東海大學

資訊工程研究所

碩士論文

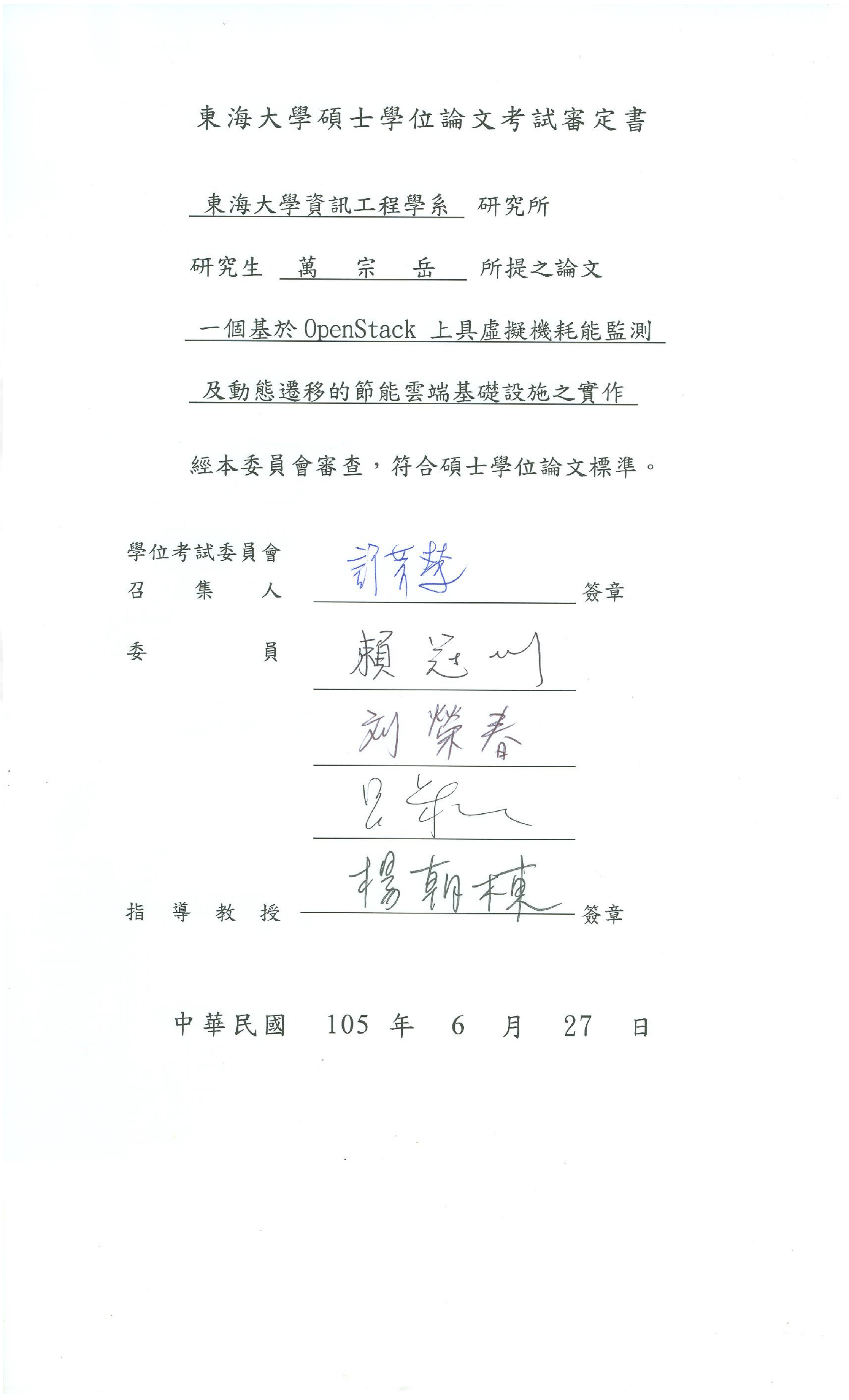
指導教授: 楊朝棟博士

一個基於 OpenStack 上具虛擬機耗能監測及動態遷移的節能雲端基礎設施之實作

Implementation of an Energy Saving Cloud Infrastructure with Virtual Machine Power Usage Monitoring and Live Migration on OpenStack

研究生: 萬宗岳

中華民國一零五年六月



東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 萬 宗 岳 所提之論文

一個基於 OpenStack 上具虛擬機耗能監測

及動態遷移的節能雲端基礎設施之實作

經本委員會審查，符合碩士學位論文標準。

學位考試委員會 召 集 人	[illegible]	簽章
委 員	賴冠州	
	劉榮春	
	[illegible]	
指 導 教 授	楊朝棟	簽章

中華民國 105 年 6 月 27 日

摘　要

雲端運算是基於網路上，而且需要不少的實體機，進而消耗大量的電能，這種情況會減少雲端服務提供者的利潤，而且對整個雲端運算叢集也會有傷害。雲端運算所消耗的電量是近年來大家注重的議題，當在進行大量的計算時，消耗的電量可不容小覷，如何達到最高的效能與最低的功耗是現在各界都在追求的。虛擬化在現今被廣泛的使用，但現有的虛擬機電量優化方法都無法對於不同規格下的實體機產生很好的作用。本論文實作出一個可以即時監測 OpenStack 的狀態與監測 OpenStack 上的虛擬機即時狀態以及透過動態遷移達到節能的雲端基礎設施。監測的項目包含 CPU 的使用率、記憶體的負載、以及耗電量，都是以即時的方式顯示的，完整的監測實體機與虛擬機的即時狀態。並記錄每台實體機的 CPU 使用率與耗電量，且呈現在此雲端基礎設施上，提供使用者當作實驗依據的參考。再來是以監測到的電量為基礎，透過動態遷移的方法來自動調配各個實體機上的虛擬機，使各個實體機的耗電量得以平衡，既可以避免資源的閒置與浪費，也可以避免因實體機持續的高功耗導致機器壽命減少，進而達到節能的目的。

關鍵字: 即時監測、節能、機器監測、動態遷移、動態調配、雲端基礎設施

Abstract

Cloud computing is a kind of Internet-based computing, and requires more physical machines and consumes an large amount of power, that will reduce the profit of the service providers and harm the environment. Power consumption of cloud computing is an issue we focus on in recent years, when making a large number of operations, the power consumption cannot be underestimated. Virtualization is widely used in cloud computing nowadays. However, existing energy efficient scheduling methods of virtual machines (VMs) in cloud cannot work well if the physical machines (PMs) are heterogeneous and their total power is considered. This thesis implement a cloud infrastructure that can monitor the status of OpenStack and monitor the real-time status of virtual machine on OpenStack then achieve to energy saving through live migration. The projects of monitoring include the utilization of CPU, load of memory, and power consumption. These data show in real-time, completely monitor the real-time status of physical machines and virtual machines. It also record the utilization and power consumption of physical machines then show on this cloud infrastructure, to provide experimental evidence for the user as a reference. Base on the power consumption we monitoring, we can automatically allocate virtual machines on every physical machines by live migration, to balance the power consumption of every physical machines. It' s not only can avoid idle and waste of resources but also can avoid reducing machine life because of the physical machines always keep in high usage, and achieve to power saving.

Keywords: Real-Time Monitoring; Power Saving; Machine Monitoring; Live Migration; Dynamic Allocation; Cloud Infrastructure

致謝詞

研究所兩年的生活，所經歷的每個時刻至今仍歷歷在目，快樂的、難過的與挫折的，有的時候希望時間是過的快一點的，讓我直接進入人生的下一個階段，但在經歷過這些磨練之後，著實的感覺到自己更不一樣了，不管是在面對事情的態度或是對於專業科目上，這兩年的磨練讓我覺得自己已經變得更加成熟。

我要感謝我的研究所指導老師楊朝棟教授，在我目標不明確時，給了我最佳的的建議與方向，還有提供許多設備讓我做起實驗來更加順手，沒有了老師提供的設備，我的實驗肯定會困難重重。也感謝老師讓我去中國參加 PAAP 國際會議，學習新事務，拓展國際視野。

另外，感謝抽空前來參加論文口試的委員們，謝謝系上劉榮春老師對我的研究提供了很多的意見、指導和鼓勵。謝謝呂芳懌老師、賴冠州老師及許芳榮老師，給了我很多專業上的想法與建議，因為有您們的意見讓本來不完整與鬆散的架構，在重整之後，讓我的論文能更加完整及嚴謹。

還要感謝實驗室的同學與學長們，謝謝人豪、培倫、暐勳，在我遇到問題時，給予我最大的幫助、想法、與鼓勵。

最後要感謝我的家人，謝謝你們這些年來的支持，讓我能夠無憂無慮的讀書，因為你們對我的關心，讓我堅強的走到今天，度過這兩年的磨練。

東海大學資訊工程學系 高效能實驗室 萬宗岳 105 年 07 月

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cloud computing is an operation base on Internet; it brings a huge change for industry evolved with the use of the Internet. Not only the IT industry which provides cloud computing technology, but also the general usage of it in the government, enterprise and individuals are changed with the born of cloud computing, all the places which need a lot of computing have changed because the birth of cloud computing. In IT industry, cloud computing caused the deepest impact undoubtedly. From the most basic computer components - processors, servers, storage devices, network equipment, information security equipment, software, data centers, information services, even the smartphones, tab and other mobile devices are having close relationship with cloud computing.

