systems on a single machine. Full virtualisation runs faster than hardware virtualisation, but its

performance is not as good as that of bare machines, because Hypervisor takes up some resources.

The most significant advantage of full virtualisation is that the operating system has not been modified.[16] Its only limitation is that the operating system must be able to support the underlying hardware.

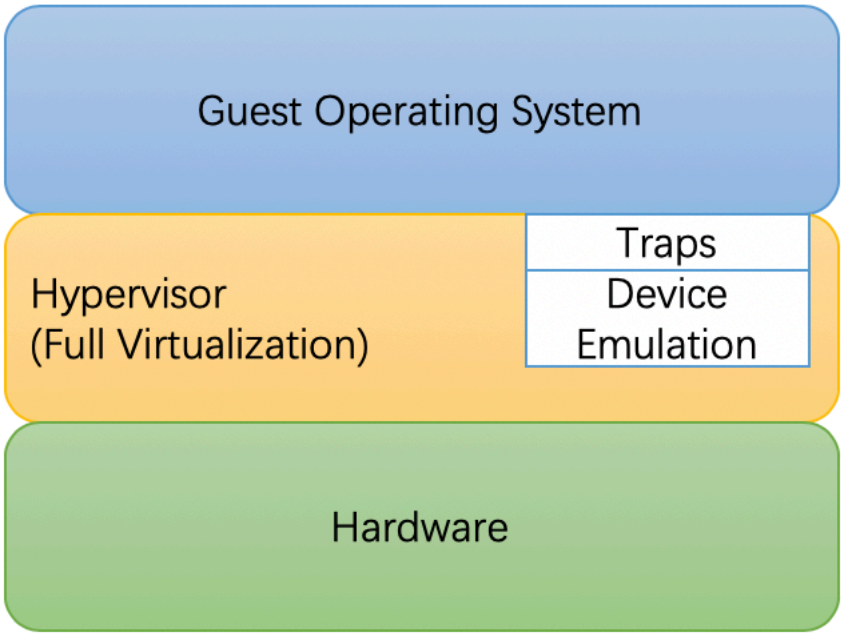


Figure 2.4.3.1 Full Virtualization

2.4.4 Para Virtualization

Para virtualisation is similar to full virtualisation. The difference is that the core code of the client operating system needs to be modified. [17] The special virtualised application program interface is added to optimise the commands issued by the client operating system. It can work with the virtual machine monitor to reduce the tasks of virtual machine monitor and the host server and improve the performance of the virtual machine. The disadvantage of para virtualisation is needed to modify the client operating system.

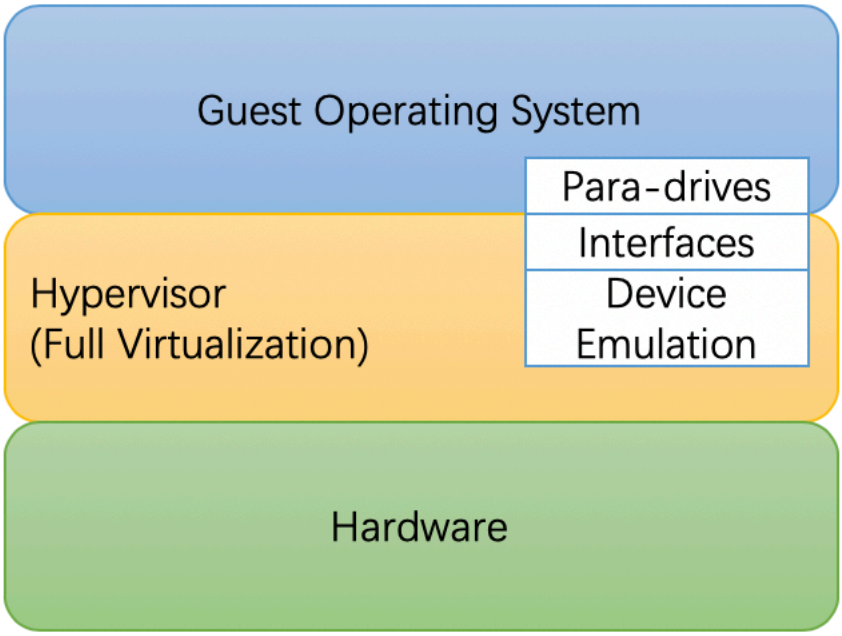


Figure 2.4.4.1 Para Virtualization

2.4.5 Operating System Virtualization

Operating system virtualisation can also be called container-based virtualisation technology, which virtualises multiple servers on the central operating system and supports isolating each virtual server on a single operating system. It is like running a virtual operating system in a container. The guest operating system shares the same operating system resources, and every virtual server runs the same kernel.

Operating system virtualisation requires modification of the operating system kernel, which has the advantage of having the performance of the original host.[18]Containers do not emulate any underlying hardware but virtualise the operating system or applications to interact with the host, which then makes appropriate calls to use real hardware. Container-based virtualisation technology provides a shared virtualised operating system, including a unique root file system, a shared executable program and library file set. Each guest operating system can be started or shut down like a standard operating system, and even restarted in a few seconds if necessary. Operating system virtualisation software includes Solaris Container and Open VZ.[19]

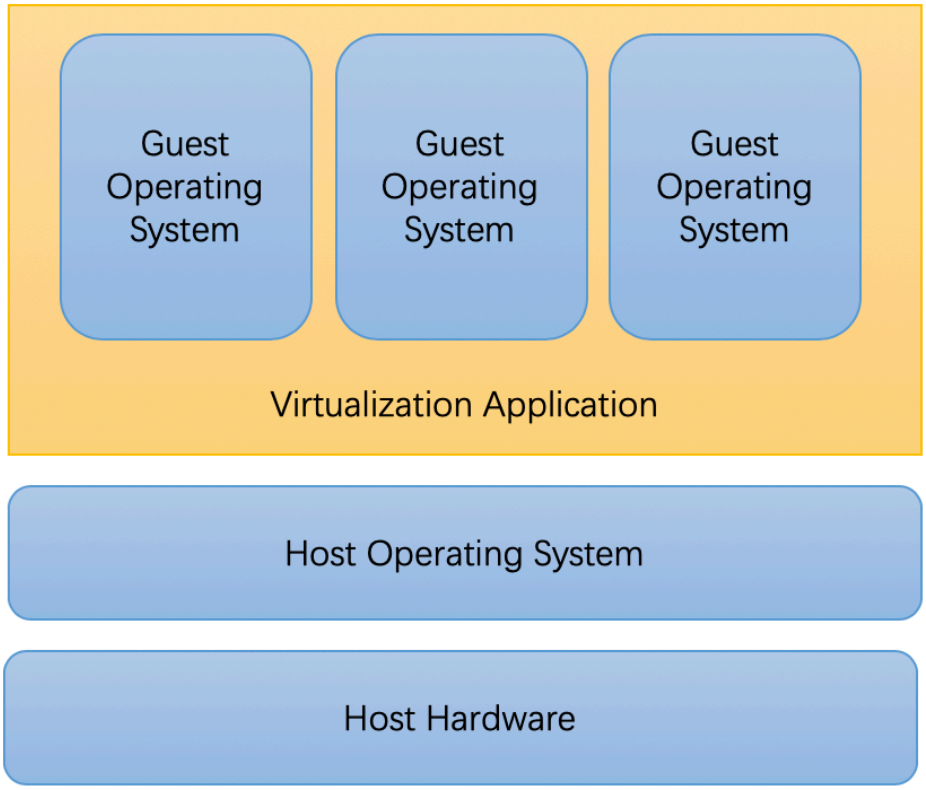


Figure 2.4.5.1 Operating System Virtualization

2.5 Binary Translation

Binary translation is a code conversion technology on different platforms. It can translate and execute origin platform programs on the target platform without source code and hardware support of the target platform. [20] The origin of binary translation is that it restores binary code to the binary translation of high-level language programs through this technology. The characteristic of binary translation is that when translating a program, only need the binary code of the program to be translated. Binary translation technology plays an essential role in software transplantation, especially in virtualisation.

Figure 2.5.1 shows the binary translation approach. Based on this approach, the guest operating system located on a higher ring, and the kernel code is translated by the Hypervisor to influence the virtual hardware it runs. The Hypervisor translates all operating system commands in real-time. The Hypervisor provides all the services provided by the hardware for the virtual machine. The user code that typically runs on Ring 3 is directly executed to lead to higher performance.

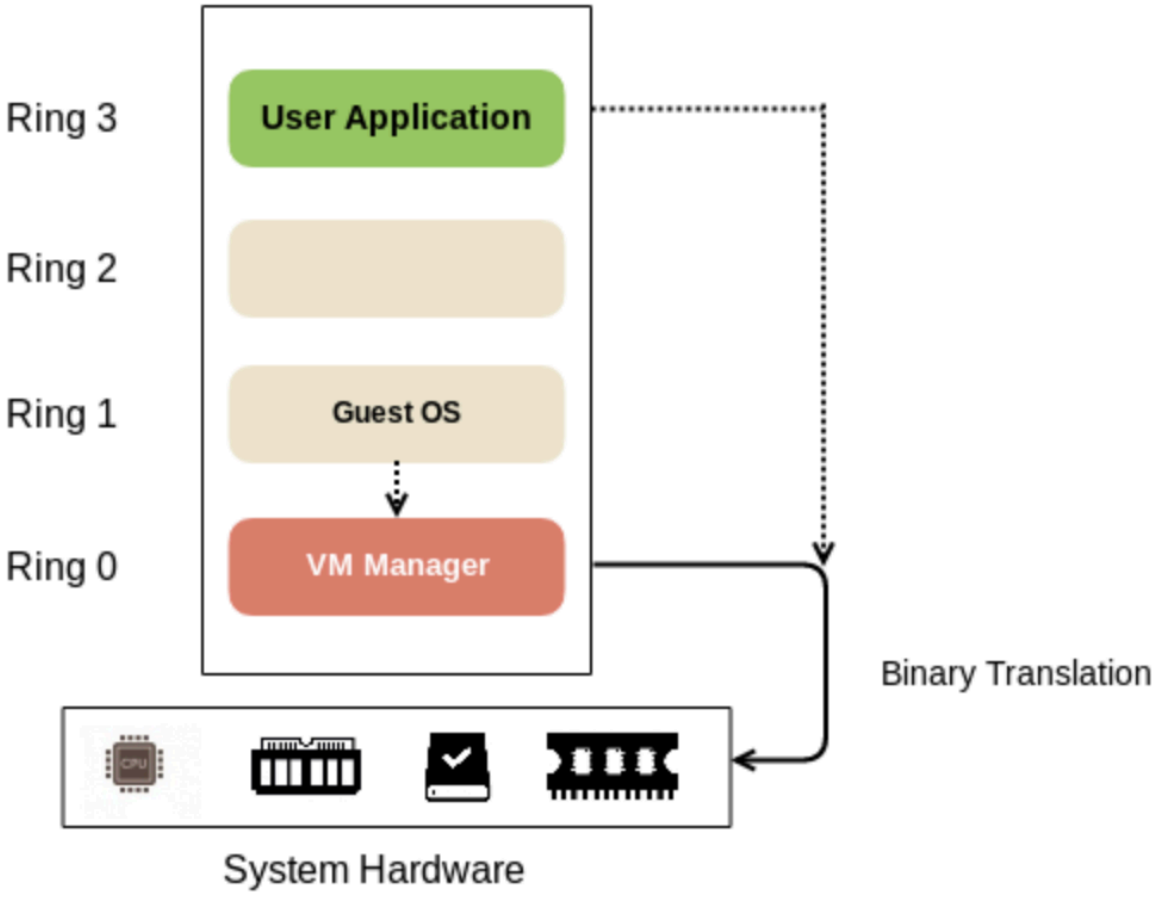


Figure 2.5.1 Binary Translation Approach

2.6 Container

Container technology, also known as lightweight virtualisation technology. The basic unit of container technology is the process of the operating system. It does not need to run a complete operating system on virtual hardware but isolates and abstracts the resources of the host to provide a relatively independent operating environment. Different ways of virtualisation lead to containers, which consume fewer resources and have higher resource utilisation. Therefore, under the same physical hardware, the number of Container is much more than the number of virtual machines. Containers can be created to start in milliseconds, much faster than virtual machines that start in minutes. Also, container technology can be used to package target applications into a component which is single address access, mirror access and simple commands are required to deploy.[21] Fundamentally simplify the deployment of applications and reduce the difficulty of deployment. Using container technology to encapsulate different applications in different container instances makes the system easier to assemble, reduces the operational risk of traditional deployment.