Cloud computing mainly combines virtualization, service management automation and standardized technology to provide flexible computing ability and data analysis method with high performance. Companies can run many kinds of service on the cloud platform without the need to construct data centers. This innovative computing and business model has attracted widespread attention in industry and academia.

Cloud computing is a major trend in recent years, especially in Big Data, when the data collect and analysis have to use cloud computing service. Although there are many cloud computing platform on the market, but the special needs of the enterprise and cannot choose the most suitable solution to use, OpenStack which the open-source service that can solve this problem, with each company to design its own environment.

1.1 Motivation

Cloud computing is the trend in IT industry today, companies, organizations and schools are following the trend of the cloud computing, establish a large-scale cloud computing cluster replace of one person with one computer. Although virtualization can reduce the cost of the hardware equipment, but also spawned a problem –energy consumption. Many places which need to deploy lots of virtual machines are using OpenStack, because of the characteristics of OpenStack that easy management of the virtual machine, OpenStack is widely used. But there does not show the resource utilization of physical machines and power consumption on the interface they provide, so we want to give the completely message of physical machines and virtual machines to users, make it more convenient to manage machines, then through live migration and algorithm to achieve to balance power consumption and energy saving.

1.2 Thesis Goal and Contributions

The objective of this paper is implement an energy saving cloud infrastructure with virtual machine power usage monitoring and live migration on OpenStack. It's mainly to monitor the real-time status and power consumption of every compute nodes and virtual machines on OpenStack, and achieve power saving through live migration. To monitor the utilization of CPU, load of memory, and power consumption of every compute nodes and virtual machines on OpenStack. These data show in real-time, completely monitor the real-time status of physical machines and virtual machines. It also record the utilization and power consumption of physical machines then show on this cloud infrastructure, to provide experimental evidence for the user as a reference. Base on the power consumption we monitoring, we can automatically allocate virtual machines on every physical machines by live migration, to balance the power consumption of every physical machines. It' s not only can avoid idle and waste of resources but also can avoid reducing machine life because of the physical machines always keep in high usage, and achieve to power saving.

1.3 Thesis Organization

In Chapter 2, we will describe some background information, including Cloud Computing, Virtualization, Hypervisor, OpenStack, Live Migration, PDU and related work. In Chapter 3, we will introduce our experimental environment and the overall architecture of our design. Chapter 4 shows the experimental results and analysis. Chapter 5 provides conclusions and future work of this work.

Chapter 2

Background Review and Related Work

2.1 Cloud Computing

The US National Institute of Standards and Technology (NIST) definition of cloud is: ”Cloud computing [1–4] is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” This cloud model is composed of five essential characteristics, three service models, and four deployment models, as shown in Figure 2.1.

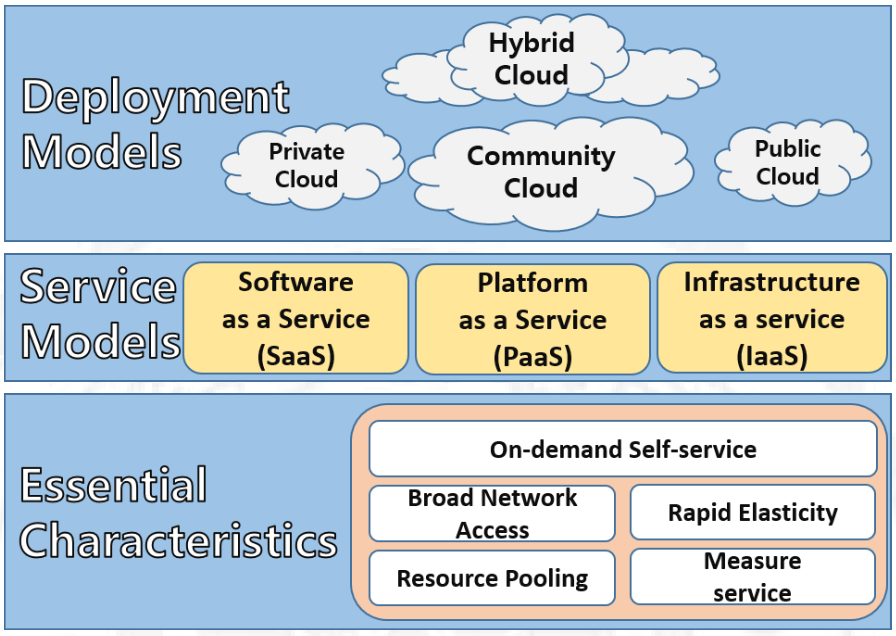


FIGURE 2.1: The overall cloud model

2.1.1 Essential Characteristics

According to the National Institute of Standards and Technology (NIST) definition of cloud computing identifies "five essential characteristics":

- On-demand Self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- Broad Network Access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- Resource Pooling: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided

resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

- Rapid Elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
- Measured Service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

2.1.2 Service Models

According to the National Institute of Standards and Technology (NIST) definition , cloud service architecture that following the service type divided into three layers, namely, infrastructure as a service (IaaS), Platform as a Service (PaaS) and software as a service (SaaS). The so-called cloud computing service type is able to provide the service to users, and through such a service which allows users to obtain resources, and how users use such services. They were introduced as follows:

- Infrastructure as a Service (IaaS): The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

The Benefits of IaaS:

  - No need to invest in your own hardware.
  - Infrastructure scales on demand to support dynamic workloads.
  - Flexible, innovative services available on demand.

- Platform as a Service (PaaS): The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

  The Benefits of PaaS:

  - Develop applications and get to market faster.
  - Deploy new web applications to the cloud in minutes.
  - Reduce complexity with middleware as a service.

- Software as a Service (SaaS): The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

  The Benefits of SaaS:

  - You can sign up and rapidly start using innovative business apps.
  - Apps and data are accessible from any connected computer.

  - No data is lost if your computer breaks, as data is in the cloud.
  - The service is able to dynamically scale to usage needs.

2.1.3 Deployment Models

Cloud computing providers and users according to their ownership can be divided into four categories, namely public cloud, private cloud, community and hybrid clouds.

- Public Cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

  Key aspects of public cloud:

  - Innovative SaaS business apps for applications ranging from customer resource management (CRM) to transaction management and data analytics.
  - Flexible, scalable IaaS for storage and compute services on a moment's notice.
  - Powerful PaaS for cloud-based application development and deployment environments.

- Private Cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

  Key aspects of private cloud:

  - A self-service interface controls services, allowing IT staff to quickly provision, allocate and deliver on-demand IT resources.

  - Highly automated management of resource pools for everything from compute capability to storage, analytics and middleware.
  - Sophisticated security and governance designed for a company's specific requirements.

- Hybrid Cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

  Key aspects of hybrid cloud:

  - Allows companies to keep the critical applications and sensitive data in a traditional data center environment or private cloud.
  - Enables taking advantage of public cloud resources like SaaS, for the latest applications, and IaaS, for elastic virtual resources.
  - Facilitates portability of data, apps and services and more choices for deployment models.

- Community Cloud: The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

2.2 Virtualization

In computing, virtualization [5–8] refers to the act of creating a virtual (rather than actual) version of something, including (but not limited to) a virtual computer hardware platform, operating system (OS), storage device, or computer network resources. With virtualization, the computer's physical resources, such as servers, network, memory, and storage, are abstractly presented after conversion, so that users can apply those resources in a better way than the original configuration.

Simply put, Virtualization is a technology that allows you to transform hardware into software, and it allows you to run multiple operating systems simultaneously on a single computer.

Virtual architecture is different from traditional architecture, as shown in Figure 2.2 Traditional architecture can run single operating system on a single computer, but the virtual architecture can run multiple operating system on a single computer.

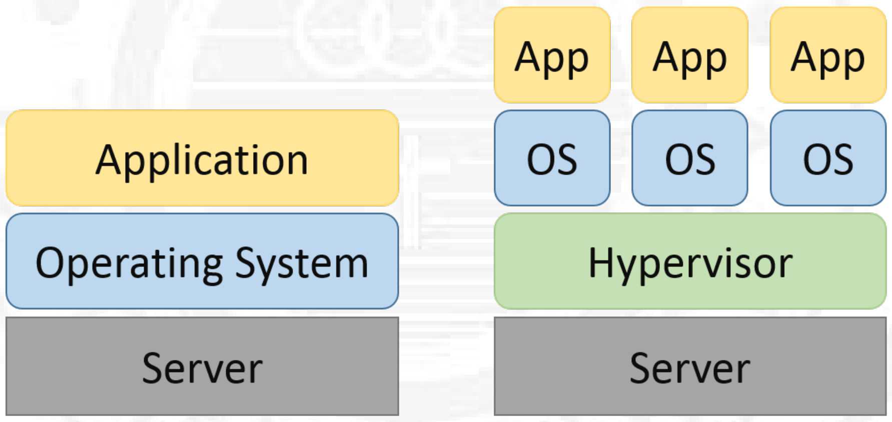


FIGURE 2.2: The different between traditional architecture and virtual architecture

There are many benefits of virtualization, such as:

- Encapsulation - VMs can be described in a file
  - Possible to ”snapshot”
  - Easy to move
- Enables running multiple operating systems
- Consolidation and use of unused computation power
- Resource management
- High availability and disaster recovery

- Create ”Base Environment”
- Safe testing of new software
- Easy Management

2.3 Hypervisor

A hypervisor [9, 10] or virtual machine monitor (VMM) is a piece of computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor is running one or more virtual machines is defined as a host machine. Each virtual machine is called a guest machine. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources.

Simply stated, a hypervisor creates a layer of abstraction that isolates an OS and its associated applications from the underlying computing hardware. The isolation effectively mitigates software from its traditional reliance on hardware devices and their drivers. The implications of this behavior are profound. A hypervisor allows OSes and their application workloads to run on a broader array of hardware. Similarly, multiple OSes and workloads, each a unique virtual machine (VM) or VM instance, can reside on the same system to simultaneously share computing resources. You can migrate each VM between computing platforms on demand with little (if any) processing disruption. The result is better use of computing platforms with seamless workload migration and backup capabilities. Hypervisors generally fall into two categories: hosted and bare-metal. Both offer distinct benefits and drawbacks.

2.3.1 Hosted Hypervisor

A hosted hypervisor, as shown in Figure 2.3, runs within the OS and allows additional OS and application instances to run on top of it. VMware Server and Microsoft Virtual Server, as well as numerous endpoint-based virtualization platforms like VMware Workstation, Microsoft Virtual PC and Parallels Workstation are hosted hypervisors.

There are advantages of Hosted Hypervisor:

- Virtualization installs like application rather than like OS.
- Can run alongside conventional applications.
- Avoid code duplication –OS already has process scheduler, memory management, device support etc.
- More suitable for personal users.

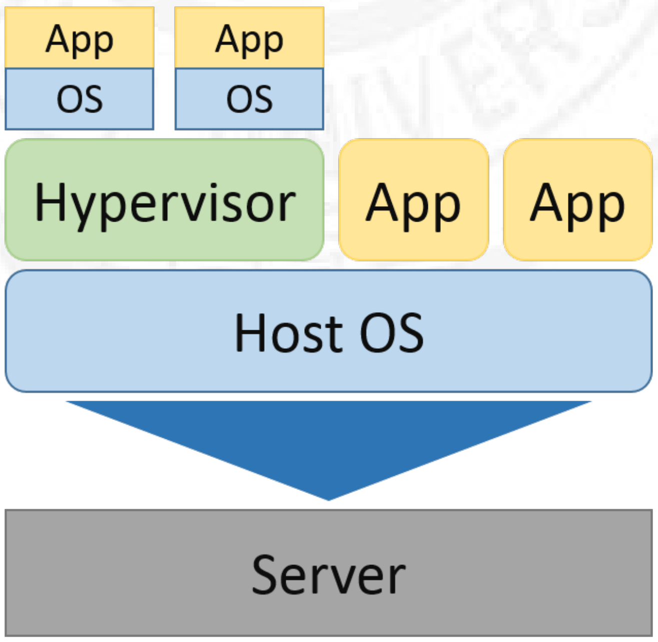


FIGURE 2.3: The hosted hypervisor architecture

2.3.2 Bare-Metal Hypervisor

Bare-metal hypervisors, as shown in Figure 2.4, is the most commonly deployed type, can install directly onto the computing hardware. The OS installs and runs above the hypervisor. Major virtualization products can be termed as bare-metal hypervisors, including Oracle VM, VMware ESX Server, Microsoft Hyper-V and Citrix XenServer.

There are advantages of Bare-Metal Hypervisor:

- Better performance with lower overhead.
- Highly efficient direct I/O pass-through architecture for network and disk.
- Complete control over hardware.
- Advanced features like live migration available.
- Suitable for production environments.

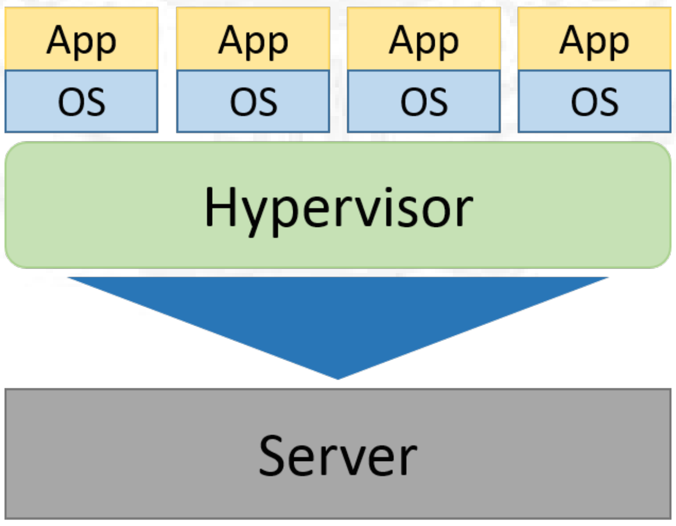


FIGURE 2.4: The bare-metal hypervisor architecture

2.4 OpenStack

OpenStack [11–14] is a free and open-source cloud computing software platform. It began in 2010 as a joint project of Rackspace Hosting and NASA. Currently, it is managed by the OpenStack Foundation, a non-profit which oversees both development and community-building around the project. And OpenStack.org released it under the terms of the Apache License. Users primarily deploy it as an infrastructure as a service (IaaS) solution. The technology consists of a series of interrelated projects that control pools of processing, storage, and networking resources throughout a data center which users manage through a web-based dashboard, command-line tools, or a RESTful API.

2.4.1 OpenStack Component

OpenStack has a modular architecture with various code names for its components.

- Compute (Nova)

  OpenStack Compute (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system. It is designed to manage and automate pools of computer resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC) configurations. KVM [15], VMware, and Xen are available choices for hypervisor technology, together with Hyper-V and Linux container technology such as LXC.

  It is written in Python and uses many external libraries such as Eventlet (for concurrent programming), Kombu (for AMQP communication), and SQLAlchemy (for database access). Compute's architecture is designed to scale horizontally on standard hardware with no proprietary hardware or software requirements and provide the ability to integrate with legacy systems and third-party technologies.

- Object Storage (Swift)

OpenStack Object Storage (Swift) is a scalable redundant storage system. Objects and files are written to multiple disk drives spread throughout servers in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally simply by adding new servers. Should a server or hard drive fail, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used. The Total Cost of Ownership (TCO) can be higher than using enterprise-class storage due to many copies required to get high availability.

In August 2009, Rackspace started the development of the precursor to OpenStack Object Storage, as a complete replacement for the Cloud Files product. The initial development team consisted of nine developers. SwiftStack, an object storage software company, is currently the leading developer for Swift.

- Block Storage (Cinder)

  Cinder is a Block Storage service for OpenStack. It's designed to allow the use of either a reference implementation (LVM) to present storage resources to end users that can be consumed by the OpenStack Compute Project (Nova). The short description of Cinder is that it virtualizes pools of block storage devices and provides end users with a self service API to request and consume those resources without requiring any knowledge of where their storage is actually deployed or on what type of device.

- Networking (Neutron)

  OpenStack Networking (Neutron, formerly Quantum) is a system for managing networks and IP addresses. OpenStack Networking ensures the network is not a bottleneck or limiting factor in a cloud deployment, and gives users self-service ability, even over network configurations.

OpenStack Networking provides networking models for different applications or user groups. Standard models include flat networks or VLANs that separate servers and traffic. OpenStack Networking manages IP addresses, allowing for dedicated static IP addresses or DHCP. Floating IP addresses let traffic be dynamically rerouted to any resources in the IT infrastructure, so users can redirect traffic during maintenance or in case of a failure.