The essence of the Container is the process running on the host. It controls resources through Cgroup and isolates resources through Namespace. Figure 2.6.1 shows that relationship between Cgroup and Namespace. Figure 2.6.1 shows that relationship between Cgroup and Namespace

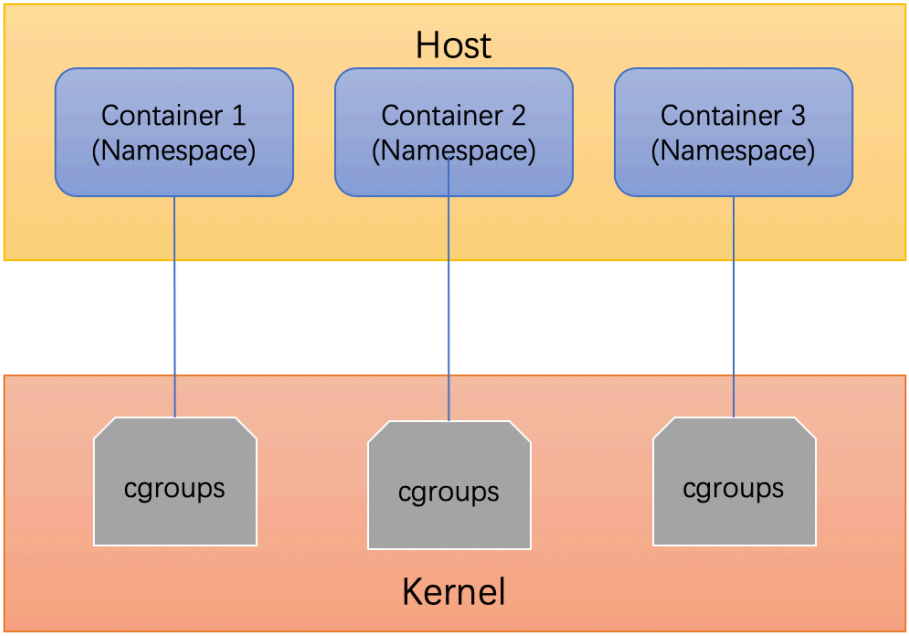


Figure 2.6.1 Relationship between Cgroup and Namespace

• Namespace

Namespace is a resource isolation method based on kernel level provided by a Linux system. There is a system function called chroot () in a UNIX system, which implements the essential isolation function and ensures the isolation of external services from the internal file system of chroot. [22] On this basis function, Linux Namespace adds six different namespaces: PID Namespace, NET Namespace, IPC Namespace, Mnt Namespace, UTS Namespace and User Namespace. Processes can communicate with each other under the same namespace, but cannot communicate external processes.

• Cgroups

Cgroups is a technology provided by the Linux kernel, which restricts, record, and separate physical resources such as CPU, memory, hard disk, which provides opportunities for container virtualisation and is the cornerstone of building a series of virtualisation management tools such as Docker.[23]

The primary purpose of Cgroups is to provide a unified interface for resource management at the different user. From resource control of a single process to virtualisation at the operating system level, Cgroups provides the following four features.

o Resource Limitation

Cgroups can limit the total amount of resources used by process groups. For example, set the maximum of used memory, issue out of memory exception once exceed this limit.

o Prioritisation

By allocating the number of CPU time slices and the size of hard disk IO bandwidth, it is equivalent to controlling the priority of the process.

o Accounting

Cgroups can count system resource usage, such as CPU usage, memory usage. It is very suitable for billing.

o Control

Cgroups can perform mount, recovery and other operations on process groups.

2.7 Virtual Machine Vs Container

• Virtual Machine

Traditional virtual machines need to simulate the whole machine, including hardware. Each virtual machine has its operating system. Once the virtual machine started, all the resources allocated to it will be occupied. Each virtual machine includes applications, necessary binaries, necessary libraries and a guest operating system.

• Container

Container technology can share hardware resources and operating systems with hosts to achieve dynamic resource allocation. Containers contain applications and all their dependency packages, but share the kernel with other containers. Containers run as separate processes in the host operating system.

Figure 2.7.1 shows that architecture of traditional virtual Machine and Docker. Docker is a kind of Container. In cloud computing, containers usually refer to Docker. This figure shows the apparent difference between Virtual Machine and Container.

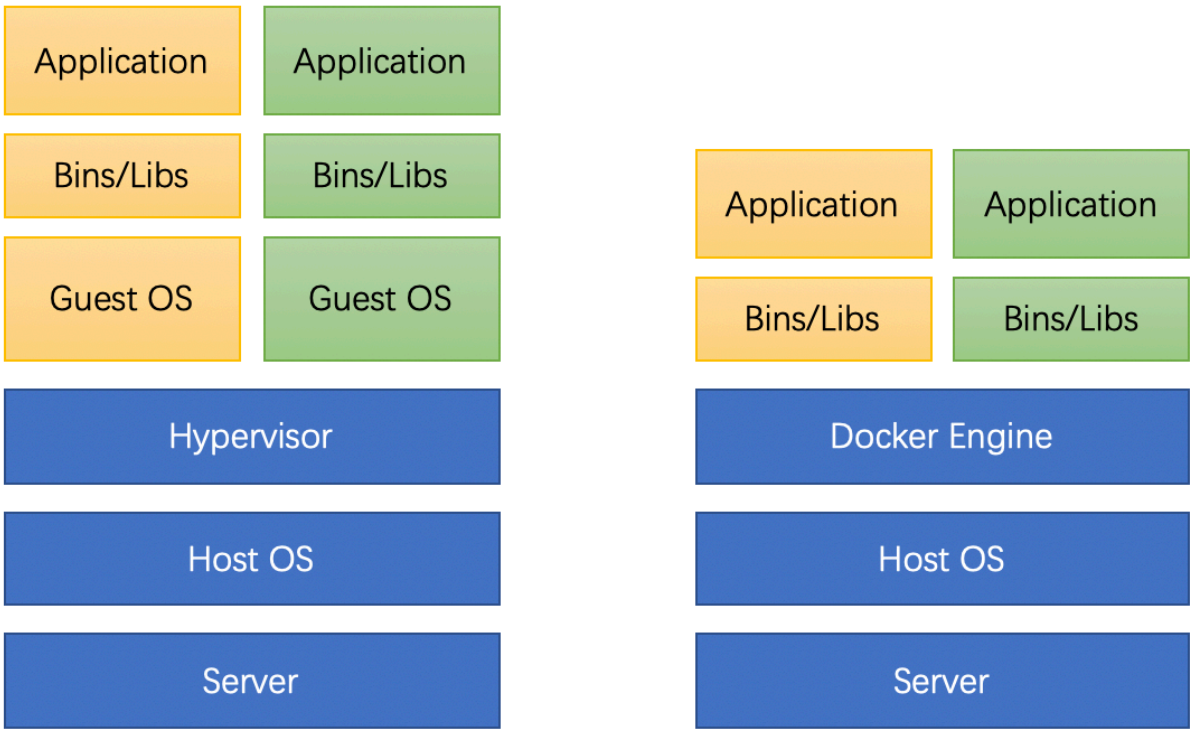


Figure 2.7.1 Architecture of VM and Container

Virtual machines and containers are deployed on hardware and operating systems. Virtual machines have a hypervisor, which is the core of the whole virtual machine. It provides a virtual operating platform for virtual machine and manages the operation system of the virtual machine. Virtual machine has its own system and system library as well as applications.

Containers do not have Hypervisor layer, and each Container shares hardware resources and operating system with the host. The performance loss caused by Hypervisor that does not exist on the Linux container. However, virtual machine technology also has its advantages, which can provide a more isolated environment for applications, and does not pose any threat to the host due to application vulnerabilities. It also supports cross-operating system virtualisation, such as running Windows virtual machines under the Linux operating system.

From the perspective of virtualisation, traditional virtualisation technology is the virtualisation of hardware resources, while container technology is the virtualisation of processes, which can provide lightweight virtualisation and achieve isolation of process and resource. From the perspective of architecture, Container removes the hypervisor layer and GuestOS layer, uses Docker Engine for scheduling and isolation, and all applications share the host operating system. As a result, Container is lighter than VM in volume, better than virtualisation in performance, and close to bare machine performance. From the application

scenario, Container and Virtual Machine have their advantages and disadvantages in software development, test scenario and production operation and maintenance.[24]

FEATURE	CONTAINER	VIRTUAL MACHINE
STARTUP	Second	Minute
HARD-DISK	MB	GB
PERFORMANCE	Near Bare Host	Less than Bare Host
ISOLATION STRATEGY	Cgroups	Hypervisor
ISOLATION LEVEL	Process	Operator System
IMAGE STORAGE	GB-TB	KB-MB
PORTABILITY	Excellent	Good
MEMORY	small	more
QUANTITY(FOR SINGAL HOST)	Thousand	Less than Hundred
ISOLATION	Safety	Safety

Table 2.7.1 Comparison of Virtual Machine and Container

- **Comparison**
  - **Startup**

Container usually takes a few seconds to start. Virtual machines usually take a few minutes to start.

  - **Performance**

Each virtual machine has its operating system. When the application is running in a virtual machine, memory usage may have a high number, and virtual machine may use up host resource. Container shares operator system environment and kernel, so Container uses fewer resources than virtual machine and reduces the stress on host memory.[25]

  - **Volume**

In the same hardware environment, the quantity of running mirrors on Container is more than the quantity of virtual machine.

  - **Isolation**

Compared with virtual machines, container isolation is weaker. Container isolation is belonged to process isolation, and virtual machine isolation belongs to operating system isolation.

- **Security**

Container security is also weaker. The guest root is the same as the host root in Container. Once user upgrade to root, it has modified host file or configuration. Virtual machine guest root is isolated from the host root.

- **Availability**

The availability of Container is re-deployment. Because Container can fast deploy the image. VM has some features which are load balancing, high availability, fault tolerance, migration and data protection. VMware has promised 99.999% high availability of virtual machines.

- **Create And Delete**

Virtual machine creation is minute-level, Docker container creation is second-level, and Container's fast iteration determines that can save much time in development, testing and deployment.

- **Delivery and deployment**

Virtual machines can achieve consistency of environment through a mirror. Docker records container building process in Dockerfile, which can achieve fast distribution and deployment in a cluster.

- **Maintain and Upgrade**

Virtual machine must upgrade or maintain each guest operating system separately. For containers, only the operating system of the container host needs to be maintained, that significantly simplifies maintenance.

Chapter 3
Project Design

3.1 Overview

This chapter presents the design of the experiments that will be performed and preparation works before project implementation. The overall workflow of the project will be discussed. The choice of frameworks for VM and Container will be explained and discuss how to select benchmark and performance metrics tools. Several possible plans for this project will be given and will explain how to determine the final plan.

3.2 Plans

This project has two ways to implement automated test application for measuring the performance overhead of virtual machines and containers. One is to run application in the background to collect data, the other is to run application in real time to crawl data. Both implementation use the same framework of virtual machines and containers and the same benchmark tool. The way of collecting performance metric data is different.

o **Running Background**

Figure 3.2.1 shows that running background structure of workflow in the virtual machine. It separate performance metrics data collection and benchmark metrics data collection. When running the project software, the project software only needs to run the script that represents the benchmark, instead of collecting data for performance metrics. Due to the performance metric tool is running in the background, so only need to implement software about benchmark, and it is easy to implement. But the performance metric will generate a large volume of data because it collects the performance of each thread in the system every second. It will make data difficult to analyze and process. To collect performance metric data in a database can solve this problem, the database will process data structurally. The benchmark data that the application runs can also be collected into the database. It is easy to analyze the data. Because only need to query the results of benchmark running in a certain period of time and the system performance metrics of this period of time in the database.

The advantage of running background is that it requires less code and only needs to write scripts for benchmark during implementation. It only needs to write some database queries to get data.

The disadvantage of running background is complicated to deploy environment. It needs to create a dedicated database and create a suitable table that can collect performance data. It

also needs to convert performance metric data to insert in the database. So it may need some applications to convert performance data.

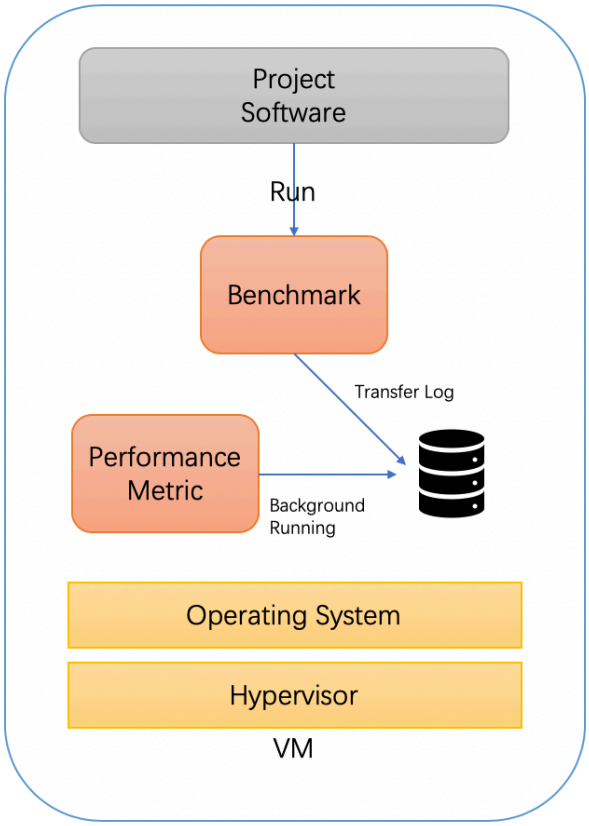


Figure 3.2.1 Running Performance Metric Tool Background

o **Running Real-Time**

Figure 3.2.2 shows that running real-time structure of workflow in the virtual machine. It synchronizes performance metrics data collection with benchmark metrics data collection. The application runs the performance metric tool and benchmark tool together. When the benchmark tool stop, the performance metric tool stops at the same time, then integrate data of both into a log folder. It only needs one period of data. The disadvantage is that earlier deployment will take more time because the application needs to include the script of collecting performance metric data.

This project decided to use running real-time. Running background will run performance metric tool in a long time, and it can cause the process of tool to be disrupted and data loss. The capacity of the hard disk is limited, if collect data all the time, it will fill up the hard disk, so that benchmark tool failed during running. Performance metric tools usually collect data per second. If the performance metric tool is running all the time, it may make data loss. For example, the performance metric tool starts to collect data on the time of 00:00:00(min:sec: mills). Project software starts the benchmark at 00:00:50. To suppose that benchmark finishes in ten seconds, that is 00:10:50. Benchmark will have ten metric data. Performance

metric tool only collected the data from 00:01:00 to 00:10:00. Because there is no benchmark tool process in 00:11:00. So the performance metric tool only has nine metric data.

For the accuracy of the data, running real-time is the best.

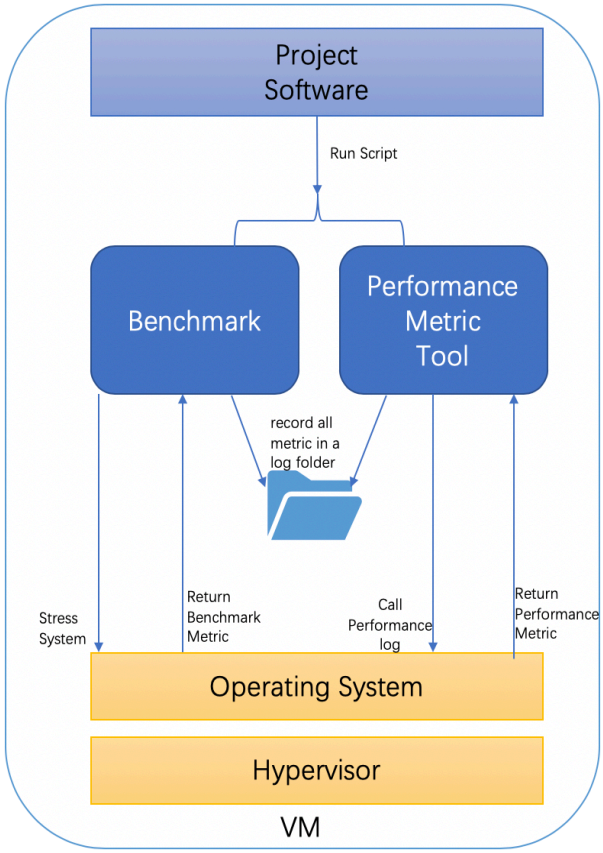


Figure 3.2.2 Running Performance Metric Tool Real Time

3.3 Architecture

This project aims to monitor and measure the performance overhead of Virtual Machine and Container, so need to measure the performance overhead of virtual machines and containers separately. Also, it also needs to measure the performance overhead of the native server. Because virtual machines and containers generally run under the native server. Virtual machine deployed by partitioning part of hardware resources on the native server. Containers are similar to a process on the native server, the hardware resources used by container are the hardware resources used by native. So virtual machines and containers are related to native. Measuring the performance overhead of native can compare the performance overhead of the virtual machine and container excellently, and it is easy to investigate the difference between the virtual machine and container.

The architecture of this project can be divided into two parts, one is the external architecture, and the other is the internal architecture. The external architecture is the software execution status from the user's perspective. It does not include the component running status and relationships of each component. The internal architecture is the internal execution state of software, which can clearly understand the tasks required and relationships for each component.

3.3.1 External Architecture

To develop the automated test program, it requires to collect performance metric and benchmark metric on native, virtual machine and container at one time. The application needs to select one of the three platforms as the primary environment, and then connect remotely or run scripts on the target platform directly to store data. It is necessary that determine which platform the software runs on. First, the container is inappropriate because running scripts on container requires remote connections to the native server and virtual machine. Second, the virtual machine is the same as the container, and it requires remote connections to native and container. So install software on native server is the best choice, because it only needs to connect virtual machine remotely, and container only needs to execute specified commands locally.

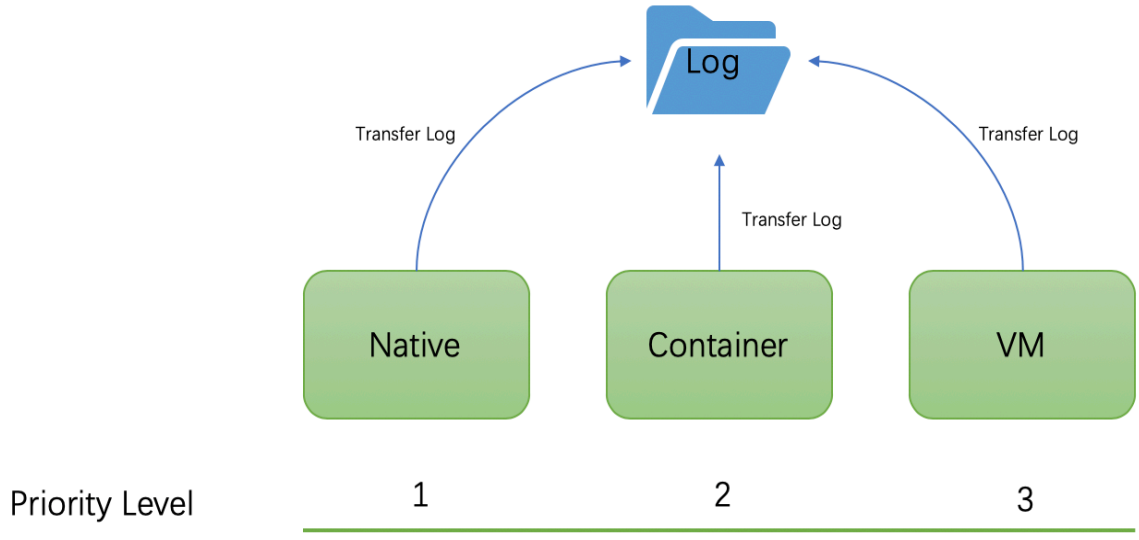


Figure 3.3.2.1 External Architecture

Figure 3.3.2.1 shows that external architecture. There are three modules which are the native server, container and virtual machine. This project set a priority level for executing them. Three of them are deploying on one physical server. If the native server and virtual machine run the workload tool at the same time, the output of the benchmark metric or performance metric is the sum value of them. Like when running CPU workload on native, it may get one hundred per cent CPU usage. But if run CPU workload on the native and virtual

machine at the same time, the CPU usage may still one hundred per cent CPU usage because the maximum of one CPU is one hundred. So it is necessary to run the native, container and virtual machine separately. When benchmark runs out, the native server and container will store the benchmark metric and performance metric in the log fold on a native server, but the virtual machine is not. Virtual machine stores benchmark metric and performance metric on local firstly, then transfer remotely to the native server log folder.

3.2.2 Internal Architecture

Figure 3.3.2.2 shows that native internal architecture. The internal architecture of the virtual machine is the same as Figure 3.2.2. Container internal architecture is similar to virtual machine internal architecture, but the container does not have hypervisor. For transferring the log file, the virtual machine will move the log file to the native server log folder. But container will store the log file in native server log folder directly, and it does not transfer log file remotely.