Users can create their own networks, control traffic, and connect servers and devices to one or more networks. OpenStack Networking provides an extension framework that can deploy and manage additional network services —such as intrusion detection systems (IDS), load balancing, firewalls, and virtual private networks (VPN).

- Dashboard (Horizon)

  OpenStack Dashboard (Horizon) provides administrators and users a graphical interface to access, provision, and automate cloud-based resources. The design accommodates third party products and services, such as billing, monitoring, and additional management tools. The dashboard is also brandable for service providers and other commercial vendors who want to make use of it. The dashboard is one of several ways users can interact with OpenStack resources. Developers can automate access or build tools to manage resources using the native OpenStack API or the EC2 compatibility API.

- Identity Service (Keystone)

  OpenStack Identity (Keystone) provides a central directory of users mapped to the OpenStack services they can access. It acts as a common authentication system across the cloud operating system and can integrate with existing backend directory services like LDAP. It supports multiple forms of authentication including standard username and password credentials, token-based systems and AWS-style (i.e. Amazon Web Services) logins. Additionally, the catalog provides a queryable list of all of the services deployed in an OpenStack cloud in a single registry. Users and third-party tools can programmatically determine which resources they can access.

- Image Service (Glance)

  OpenStack Image Service (Glance) provides discovery, registration, and delivery services for disk and server images. Stored images can be used as a template. It can also be used to store and catalog an unlimited number of backups. The Image Service can store disk and server images in a variety of back-ends, including OpenStack Object Storage. The Image Service API provides a standard REST interface for querying information about disk images and lets clients stream the images to new servers.

  Glance—OpenStack's image service module—is a compute module, as it does not store images, variations, or instances—but rather catalogs them and holds their metadata from Swift or a storage backend datastore. Other modules must communicate with the images metadata through Glance—for example, Heat. Also, Nova can present information about the images, and configure a variation on an image to produce an instance. However, Glance is the only module that can add, delete, share, or duplicate images.

- Telemetry (Ceilometer)

  OpenStack Telemetry Service (Ceilometer) provides a Single Point Of Contact for billing systems, providing all the counters they need to establish customer billing, across all current and future OpenStack components. The delivery of counters is traceable and auditable, the counters must be easily extensible to support new projects, and agents doing data collections should be independent of the overall system.

- Orchestration (Heat)

  Heat is the main project in the OpenStack Orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files that can be treated like code. A native Heat template format is evolving, but Heat also endeavours to provide compatibility with the AWS CloudFormation template format, so that many existing CloudFormation templates can be launched on OpenStack. Heat provides both an OpenStack-native ReST API and a CloudFormation-compatible Query API.

- Database (Trove)

  Trove is Database as a Service for OpenStack. It's designed to run entirely on OpenStack, with the goal of allowing users to quickly and easily utilize the features of a relational or non-relational database without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed. Initially, the service will focus on providing resource isolation at high performance while automating complex administrative tasks including deployment, configuration, patching, backups, restores, and monitoring.

2.4.2 OpenStack Conceptual Architecture

Launching a virtual machine or instance involves many interactions among several services. The Figure 2.5 provides the conceptual architecture of a typical OpenStack environment.

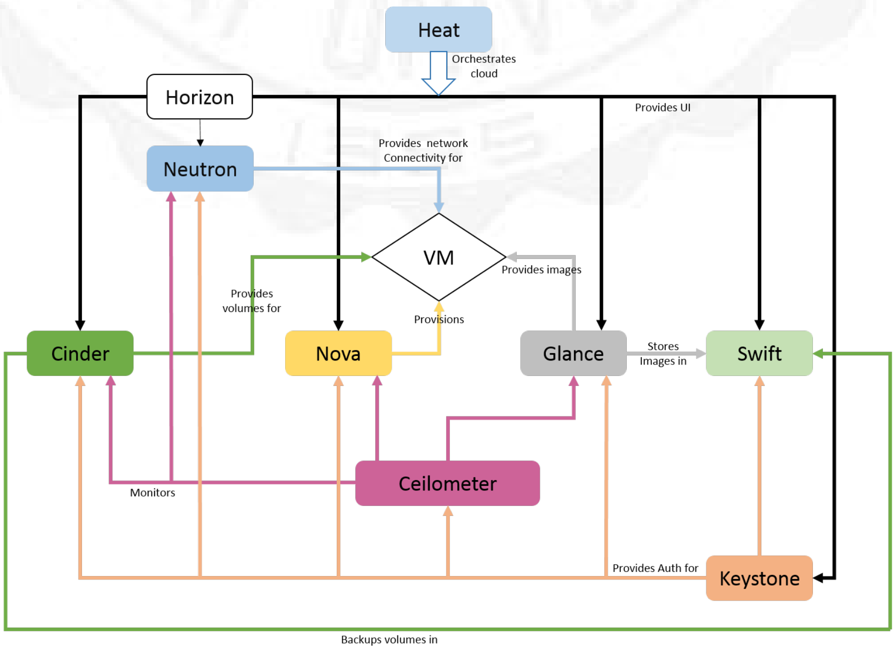


Figure 2.5: The conceptual architecture of OpenStack

In this work, we use version Kilo. We just use Nova, Glance, Keystone, Horizon in our model.

2.5 Live Migration

Live migration [16–20],as shown in Figure 2.6, refers to the process of moving a running VM or application between different physical machines without disconnecting the client or application. Memory, storage, and network connectivity of the VM are transferred from the original guest machine to the destination.

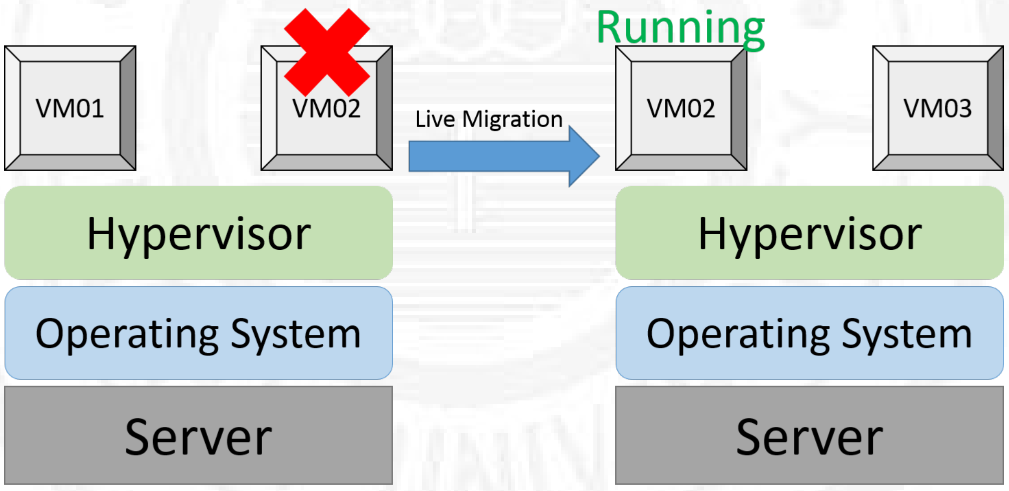


FIGURE 2.6: The concept of Live Migration

Two techniques for moving the VM's memory state from the source to the destination are pre-copy memory migration and post-copy memory migration.

- Pre-copy memory migration, as shown in Figure 2.7.
  - Warm-up phase

    In pre-copy memory migration, the hypervisor typically copies all the memory pages from source to destination while the VM is still running on the source. If some memory pages change (become 'dirty') during this process, they will be re-copied until the rate of re-copied pages is not less than the page dirty rate.

– Stop-and-copy phase

After the warm-up phase, the VM will be stopped on the original host, the remaining dirty pages will be copied to the destination, and the VM will be resumed on the destination host. The time between stopping the VM on the original host and resuming it on destination is called "down-time", and it ranges from a few milliseconds to seconds according to the size of memory and applications running on the VM. There are some techniques to reduce live migration down-time, such as using probability density function of memory change.