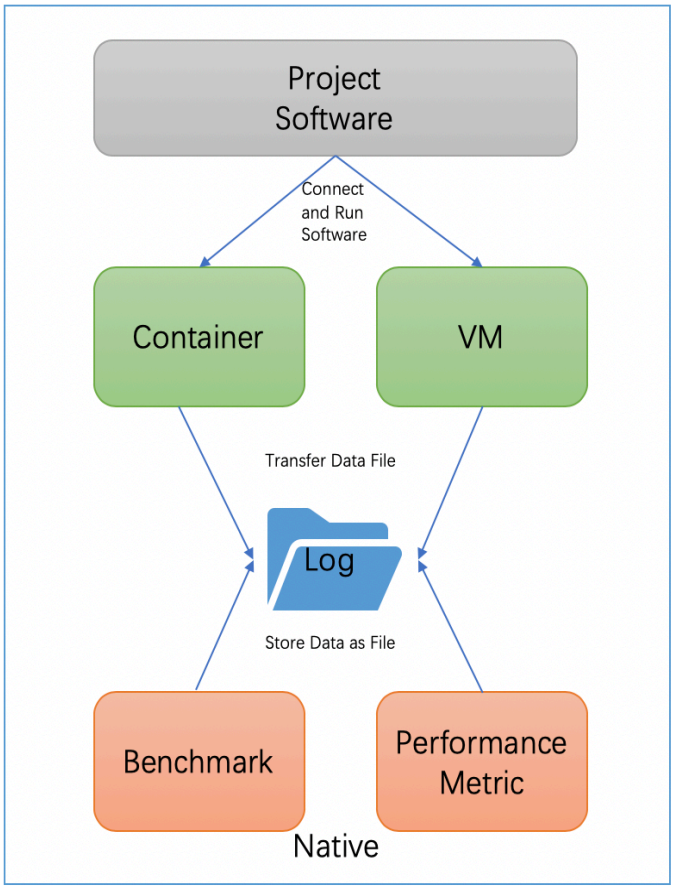


Figure 3.3.2.2 Native Internal Architecture

The general workflow of internal architecture is executing the main function of project software in native, and it will run the benchmark and performance metric tool on native firstly, then run the main function of project software in the container and then virtual machine.

3.4 Tool

There are four modules of the project need to determine the technology of them, which are a virtual machine, container, benchmark tool and performance metric tool. Each module has many alternatives. So it's hard to decide on a tool or framework. The tools or frameworks used in this project have been considered. These are all suitable for this project and can produce the best results.

3.4.1 Python

For this programming language, there are several requirements:

o Cross-Platform
Project software may installs on any platform like Ubuntu, Windows or CentOS.

o Rich Library

During development, there may be some algorithm required. Especially, the library which can use Shell. It will save much time.

o Easy to Install.

So Python is the best choice. It is a popular interpreted, general-purpose, high-level programming language. It is easy to learn, and it has enough syntactic sugar. To suppose to use C programming language to develop an application that requires one thousand lines of code in then it will need one hundred lines of code in Java, then it may has twenty lines of code in Python..

3.4.2 Qemu

Qemu is an open-source simulator and virtual machine monitor. Qemu provides two functions for users to use. First, as a user-mode simulator, dynamic code translation mechanism is used to execute code that is different from the host architecture. Second, as a virtual machine monitor, it stimulates the whole system and uses other VMMs (Xen, KVM, etc.) to use the virtualization support provided by hardware to create a virtual machine close to the performance of the host.

This project only needs to build a virtual machine for testing. To save time and convenience, Qume is used as a tool to install the virtual machine. It is simple that use Qemu to install virtual machine. After installing Qemu package, use the command line to set the parameters of the desired virtual machine, such as memory, CPU. Then use the downloaded image to

install. There are other virtual machine frameworks, such as OpenNebula and OpenStack. But they are used to build a cloud, which can build a virtual machine cluster as a cloud.

3.4.3 Docker

Container is a new virtualization technology. So it has very few frameworks, the most representative of container is Linux Container and Docker.

Figure 3.4.3.1 shows that comparison Linux Container and Docker. Linux Container is a technology of Linux kernel container virtualization, which can isolate and control resources, that is the control of Cgroup and Namespace. Docker integrates Linux Container technology and adds many components such as federated file system, libcontainer, libenetwork and so on. Docker is the advanced Linux Container, but its basic technologies are Cgroup and Namespace.

This project uses Docker to create container.

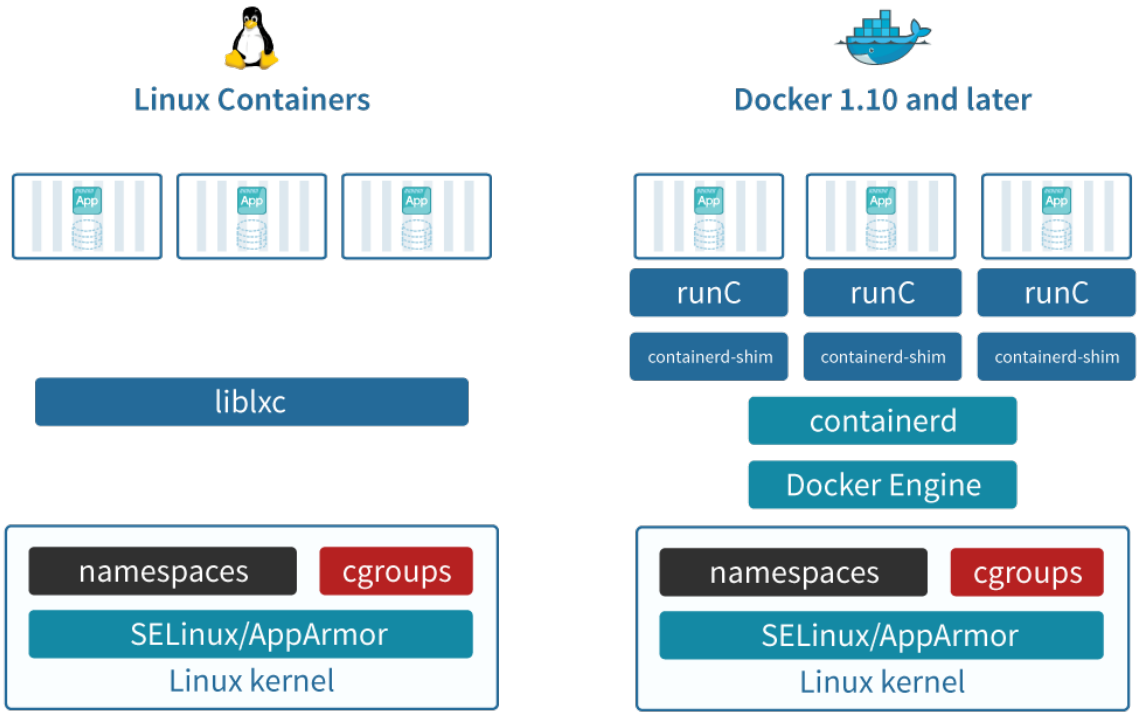


Figure 3.4.3.1 Comparison LXC and Docker [26]

3.4.4 Sysbench

Sysbench is an open source, modular, cross-platform product that enable to execute CPU, memory, disk I/O, threading, and database performance testing. Figure 3.4.4.1 shows that an example of Sysbench result.

There are many alternative performance testing tools. Stress is a traditional performance testing tools. It usually uses in Linux. In the function of testing hardware, Stress is same as Sysbench, they all can test CPU, memory, I/O. But there is a difference that Stress does not display the information during running. Sysbench shows execution time, latency and testing detail in each second.

For this project, Sysbench is better than Stress, because detail which Sysbench display will help to measuring the relative overhead between Container and Virtual Machine.

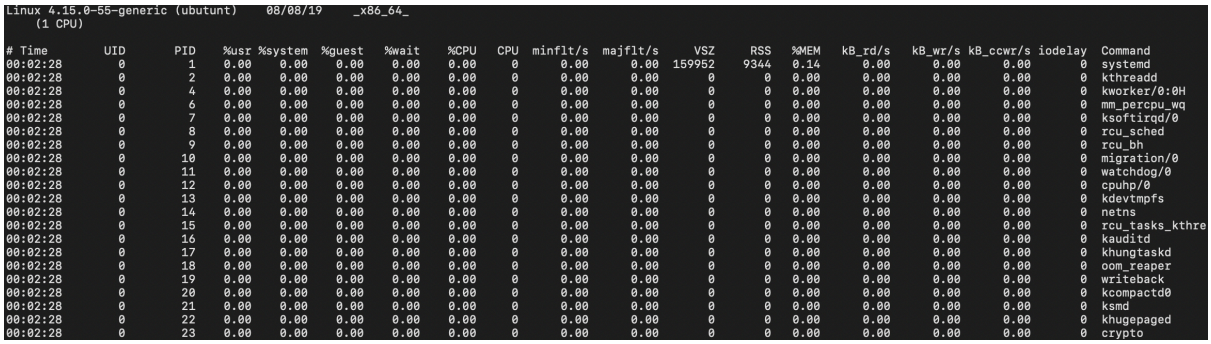
```
Prime numbers limit: 10000

Initializing worker threads...

Threads started!

[ 1s ] thds: 1 eps: 1066.41 lat (ms,95%): 1.01
[ 2s ] thds: 1 eps: 1071.40 lat (ms,95%): 0.99
[ 3s ] thds: 1 eps: 1069.09 lat (ms,95%): 0.97
[ 4s ] thds: 1 eps: 1070.17 lat (ms,95%): 0.99
[ 5s ] thds: 1 eps: 1073.61 lat (ms,95%): 0.97
[ 6s ] thds: 1 eps: 1073.68 lat (ms,95%): 0.97
[ 7s ] thds: 1 eps: 1074.63 lat (ms,95%): 0.97
[ 8s ] thds: 1 eps: 1072.28 lat (ms,95%): 0.97
[ 9s ] thds: 1 eps: 1075.56 lat (ms,95%): 0.97
[ 10s ] thds: 0 eps: 1073.62 lat (ms,95%): 0.97
CPU speed:
    events per second:  1071.91

General statistics:
    total time:                          10.0015s
    total number of events:              10722

Latency (ms):
         min:                                    0.91
         avg:                                    0.93
         max:                                    1.63
         95th percentile:                        0.97
         sum:                                 9996.12

Threads fairness:
    events (avg/stddev):           10722.0000/0.00
    execution time (avg/stddev):   9.9961/0.00
```

Figure 3.4.4.1 Sysbench

3.4.5 Sysstat

Sysstat is a software package that contains a set of tools for monitoring system performance and efficiency. It can monitor information such as CPU utilization, hard

disk and network throughput data. The collection and analysis of these data is good to measure the performance overhead of the system. Sysstat provides a set of tools for Linux performance monitoring, including sar, sadf, mpstat, iostat, pidstat, etc. This project uses pidstat to collect performance metric data. Figure 3.4.5.1 shows that the pidstat data. It can collect CPU percent, memory percent, I/O for each process of system.



```
Linux 4.15.0-55-generic (ubutunt)    08/08/19    _x86_64_
    (1 CPU)

# Time        UID       PID    %usr %system  %guest   %wait    %CPU   CPU  minflt/s  majflt/s     VSZ     RSS   %MEM   kB_rd/s   kB_wr/s kB_ccwr/s iodelay  Command
00:02:28        0         1    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00  159952    9344   0.14      0.00      0.00      0.00        0  systemd
00:02:28        0         2    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  kthreadd
00:02:28        0         4    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  kworker/0:0H
00:02:28        0         6    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  mm_percpu_wq
00:02:28        0         7    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  ksoftirqd/0
00:02:28        0         8    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  rcu_sched
00:02:28        0         9    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  rcu_bh
00:02:28        0        10    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  migration/0
00:02:28        0        11    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  watchdog/0
00:02:28        0        12    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  cpuhp/0
00:02:28        0        13    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  kdevtmpfs
00:02:28        0        14    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  netns
00:02:28        0        15    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  rcu_tasks_kthre
00:02:28        0        16    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  kauditd
00:02:28        0        17    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  khungtaskd
00:02:28        0        18    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  oom_reaper
00:02:28        0        19    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  writeback
00:02:28        0        20    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  kcompactd0
00:02:28        0        21    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  ksmd
00:02:28        0        22    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  khugepaged
00:02:28        0        23    0.00    0.00    0.00    0.00    0.00     0      0.00      0.00       0       0   0.00      0.00      0.00      0.00        0  crypto
```

Figure 3.4.5.1 Pidstat

Chapter 4
Project Implementation

4.1 Overview

In this chapter, will show the detail of deploying environment and will discuss the application structure. Each algorithm in application will be explained. Each library in application played important role. This chapter will show how native server connect virtual machine to execute and transfer data back to native server, how native server run application under docker and the detail of the entire workflow for this architecture.

4.2 Environment

	VIRTUAL MACHINE	NATIVE SERVER	CONTAINER
PROCESS	Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz		
CPU(CORE)	4	14	14
RAM	4GB	40GB	40GB
DISK	60GB	500GB	500GB
PLATFORMS	CentOS	Ubuntu 18.04	Docker v19.03.1
MONITORING TOOL	Sysstat		

Table 4.2.1 Environment

The first part of this project is deploying environment. Table 4.2.1 shows that the environment of native server, virtual machine and container.

Native Server

Native server environment is the information of physical machine. Installed Ubuntu 18.04 on it. The follow field shows the native server environment installation.

```
sudo apt-get update
sudo apt-get install docker
sudo apt-get install qemu-kvm qemu virt-viewer libvirt-bin
curl -s https://packagecloud.io/install/repositories/akopytov/sysbench/script.deb.sh | sudo bash
sudo apt-get -y install sysbench
```

Virtual Machine

Virtual machine need to use Qemu to set the configuration. To create the virtual machine, first need to create a img file, it decides the volume of hard disk of virtual machine. The following line is the img file creation about virtual machine in this project. This project used CentOS as virtual machine environment.

```
qemu-img create -f qcow2 centos.img 60G
```

The following line is to create the entire virtual machine. There are some parameters need to set, such as memory, network, hard disk and set port for virtual machine that allow the external connection.

```
qemu-system-x86_64 -m 4096 -curses -boot d -cdrom ~/Downloads/CentOS-7-x86_64-DVD-1810.iso centos.img  -net user,hostfwd=tcp::5555-:22
```

It just set the virtual machine, Sysbench and sysstat is required to install after login virtual machine. The follow field shows the detail of virtual machine environment installation.

```
curl -s https://packagecloud.io/install/repositories/akopytov/sysbench/script.rpm.sh | sudo bash
sudo yum -y install sysbench
sudo yum -y install sysstat
```

Container

Docker is easy to install on Ubuntu. It just run a command that can install.

In this project, need to create a customized image for docker. There is available docker image for Sysbench on docker hub. But those image just install sysbench, sysstat is not available in those image. So create a docker image is necessary.

To create an image need a docker file which includes some command lines. In this project, the docker image uses ubuntu as environment to run Sysbench or Sysstat. So in docker file, need to include ubuntu environment and the install command of Sysbench and Sysstat.
The follow field shows the docker file detail.

```
FROM ubuntu:latest
MAINTAINER reggiecril <reggiecril0618@gmail.com>
RUN apt-get update
RUN curl -s https://packagecloud.io/install/repositories/akopytov/sysbench/script.deb.sh | sudo bash
RUN apt -y install sysbench
```

The following URL is the docker image in docker hub.
https://hub.docker.com/r/reggiecril/sysbench
If need to run application, need to pull this image from docker hub.

```
docker pull reggiecril/sysbench
```

It will not only pull the image of sysbench, but also pull the newest version of ubuntu, because sysbench is running in ubuntu environment.
Then need to create a container that can run sysbench.

```
docker run -it --name sysbench reggiecril/sysbench bash
```

4.3 Automated Testing Application

4.3.1 Packages

During the development of this project, there are many packages used in this project. It save much time for implementation. Author does not need to build algorithm to develop.

subprocess

This package allows user to create subprocesses, connect their input, output, error pipelines, and output the return value. It is very important in this project. It can run two subprocesses , one for benchmark tool and the other for performance metric tool. Both tools can run synchronously.

os

This package encapsulates common file and directory operations. This project uses it to check the exist of file or directory. There is a method called system(). This method can execute shell command line in main process. subprocess is execute command line with subprocess, so when need a command execute in main process, this method will use.

paramiko

This package is based on SSH2 protocol, support the connection of remote server in the way of encryption and authentication. Using this package, can easily connect SSH and transfer SFTP files with SFTP protocol. Native server need to connect virtual machine for implementing automated test application. So this package is necessary.

sys

This package is same as os. Sys can run command with parameter. For this project, need to label each running of native server, virtual machine and container. It can recognize the output of each running in native server, virtual machine and container. This project decides to label each running as timestamp.

ConfigParser

This package is using to load the information of configuration file. This project uses configuration file to change the test parameter. To use this package can make author easy to load configuration file.

Matplotlib

This package is very popular, it is a tool for data visualization. After implementation, need to evaluate the overhead from the data file. Matplotlib can virtualize those data as figure, that can make author easy to analysis or evaluation.

4.3.2 Development

This project uses Python as programming language. It can give many benefits for this project, which is object orientation and cross platform. Especially object orientation, it saves much time for the implementation of this project.

This project provides configuration file. There are many parameter need to change in Sysbench and pidstat. If this project does not provide the configuration file, developer need

to change parameters inside codes. Especially once parameter need to change, then need to change the parameter of native server, container and virtual machine. It is inconvenience. So create a configuration file is necessary.

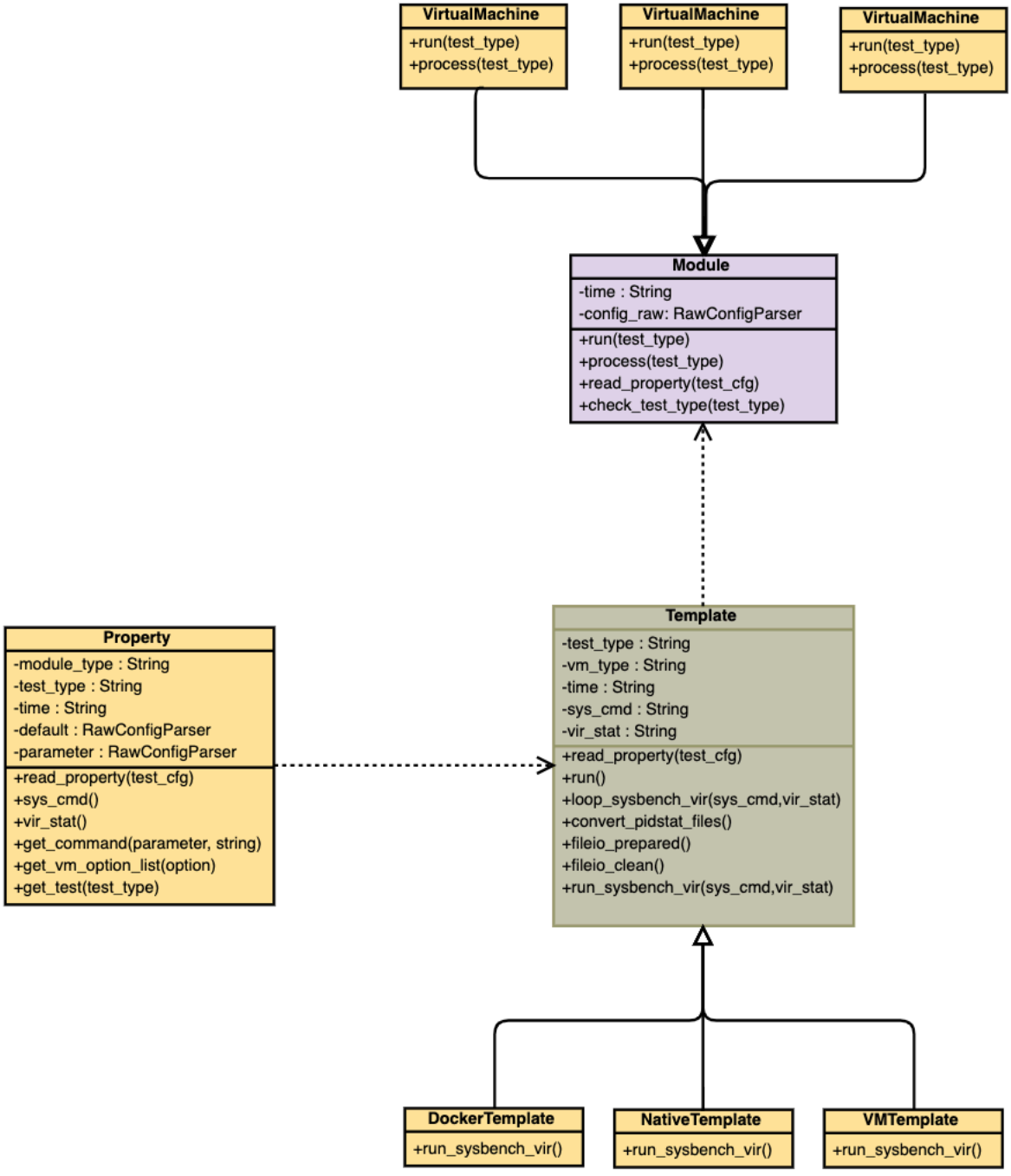


Figure 4.3.2.1 Project UML Figure

The configuration file include several set of parameters, include default parameter, CPU, memory,I/O, threads and remote connect virtual machine. Default parameter is a command which is never change. Remote connect virtual machine parameter includes virtual machine ip address, port, username and password. The other is providing for Sysbench.

Figure 4.3.2.1 shows that the figure of project UML. This project need to transform configuration file to string information. ConfigParser helps project build a class to transform. It is using in Property class, it will transform parameter as command. Like the following commands.

```
sysbench cpu --cpu-max-prime=50000 --report-interval=1 run
```

Sysbench command of native server is same as Sysbench command of virtual machine. But docker command is difference, it likes the following command:

```
docker exec -it  sysbench sysbench cpu --cpu-max-prime=50000000 --report-interval=1 run
```

The transformed string will send to template. Template is a class which run Sysbench command and pidstat command. Due to there are differences in native, virtual machine and container. So there are three classes extend template. This part is the main section of this project.

To collect data, need to run Sysbench and pidstat synchronously. The package subprocess can create a sub process to run Sysbench, pidstat will start when Sysbench start and be killed after Sysbench stopped. Data collection can use subprocess output pipeline for Sysbench and pidstat will generate a file.

In this section, set a function to loop each running. It solves the problem which little data can not analysis.

After data collection, data will be recorded as string format. It will send to module class, it is using to save data as file. The path of native, virtual machine and container are difference, so need to create three classes to extend module.

There are some tool classes include data integration and virtual machine connection remotely. There is a class encapsulated to support to integrate Sysbench data and pidstat data as list or dictionary. There is a class encapsulated to support to connect virtual machine, it requires the configuration file.

4.3.3 Test Parameter

For the accuracy of data, this project used five set of testing parameter. The value of set is from small to large by the number of set. It can collect the elastic and maximum work load of system. Table 4.3.3.1 shows that detail of test parameter.