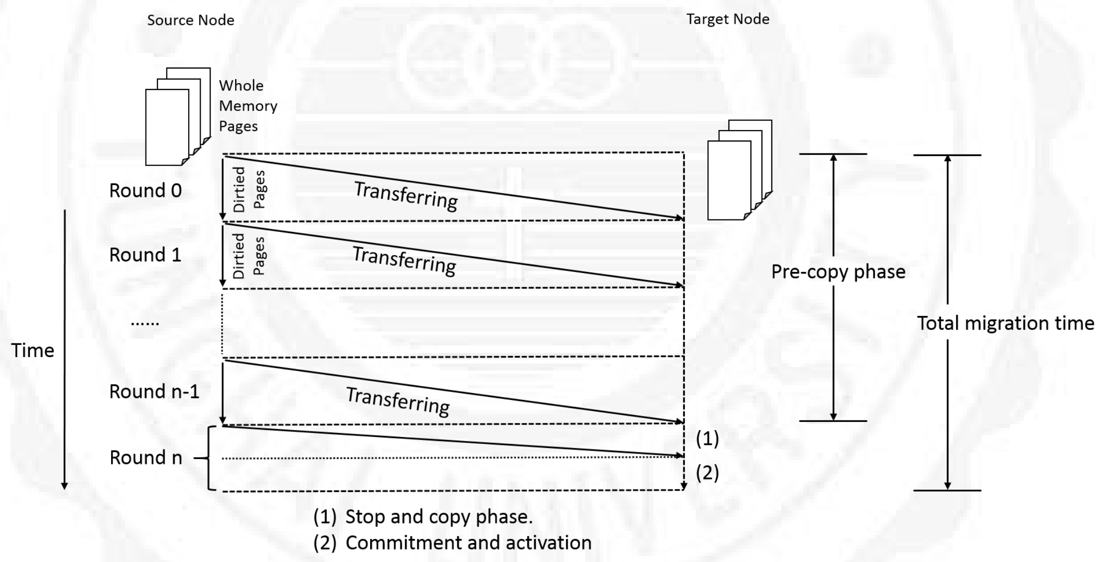


FIGURE 2.7: The phase of Pre-copy memory migration

- Post-copy memory migration

Post-copy VM migration is initiated by suspending the VM at the source. With the VM suspended, a minimal subset of the execution state of the VM (CPU state, registers and, optionally non-pageable memory) is transferred to the target. The VM is then resumed at the target. Concurrently, the source actively pushes the remaining memory pages of the VM to the target - an activity known as pre-paging. At the target, if the VM tries to access a page that has not yet been transferred, it generates a page-fault. These faults, known as network faults, are trapped at the target and redirected to the source, which responds with the faulted page. Too many network faults can degrade performance of applications running inside the VM. Hence pre-paging can dynamically adapt the page transmission order to network faults by actively pushing pages in the vicinity of the last fault. An ideal pre-paging scheme would mask large majority of network faults, although its performance depends upon the memory access pattern of the VM's workload. Post-copy sends each page exactly once over the network. In contrast, pre-copy can transfer the same page multiple times if the page is dirtied repeatedly at the source during migration. On the other hand, pre-copy retains an up-to-date state of the VM at the source during migration, whereas with post-copy, the VM's state is distributed over both source and destination. If the destination fails during migration, pre-copy can recover the VM, whereas post-copy cannot.

2.6 NFS (Network File System)

Network File System (NFS) [21–23], as shown in Figure 2.8, is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network much like the local storage. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. The Network File System is an open standard defined in Request for Comments (RFCs), allowing anyone to implement the protocol. Even though different universities and laboratories have developed a variety of distributed file systems, NFS is the first product is applicable for both academic and commercial use.

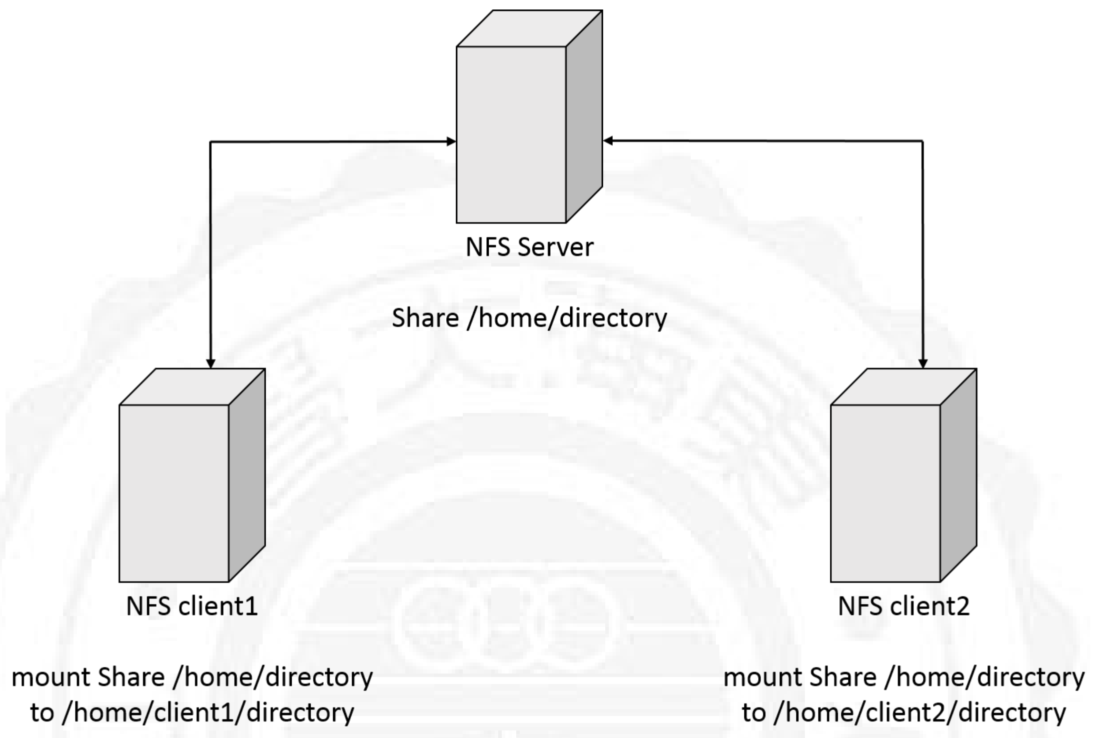

NFS Server
Share /home/directory
NFS client1
mount Share /home/directory
to /home/client1/directory
NFS client2
mount Share /home/directory
to /home/client2/directory

Figure 2.8: The Network File System (NFS)

NFS's basic principle is "to allow different clients and server nodes share the same file system through a set of RPC", thus, independent of the operating system, NFS allows different hardware and operating systems to share a common file system. NFS provides the following services:

- Search file in the directory.
- List the files in the directory.
- Manage Directory.
- Obtain attribute of all files.
- The file read / write

NFS is often used with Unix operating systems (such as Solaris, AIX and HP-UX) and Unix-like operating systems (such as Linux and FreeBSD). It is also available to operating systems such as the classic Mac OS, OpenVMS, IBM i, certain editions of Microsoft Windows, and Novell NetWare, and alternative remote file access protocols including the Server Message Block (SMB, also known as CIFS), Apple Filing Protocol (AFP), NetWare Core Protocol (NCP), and OS/400 File Server file system (QFileSvr.400).

2.7 PDU (Power Distribution Units)

A Power Distribution Unit (PDU) [24, 25],as shown in Figure 2.9, is a device used in datacenters to distribute AC power to multiple servers and other equipment. Power distribution units (PDUs) range from simple 120v power strips to units that break out 120 volts from 240v and three-phase power. Advanced units are managed remotely via the SNMP management protocol or from a Web browser or other management console, causing outlets to be turned on and off at prescribed times and in a proper sequence for shutting down and powering up equipment.

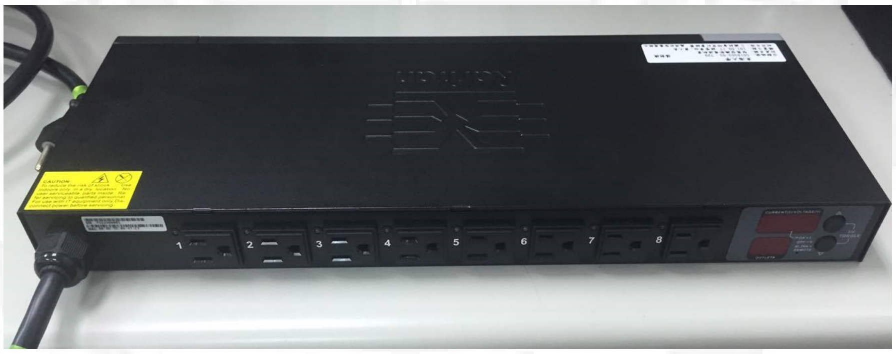


FIGURE 2.9: Raritan's PDU

The growing complexity of IT environments, from wiring closets and server rooms to data centers of all sizes, has increased the need for reliable power distribution to the rack level. Eliminating power management issues is essential for IT and Facilities managers to maintain system availability of increasing higher density equipment. Power Distribution Units are an essential element in managing power capacity and functionality for critical network, server and data center equipment.

- Basic PDU

The most basic PDU is a large power strip without surge protection. It is designed to provide standard electrical outlets for data center equipment and has no monitoring or remote access capabilities. The floor-mounted and rack-mounted PDUs can be more sophisticated, providing data that can be used for power usage effectiveness (PUE) calculations.

- Floor-mounted PDU

  A floor-mounted PDU, sometimes called a main distribution unit (MDU), provides an important management bridge between a building's primary power and various equipment racks within a data center or network operations center (NOC). Each PDU can handle larger amounts of energy than an ordinary power strip (300 kilovolt-amps and higher depending on the manufacturer and model) and typically provides power to multiple equipment racks.

- Rack-mountable PDU

  A rack-mountable PDU mounts directly to an equipment rack so it can control and monitor power to specific servers, switches and other data center devices and assist in balancing power loads. Rack-mountable PDAs are known by several different names, including smart PDUs and intelligent PDUs. Such PDUs include three-phase displays for devices sharing power well as remote management tools that use the Simple Network Management Protocol (SNMP) to provide administrators with the ability to adjust and monitor power demands from offsite locations.

2.8 Related Work

In the recent years, there are many power saving method research. We choose some research about power saving method to discussion.

In first paper [26], it is about power saving method for virtual machine management platform in cloud. They use the open source codes and PHP web programs to implement a virtualization resource management system for power-saving. In this thesis, they propose to adopt system integrated open source software like KVM and libvirt to construct a virtual cloud management platform, which detects the status of resources via SNMP, calculates the operation efficiency of the overall system, allocates virtual machines through the live migration technology and turns off extra machines on the cloud to save energy.

Their objective is to provide enterprises or end users with power-saving private cloud solutions. In this work they also have built a webpage to allow users to easily access the cloud virtualization resources, i.e., users can manage virtual machines and monitor the status of resources via the web interface. From analysis of the experimental results of live migration of virtual machines, this work demonstrates that efficient use of hardware resources is realized by the power-saving method, and the aim of power-saving is achieved.

It is about a cloud infrastructure monitor platform with power saving method in second paper [27]. The aim of this thesis is based on the Cloud service Infrastructure as a Service, use KVM and libvirt to build cloud environments; in addition, they calculate effective power consumption by writing a PHP program to collect host and VM information to calculate the efficiency of the entire system resource. By giving every host and VM a power cap, they perform live-migration on VMs and shutdown idle hosts to save power. They also build an interface that enables users to monitor virtual machines and physical machines. In the experiment, they use PDU to record the power consumption information, and from analysis of these data they prove that they can more effectively utilize hardware resources and save power.

In third paper [28], it is about a cloud energy saving system with virtual machine dynamic resource allocation method base on OpenStack. They mention that many companies, organizations and academic institutions are following the cloud computing trend; the establishment of large-scale cloud computing clusters avoids the need to provide one person with one computer. Even though virtualization can reduce the cost of hardware equipment, but it still faces with two problems: energy consumption and the waste of the idle resources.

To solve these two problems, they propose two algorithms, i.e., dynamic resource allocation and energy saving. In order to implement these two algorithms with live migration of virtual machines, they first build an infrastructure platform based on cloud software – OpenStack. Next, dynamic resource allocation and energy saving algorithms are designed and implemented. Finally, they use the Power Distribution Unit (PDU) to monitor system status and record power consumption; the real time status monitoring data verify that the proposed algorithms are efficient in energy saving and idle resource planning.

In A method for managing green power of a virtual machine cluster in cloud [29], they mention about the gross occupied resource weight ratio is defined as the ratio of the sum of resource weights of all virtual machines over the sum of available resource weights of all running physical machines. When the gross occupied resource weight ratio is greater than the maximum tolerant occupied resource weight ratio, preset to ensure quality of service, a standby physical machine in the non-running physical machines is selected and wakened up to join as one of the running physical machines. On the other hand, when the gross occupied resource weight ratio is less than the minimum critical occupied resource weight ratio, preset to trigger energy saving algorithms, one of the running physical machines, selected as a migration physical machine with the virtual machines therein removed after live migration, is moved from other running physical machines, and then turned off. As a result, a resource allocation process is realized to distribute loads of the running physical machines such that the total number of the running physical machines can be flexibly dispatched to achieve the objective of green power management.

Chapter 3

System Design and Implementation

With the universal of virtualization and OpenStack, how to reduce the power consumption is an issue that we face, and the user interface that OpenStack provided has some insignificance. In this section, we will introduce our system architecture and the flow of our algorithm. Finally, we will show our user interface.

3.1 System Architecture

This section introduces the architecture of the proposed cloud platform based on the infrastructure software OpenStack. The architecture consists of a controller node and two computing nodes. Several major OpenStack services are running on the Controller node, such as Identity service, Image service, Networking service, Nova service, and Dashboard. To perform live migration on the VMs between the two computing nodes, we use Network File System (NFS) as shared storage and install the NFS server on the controller node. These three computing nodes are only utilized to run the Nova service and NFS client and connected to the PDU for monitoring and recording their energy consumption. The overall system architecture is shown in the Figure 3.1.

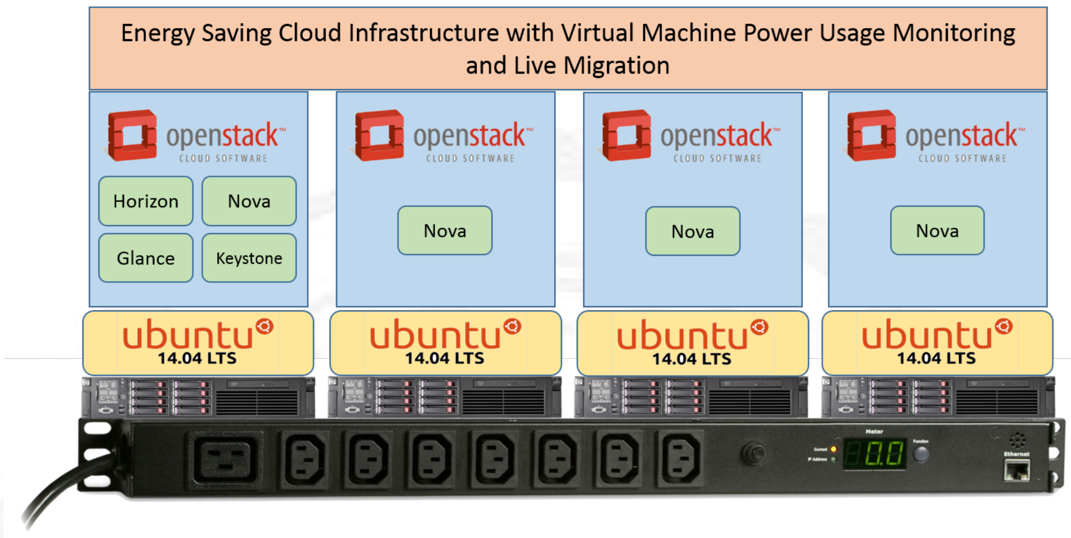


FIGURE 3.1: The overall system architecture

3.2 Design Flow and Algorithm

To achieve efficient distributed load balance and energy saving, two algorithms are designed based on OpenStack. The algorithmic design flow for DLB and the algorithmic design flow for energy saving are introduced in subsection 3.2.1 and subsection 3.2.2, respectively.

3.2.1 Design Flow and Algorithm of Distributed Load Balancing Method

To achieve efficient distributed load balance and energy saving, we consider the performance index in 3.5, where P represent the power of every compute nodes after distributing. In other words, we need to minimize the summation of differences between every two nodes as follow.

$$minimize \mid P_1 - P_2 \mid^2 + \mid P_2 - P_3 \mid^2 + \mid P_1 - P_3 \mid^2 \tag{3.1}$$

To achieve above minimum value, we propose the function,

P_m is defined the power of all VMs on m PM

$p_(m,n)$ is defined the power of n VM on m PM

$$P_m \;=\; (\textstyle\sum_{i=1}^{n} p_{m,i}) \tag{3.2}$$

and our goal is

$$P_\mu \;=\; \frac{(\sum_{i=1}^{m} P_i)}{m} \tag{3.3}$$

So we have to find the following:

$$min\ \textstyle\sum_{i=1}^{m}(P_i \;-\; P_\mu)^2 \tag{3.4}$$

The proposed method for finding the minimum value is introduced as follows. First, we calculate power of VMs on all compute nodes and arrangement in descending, and calculate the value P, P represent power of all VMs / number of compute node, it means that how much power on each compute node is the most balanced. Then it start from first VM, it judge that whether put this VM in first node will over P, it will check on other node if it over P, or it will put this VM in first node. If the VM has been check on all nodes, but all nodes will over P, it will put in the node which is most close to P. Above steps will continue until all VMs have been distributed on compute nodes, then start live migration.