	CPU	MEMORY	I/O	THREADS
1	cpu-max-prime = 5000	memory-block-size=1K	file-block-size=1000	thread-yields=1000

		memory-total-size=100G	file-total-size=2G file-test-mode=rndrw	thread-locks=8
2	cpu-max-prime = 50000	memory-block-size=1K memory-total-size=100T	file-block-size=1000 file-total-size=20G file-test-mode=rndrw	thread-yields=2000 thread-locks=20
3	cpu-max-prime = 500000	memory-block-size=1M memory-total-size=100G	file-block-size=10000 file-total-size=2G file-test-mode=rndrw	thread-yields=4000 thread-locks=40
4	cpu-max-prime = 5000000	memory-block-size=1M memory-total-size=100T	file-block-size=10000 file-total-size=20G file-test-mode=rndrw	thread-yields=6000 thread-locks=60
5	;pu-max-prime = 50000000	memory-block-size=2M memory-total-size=100T	file-block-size=100000 file-total-size=50G file-test-mode=rndrw	thread-yields=10000 thread-locks=100

Table 4.3.3.1 Test Parameter

Chapter 5
Results And Evaluation

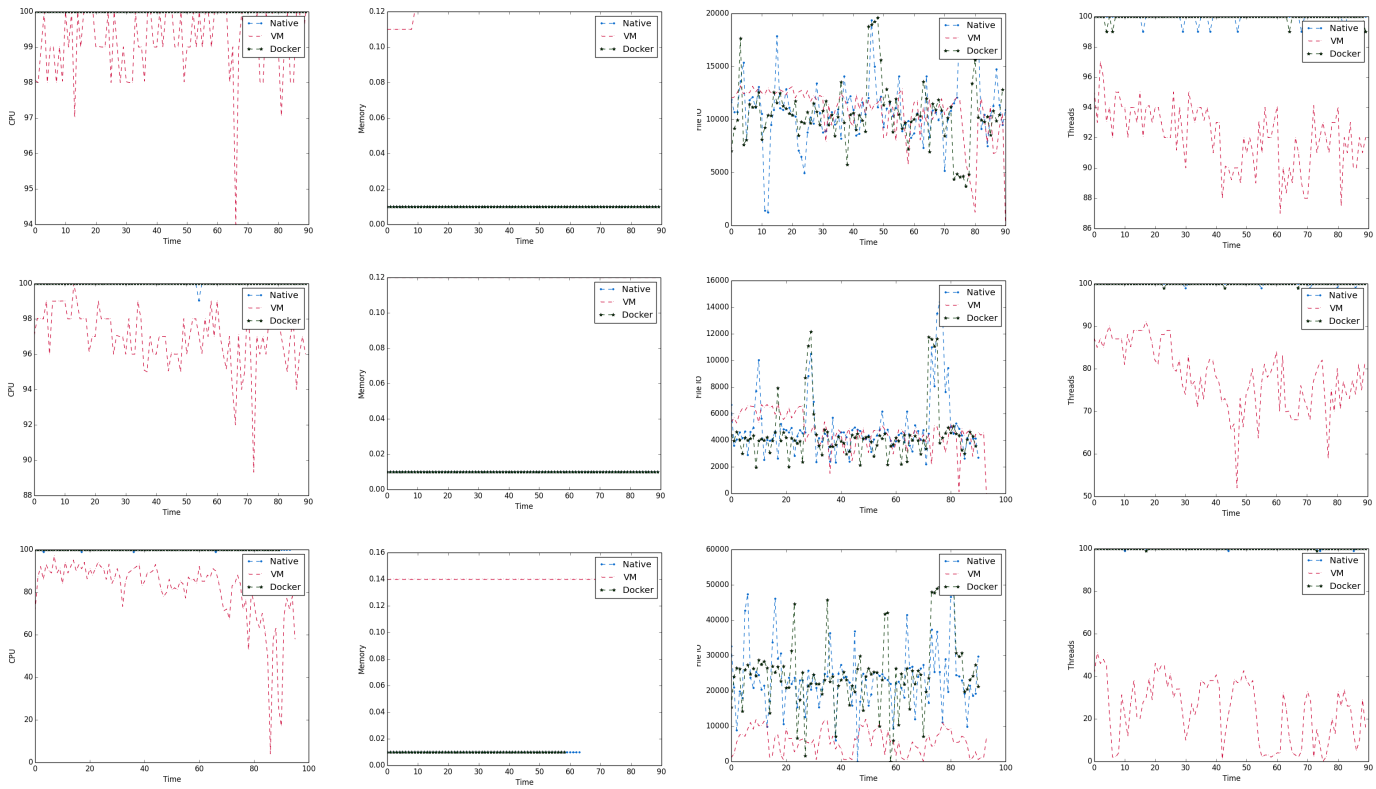
5.1 Overview

This Chapter will evaluate this project. Methodology will evaluate in this chapter. There are many figures to evaluate the comparison between virtual machine and container. To evaluate the performance and overhead between container and virtual machine.

5.2 Methodology Evaluation

This project uses Agile methodology. During implementing project, Agile methodology provides many benefits and saves much time for project. Agile methodology provides high fault tolerance. During implementing, the inital performance metric tool of docker is docker stat. But in some reason, it can not run with Sysbench at same time. Because use Agile methodology, this project can change and test the performance metric tool easily. In Waterfall methodology, it will cost much time.

5.3 Performance Evaluation



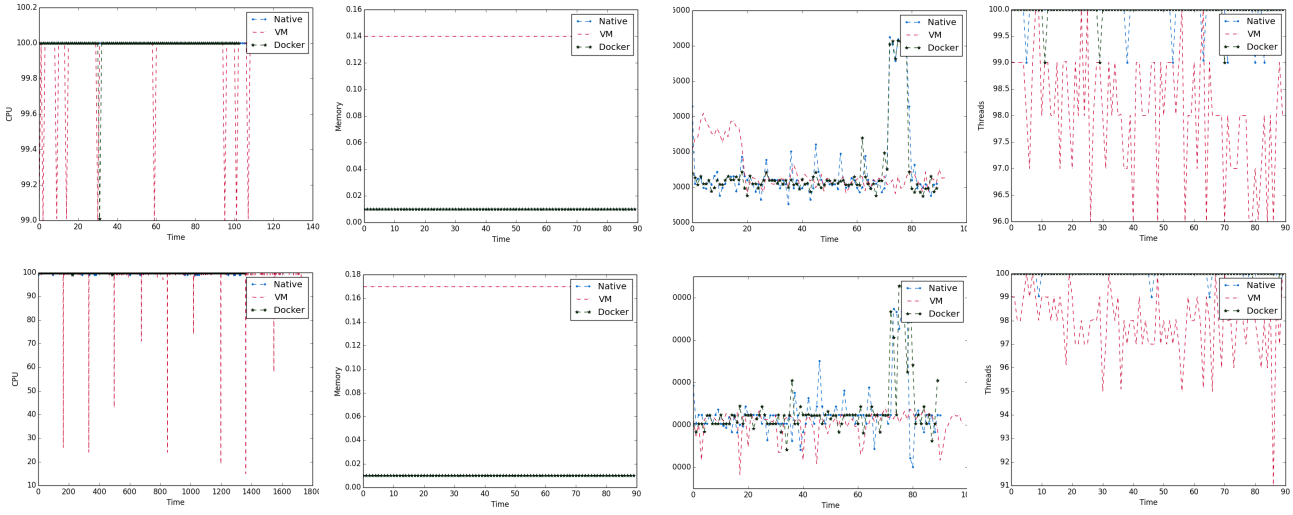


Figure 5.3.1 Set of performance metric figure

As can be seen from Figure 5.3.1. Overall, the performance of native and docker is similar, but there is a significant difference between docker and virtual machine. This figure is ordered, it sorted form small parameters to larger parameters and each column is a test. Each test include CPU, memory, I/O and threads.

In this figure, docker and native are around the peak line in CPU figure, virtual machine shows an unstable state. With the increase of parameter, it is increasing to peak line slowly. In memory figure, we can't see anything useful. Sysbench may not very support memory test, only CPU percent increase when execute memory test. Perhaps Sysbench puts too little workload on memory, so that memory percent of native and docker is close to zero, while memory percent of VM is always above 0.1, and it seems that it increases with the increase of workload. I/O is quite special. Native and docker are equally efficient, but virtual machine is similar to native. It is smaller than docker and native in the third test.

5.4 Overhead Evaluation

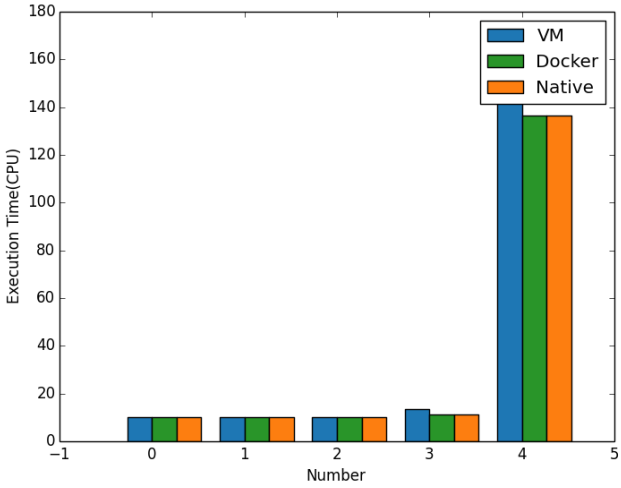


Figure 5.4.1 CPU Overhead

Execution time is a way to measure the overhead of performance. In Figure 5.4.1, there is obvious comparison after three times. First three tests got similar overhead for native, virtual machine and docker. Benchmark is calculating maximum prime to make the workload. So when set small maximum prime, three of them can calculate fast, that cannot show the comparison each other. In fourth test, native overhead is same as docker, and virtual machine overhead is 20% higher than docker overhead. In fifth test, virtual machine overhead is 15% higher than docker overhead.

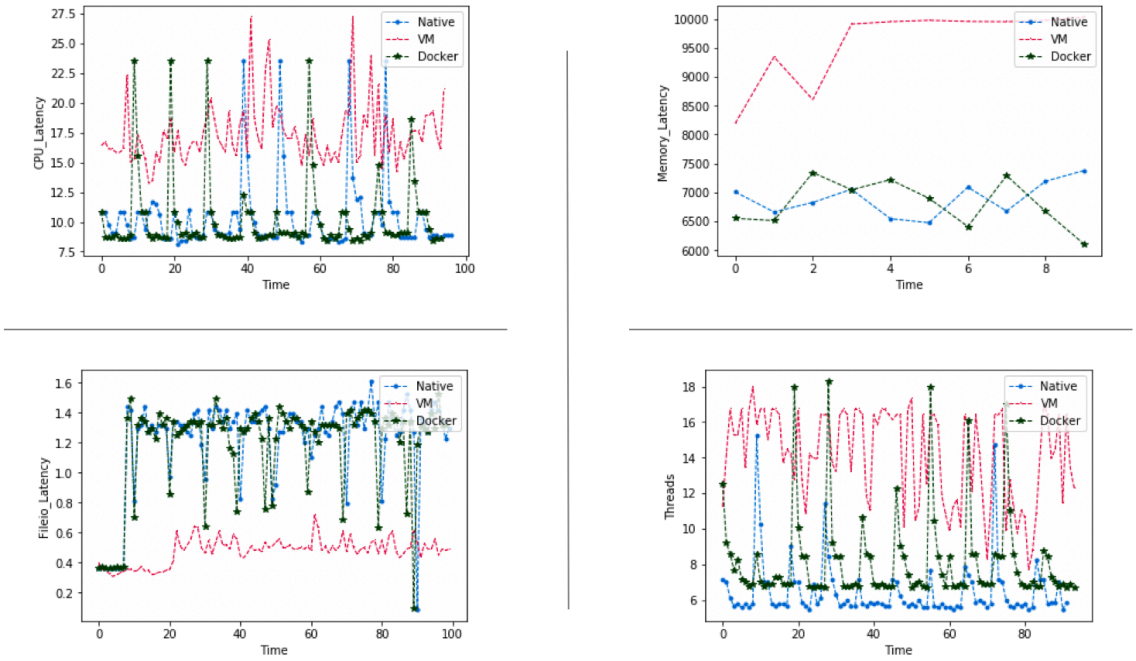


Figure 5.4.2 A Set Of Overhead for each test.