A_{tatal} is defined the total power of all VMs / number of PM

3.2.2 Design Flow and Algorithm of Power Saving Method

First, we can choose which node we want to shutdown on web site, then we will check the other compute nodes wherther have enough CPU usage. Then we calculate power of VMs on all compute nodes and arrangement in descending, and calculate the value P, P represent power of all VMs / number of compute node, it

Algorithm 1 DLB method Algorithm

```
1: if (power of VM + total power of VM on the compute node > $A_{total}$) then
2:     if (All nodes have been check) then
3:         Put this VM in the node which is the most close to $A_{total}$
4:     else
5:         Check other node
6:     end if
7: else
8:     Put this VM in the node
9: end if
10: if (All VMs have been put in nodes)  then
11:     Start live migration
12: else
13:     Keep distributing VM
14: end if
```

means that how much power on each compute node is the most balanced. Then it start from first VM, it judge that whether put this VM in first node will over P, it will check on other node if it over P, or it will put this VM in first node. If the VM has been check on all nodes except the node that we choose to shutdown, but all nodes will over P, it will put in the node which is most close to P. Above steps will continue until all VMs have been distributed on compute nodes, then start live migration, and it will shutdown the node we choose first after live migration.

We check the other compute nodes wherther have enough CPU usage by following:

$C_(p, m)$ = Number of CPU core of m host
$C_(v, n)$ = Number of CPU core of n VM

Check whether:

$$\sum_{i=1}^{m} C_{(p,i)} > \sum_{j=1}^{n} C_{(v,j)} \quad (3.5)$$

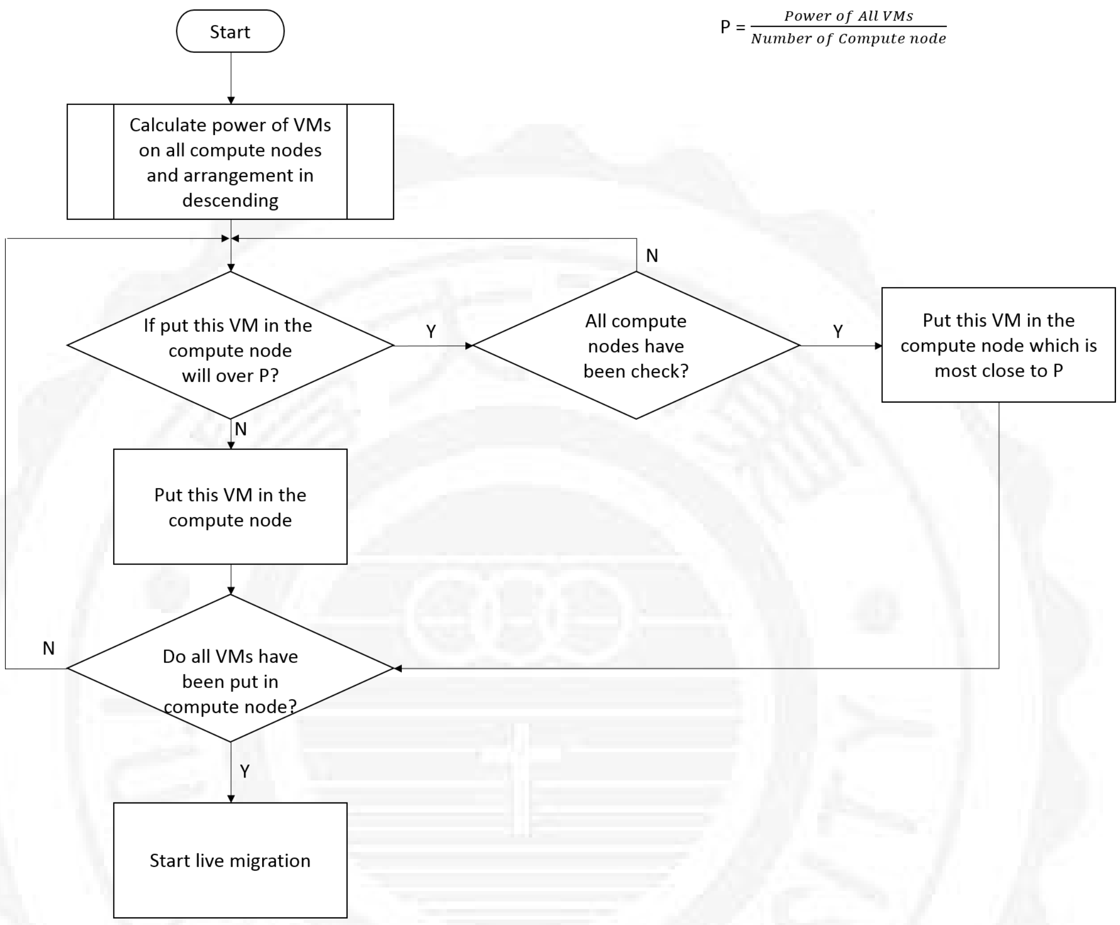

P = Power of All VMs / Number of Compute node
Start
Calculate power of VMs on all compute nodes and arrangement in descending
If put this VM in the compute node will over P?
Y
All compute nodes have been check?
Y
Put this VM in the compute node which is most close to P
N
N
Put this VM in the compute node
Do all VMs have been put in compute node?
N
Y
Start live migration

FIGURE 3.2: DLB method flow chart

We will choose which compute node will be shutdown on web site first
S is defined as the compute node which we want to shutdown
A_{tatal} is defined the total power of all VMs / number of PM

3.3 System Implementation

In this thesis, we will use the Python language and PHP to write some automatic program, including the state monitoring program which can monitor the virtual machines and the physical machines, energy consumption record program which can monitor the power consumption. The following is a detailed description of the programs.

Algorithm 2 Power Saving method Algorithm

```
1: if (Other two compute nodes have enough CPU usage) then
2:     if (power of VM + total power of VM on the compute node > $A_{total}$) then
3:         if (All nodes except $S$ have been check) then
4:             Put this VM in the node which is the most close to $A_{total}$
5:         else
6:             Check other node except $S$
7:         end if
8:     else
9:         Put this VM in the node
10:    end if
11:    if (All VMs have been put in nodes) then
12:        Start live migration then shutdown $S$
13:    else
14:        Keep distributing VM
15:    end if
16: else
17:    Reselect which compute node will be shutdown
18: end if
```

3.3.1 Status Monitoring

We monitored the states, i.e., CPU utilization and memory utilization, of physical machines and VMs via status monitoring program. We used python system and process utilities (psutil), which is a cross-platform library for retrieving information on running processes and system utilization of CPU, memory, disks, and networking in Python. It is useful mainly for system monitoring. The status monitoring function was developed with the Python programming language to capture status and post data to receiving program. We then used the receiving program which was developed with the PHP language to receive all monitoring data and insert these monitoring data into database, then show on the web site we design. The flow of status monitoring is shown in Figure 3.4.

3.3.2 Power Consumption Recording

In this work, we captured the power consumption of compute nodes via PDU. PDU's power consumption data is obtained by SNMP. The Energy Consumption Recording function, developed by PHP programing language, is an automatic recording program. Through Energy Consumption Recording program, we can automatically collect the power consumption data of compute nodes from PDU. Figure 3.6 and Figures 3.7 are the power consumption hitorical record on web site.