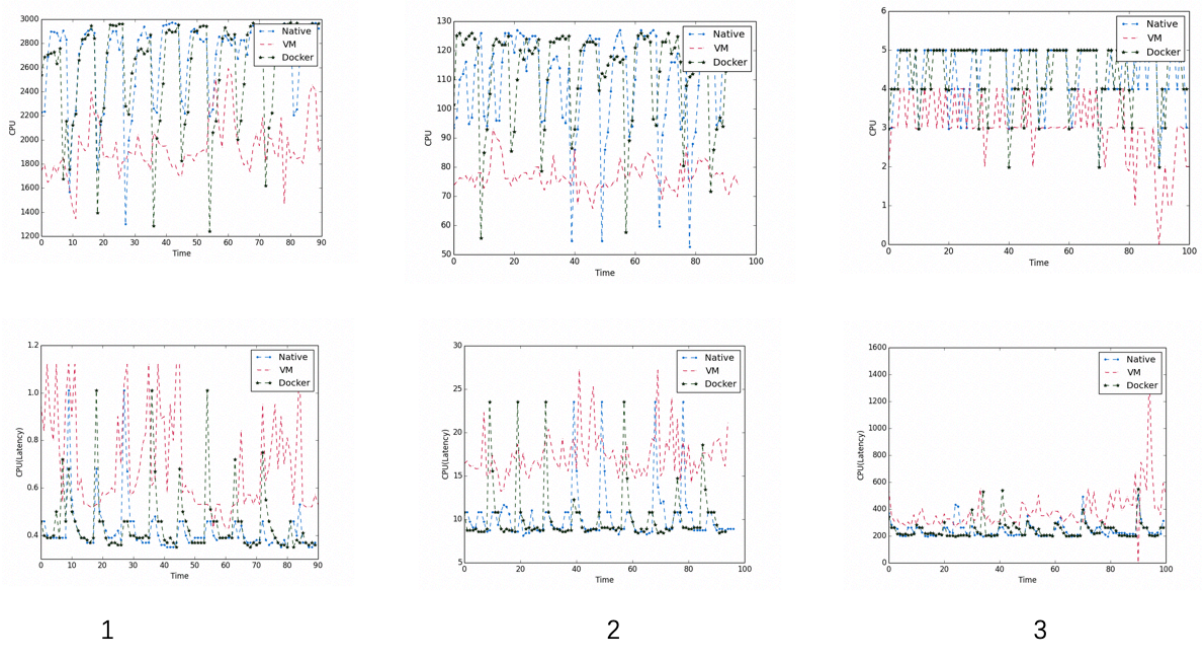
In Figure 5.4.2, used latency to measure the overhead. This set overhead shows clearly comparison for virtual machine, native and docker. This set is suitable the research in background. Virtual machine performance overhead clearly shows on this figure, it is much higher than native and docker. The following table is the comparison rate, it based on the overhead native to compare docker overhead and virtual machine overhead.

	VIRTUAL MACHINE	DOCKER
CPU LATENCY %	76%	2%
MEMORY LATENCY %	78%	0.3%
FILE I/O LATENCY %	-130%	0%
THREAD LATENCY %	164%	23%

Table 5.4.1 Overhead Comparison

At most time, virtual machine have higher overhead than native and docker, but there is different in file I/O test. Because virtual can cache any virtual machine storage, hypervisor helps it to run faster than native, so virtual overhead is lower than native and docker.

CPU



1 2 3

Figure 5.4.3 CPU eps and latency

Figure 5.4.3 shows the CPU test detail. The upper figures is figures of eps and bottom figures is figures of latency. The interesting of figures is figures of eps is gradually decreasing, and figures of latency is gradually increasing. In figures of eps, virtual machine

always in the bottom of docker and virtual machine always in the top of docker in figure of latency. It proves the performance of virtual machine is worse than docker in CPU.

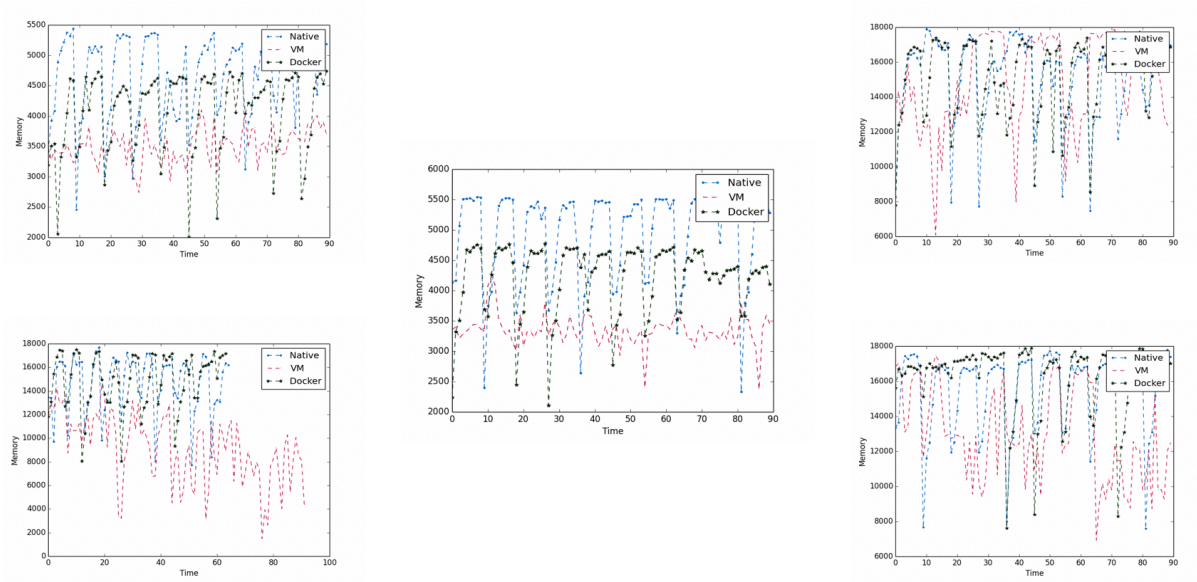


Figure 5.4.4 Memory Usage

Figure 5.4.4 shows that memory usage during memory test. The left figures and middle figures is small parameters memory test. It displays the difference between docker and virtual machine. But on the right of figures, it shows obscure comparison, virtual machine and native are unstable. On the whole figures, virtual machine has lower memory usage than docker.

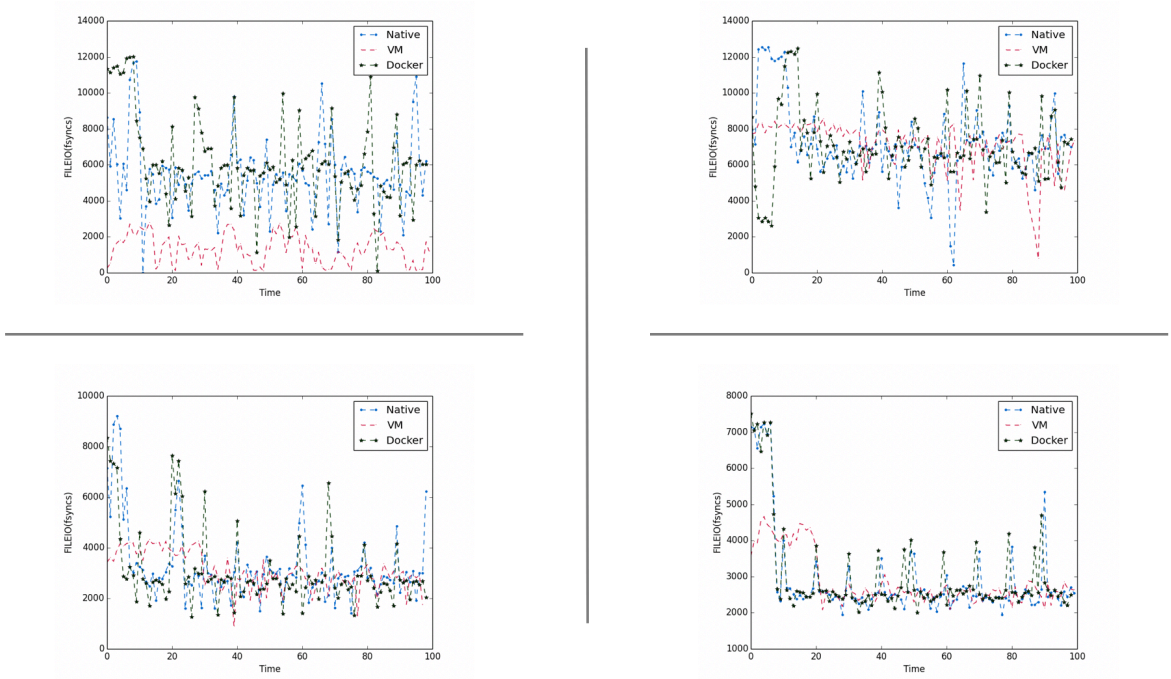


Figure 5.4.5 File I/O fsyncs

Figure 5.4.5 shows file I/O fsyncs. There is no obvious difference between docker and virtual machine.

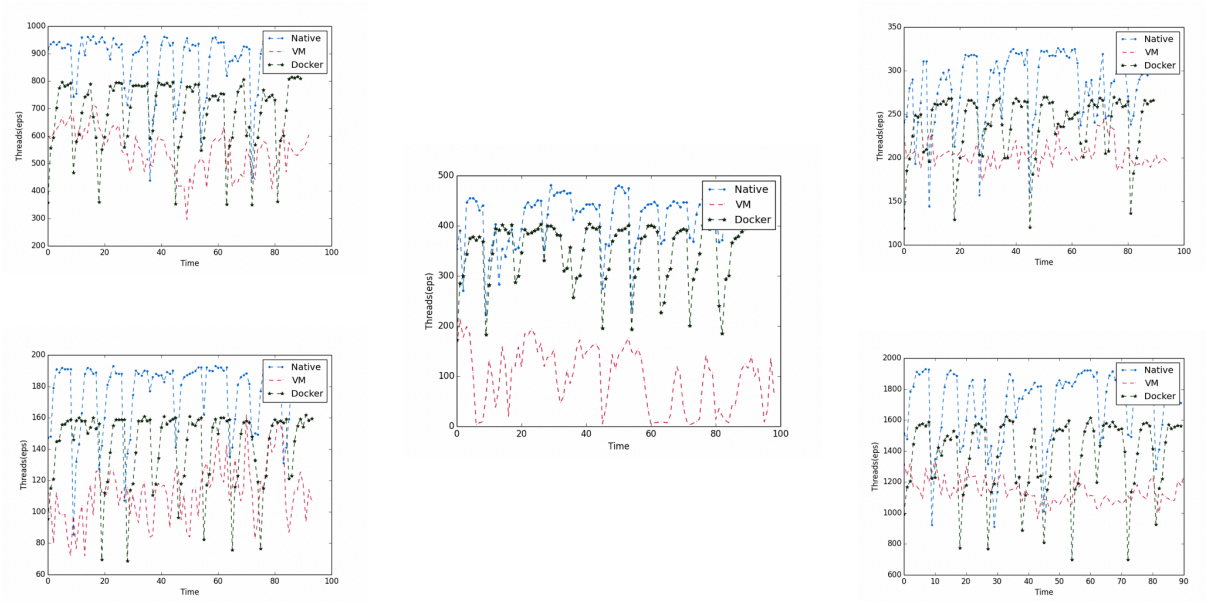


Figure 5.4.6 Threads eps

Figure 5.4.6 shows that clearly comparison between virtual machine and docker.The eps of Virtual machine is around 20% to 60% of the eps of docker, and the eps of docker is around 80% to 90% of native.

Chapter 6
Conclusion

5.1 Overview

This project aims to implement an automated test application which can measure the relative performance overhead of virtual machine and container. This chapter evaluates the success of this project, point the shortcoming of this project and discuss the future work which this project does not touch or can make this project more advanced. Personal reflection is available, and it includes author feeling about the implementation of this project, the discussion author shortcoming during implementing and author learn during planning or designing this project..

5.2 Project Accomplishment

Investigated the background of this project. Understanding the relative technologies and architecture of this project. Researched the workflow of this project and the principles of related technologies. Though the comparison of virtual machine and container, understanding the advantages and disadvantages of virtual machine and container..

Successful for deploying the necessary environment, evidencing the success of the initial project plan. Virtual machine and container can running software perfectly, it also evidence the cross-platform feature of Python.

Built a configuration file. Use can execute the different test by changing the parameter in the configuration file, instead of changing code in the software. And every time run the main function will execute CPU test, memory test, I/O test, threads test together, that will reduce the cost of time.

This project built a section which can iterate each test, and it solves a problem which is data inaccuracy because there is less data if just run the test once.

Achieved the aim of the project. It is successes to build an application which can test performance overhead automatically. Only run the main function of the application in a native server, which can collect data automatically that including the performance metric and benchmark metric of native server, virtual machine and container. And data of three of them will save at the same folder, which will give data analysis convenience.

From background research, we know that container has better performance than a virtual machine. After this experiment, the result shows that it is correct which container has better performance than a virtual machine. But in the test of I/O, the virtual machine has better than container.

5.3 Project Limitation

Due to the native server connect virtual machine by the network. So the network will influence the connection between the native server and virtual machine. Once network latency happened, it will cause virtual machine transfer data.

Different operation system may lead to the conversion of data failed because the data conversion section of this project is aimed at Linux. I have never tried this on another operating system. But the information of pidstat is different from Ubuntu..

If one test in application failed, that would lead to the application interrupt. For example, suppose that the hard disk of the virtual machine is full. When software is running I/O test in the virtual machine, the I/O test will stop because the hard disk is full. Then recall to the native main function, it will throw our exception to exit. The next test of I/O test in a virtual machine will not execute.

It is not Compatibility of some operation system for the installation of Qemu. For example, running the same Qemu command line, it is Successful for the host server of the university, but it will fail to install a virtual machine in my laptop.

When the volume of collected data is enormous. The operation efficiency of the application may be very slow. The algorithm of data conversion hasn’t optimized because the author believes the data from ten iterations of test can achieve the requirement. If the data is from over 100 iterations test, that will run very slow in data conversion.

This project just test CPU, memory and I/O. There is network can be tested. It can test the network latency of virtual machine and container.

5.4 Future Work

Qemu is an old technology. Most organizations or companies is building cloud platform by OpenStack or OpenNebula. They provide more function for virtual machine, they also optimize virtual machine.

This project just created a virtual machine and a container because of time limitation. If we want an advanced level for comparing virtual machine and container, the cluster is available. Cluster technology is outstanding nowadays. There are many popular frameworks, such as OpenNebula and OpenStack for the virtual machine, Kubernetes for container. Using cluster technology, you can not only measure the performance overhead of virtual machine and container, but also test the relative performance overhead between clusters.

Sysbench only support CPU test, I/O test, memory test and thread test. There is no network test in Sysbench. If can test the network for virtual machine and container, that will have a better comparison.

When the workload is low, the performance overhead of the virtual machine and container are the same, and then the virtual machine is the best choice for the user because of its isolation and security. To find the critical point is essential for user choice. Machine learning is a popular technology, that can predict the action of data. Using machine learning can find the critical point easily.

5.5 Personal Reflection

My many technical skill is enhanced in this project. I was using Java programming language before this project. I have never touched Python. But after implementing this project, I realize the benefits of Python, especially his easy to learn and rich syntactic sugar, that reduces the time of implementation. Due to the time constraints of the project, it is challenging to plan the design and tasks of the project. I have to give enough time and meet deadlines between tasks. It gives me a great challenge to project management. It poses a great challenge to my project management ability. The project started in February, but I still have modules and exams that need to take time, so I need good time management to balance the time between modules and projects. Academically, the project gave me a good understanding of problem statement and critical thinking. Let me have a full understanding of big data and cloud computing.
During the development of the project, the biggest challenges is to collect performance metrics data and benchmark metrics data simultaneously. I thought it would be easy to solve this problem with multithreading, but in experiment, some performance metric tools cause delays when collecting system data, such as docker stat –no-stream. This will lead to data loss. This requires replacing the performance metric tool, and finally using pidstat for performance metric data collection.

In the early stages of design, experienced supervisors can be interviewed and discussed. Because they are experienced, they can give more advanced ideas and design decisions, and they can make the best design according to their experience.

List of References

[1] Kulkarni, G.,Gambhir, J. and Palwe,R. (2011). Cloud Computing-Software as Service. International Journal of Computer Science & Information Technology Research Excellence. Vol. 2.

[2] Rouse, M. (2016). *What is a Virtual Machine and How Does it Work? - Definition from WhatIs.com*. [online] SearchServerVirtualization. Available at: https://searchservervirtualization.techtarget.com/definition/virtual-machine [Accessed 5 Aug. 2019].

[3] Kannan, V., Jhajharia, S. and Verma, S. (2014). Agile vs waterfall: A Comparative Analysis. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 3(10), pp.2680-2686.

[4] Sharma, S., Sarkar, D and Gupta, D. Agile Processes and Methodologies: A Conceptual Study. International Journal on Computer Science and Engineering. 2012

[5] Mell, P. and Grance, T. (2011). *The NIST Definition of Cloud Computing*. [ebook] National Institute of Standards and Technology. Available at: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf [Accessed 15 Aug. 2019].

[6] Gour, R. (2019). *9 Major Characteristics of Cloud Computing*. [online] DZone. Available at: https://dzone.com/articles/9-major-characteristics-of-cloud-computing [Accessed 15 Aug. 2019].

[7] Eugene,G. (2013) *Cloud Computing Models*. [ebook] Massachusetts Institute of Technology Cambridge, MA. Available: http://web.mit.edu/smadnick/www/wp/2013-01.pdf [Accessed 15 Aug. 2019].

[8] Stephen,W and Muhammad,R. (2019) saas-vs-paas-vs-iaas, digital image,bmcblogs, viewed 10st August 2019,<https://blogs.bmc.com/wp-content/uploads/2017/09/saas-vs-paas-vs-iaas.png>

[9] Lee, H. (2014). *Virtualization Basics: Understanding Techniques and Fundamentals*. [ebook] pp.1-4. Available at: http://dsc.soic.indiana.edu/publications/virtualization.pdf [Accessed 17 Aug. 2019].

[10] Lanigan, R. (2015). *Reducing energy costs with virtualization and automation*. [online] SearchServerVirtualization. Available at: https://searchservervirtualization.techtarget.com/tip/Reduce-power-consumption-to-save-money [Accessed 17 Aug. 2019].

[11] DataFlair. (2018). *Hardware Virtualization in Cloud Computing - Working, Types, Benefits - DataFlair*. [online] Available at: https://data-flair.training/blogs/hardware-virtualization-in-cloud-computing/ [Accessed 17 Aug. 2019].

[12] Daniel,J,M and Thomas,W,C.(2004) Optimised paravirtualisation for the Itanium processor family. In 3rd USENIX-VM, pages 73–82.

[13] Smith,J and, Nair R.(2005) The architecture of virtual machines. IEEE Computer. pp.32−38.

[14] Bauman, E., Ayoade, G. and Lin, Z. (2015). A Survey on Hypervisor-Based Monitoring. *ACM Computing Surveys*, 48(1), pp.1-33.

[15] Ankita,D., Rachana,O., Pratik,S. and Bhautik,P. (2013). Hypervisor: A Survey on Concepts and Taxonomy. *International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075*, 2(3), pp.222-223.

[16] Radack,S.(2011) Full Virtualization Technologies: guide for secure implementation and management. ITL Bulletin , pp.2-3.

[17]Zach,A., Daniel,A., Daniel,H., Anne,H. and Pratap,S.(2006) VMI: An interface for paravirtualization. In Ottawa Linux Symposium. Citeseer.

[18] Laadan,O. and Nieh,J.(2010).*Operating System Virtualization: Practice and Experience*. ACM Proceedings of the 3rd Annual Haifa Experimental Systems Conference. New York. pp.1-12

[19] Jeanna,N,M., Wenjin, H., Madhujith,H., Todd,D., Demetrios,D., Gary,H., Michael,M, and James,O.(2007) Quantifying the Performance Isolation Properties of Virtualization Systems, ACM Workshop on Experimental Computer Science (ExpCS)

[20] Mathias,P.and Thomas,G.(2009) *Fast Binary Translation: Translation Efficiency and Runtime Efficiency*.In: 2nd Workshop on Architectural and Microarchitectural Support for Binary Translation

[21]Felter, W., Ferreira, A., Rajamony, R., Rubio, J.(2014) An updated performance comparison of virtual machines and linux containers. Technology **28**, 32.

[22]Grunert, S. (2019). *Demystifying Containers - Part I: Kernel Space*. [online] Medium. Available at: https://medium.com/@saschagrunert/demystifying-containers-part-i-kernel-space-2c53d6979504 [Accessed 18 Aug. 2019].

[23]Lucas,C., Prateek,S., Prashant,S. and Y.C. Yay,(2016) *Containers and virtual machines at scale: a comparative study*. Middleware '16 Proceedings of the 17th International Middleware Conference, Trento, Italy

[24] Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L. and Zhou, W. (2018). A Comparative Study of Containers and Virtual Machines in Big Data Environment. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 178-185.

[25] Maheshwari, S., Deochake, S., De, R and Grover,A. (2018). Comparative Study of Virtual Machines and Containers for DevOps Developers.

[26] Adeesh,F.(2017) linux-vs-docker-comparison-architecture-docker-lxc, digital image,ROBIN, viewed 18st August 2019,<https://robin.io/wp-content/uploads/2017/03/linux-vs-docker-comparison-architecture-docker-lxc.png>

<It is expected that the list would reflect the breadth and depth of scholarly research undertaken by the student during the course of the project.>

Appendix A
External Materials

This project provides all external materials in GitHub. It includes user guide, codes, configuration file and so on. The following is the GitHub address:
https://github.com/Reggiecril/Performance