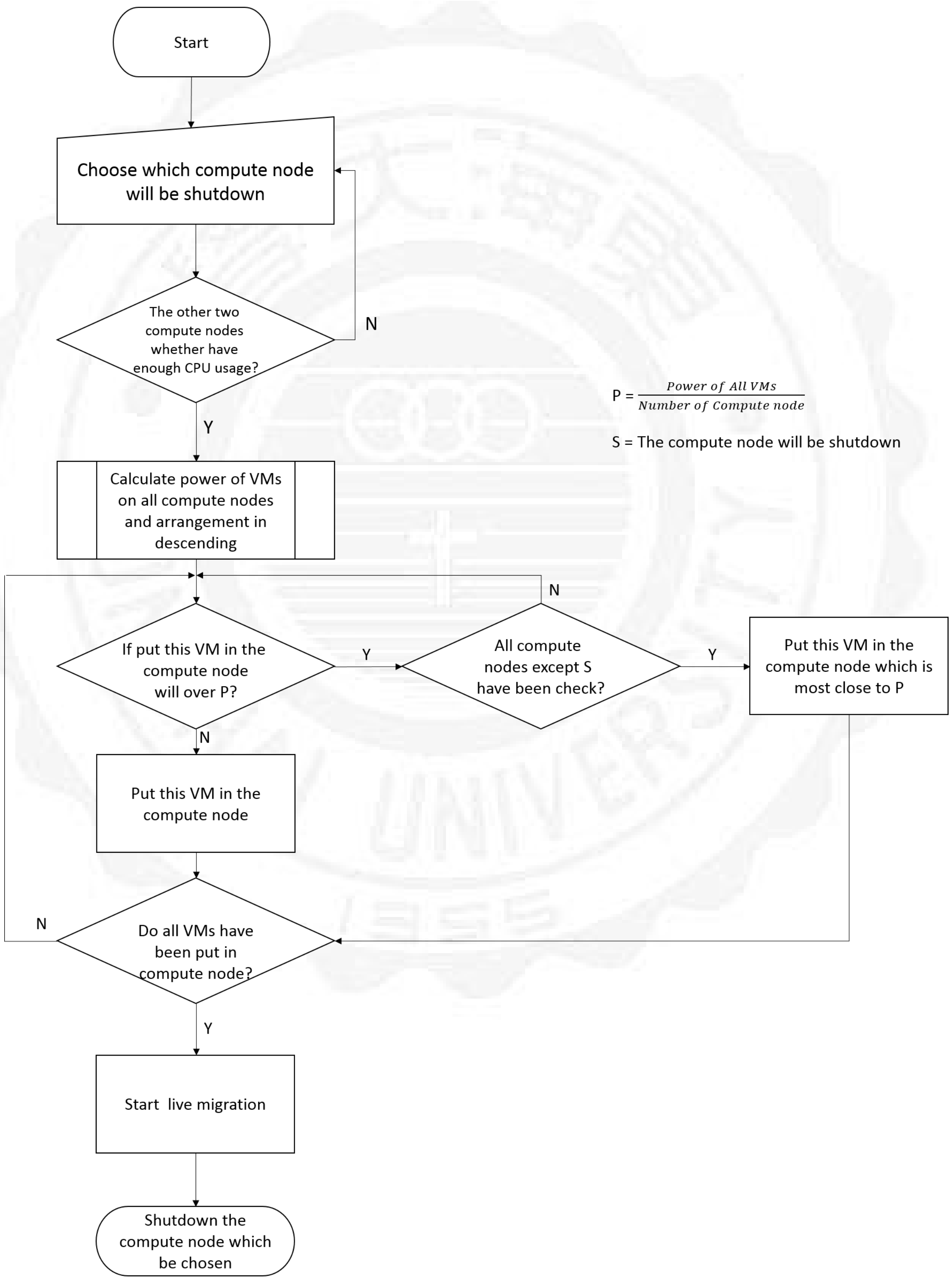

Start
Choose which compute node will be shutdown
The other two compute nodes whether have enough CPU usage?
N
Y
P = $\frac{Power\ of\ All\ VMs}{Number\ of\ Compute\ node}$
S = The compute node will be shutdown
Calculate power of VMs on all compute nodes and arrangement in descending
N
If put this VM in the compute node will over P?
Y
All compute nodes except S have been check?
Y
Put this VM in the compute node which is most close to P
N
Put this VM in the compute node
N
Do all VMs have been put in compute node?
Y
Start live migration
Shutdown the compute node which be chosen

Figure 3.3: Power Saving method flow chart

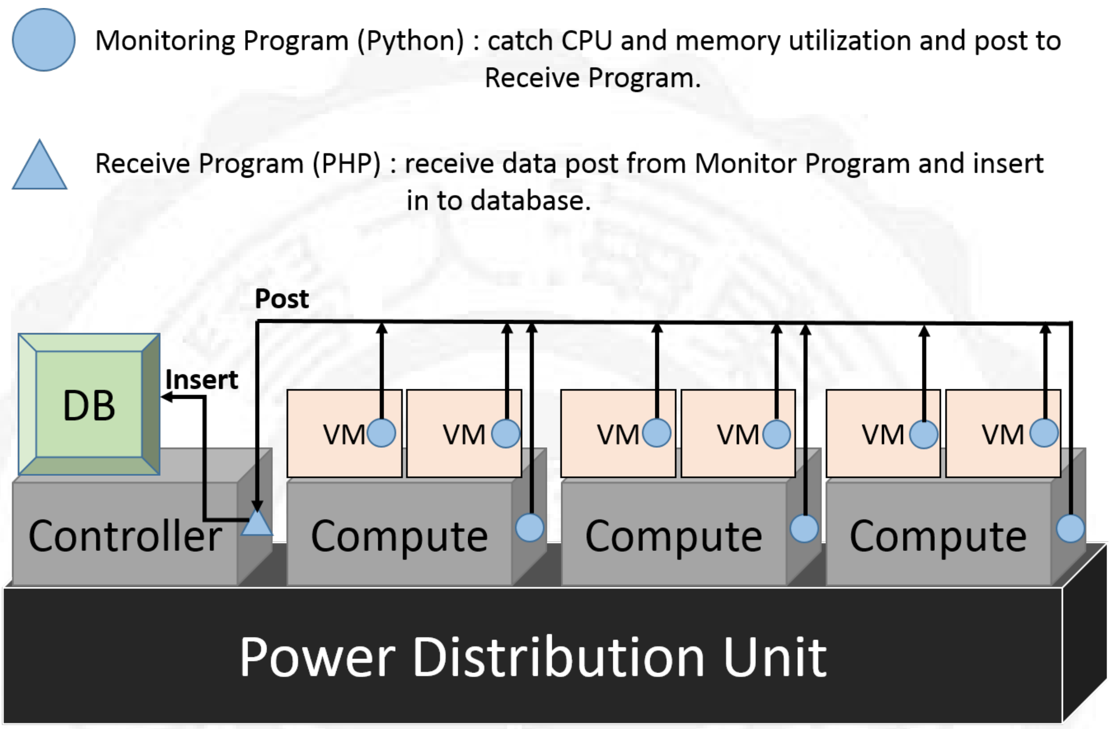
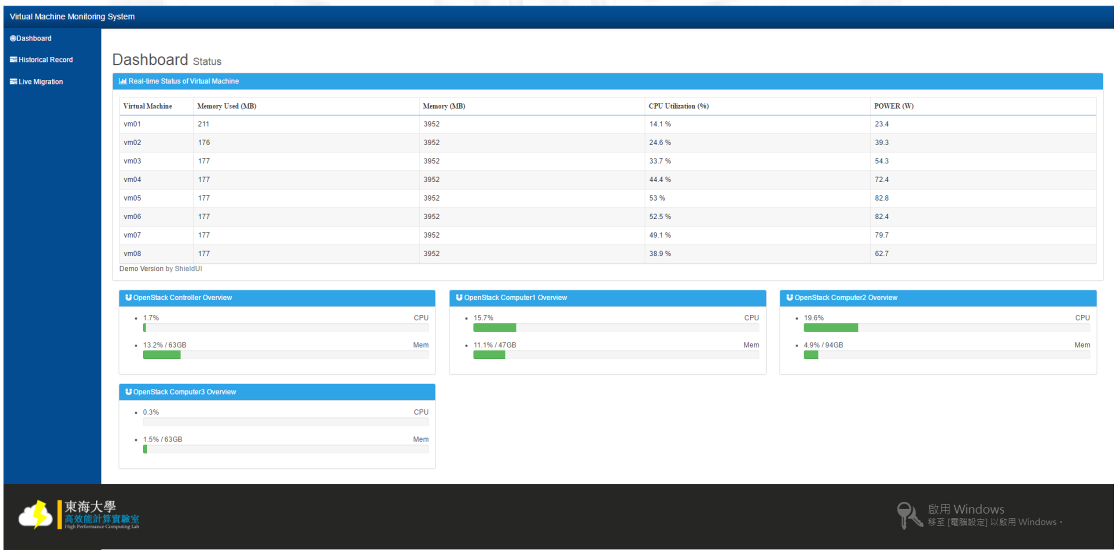

Monitoring Program (Python) : catch CPU and memory utilization and post to Receive Program.
Receive Program (PHP) : receive data post from Monitor Program and insert in to database.
Post
DB
Insert
VM
VM
VM
VM
VM
VM
Controller
Compute
Compute
Compute
Power Distribution Unit

FIGURE 3.4: The flow of status monitoring



Dashboard Status

Real-time Status of Virtual Machine

Virtual Machine	Memory Used (MB)	Memory (MB)	CPU Utilization (%)	POWER (W)
vm01	211	3952	14.1 %	23.4
vm02	176	3952	24.6 %	39.3
vm03	177	3952	33.7 %	54.3
vm04	177	3952	44.4 %	72.4
vm05	177	3952	53 %	82.8
vm06	177	3952	52.5 %	82.4
vm07	177	3952	49.1 %	79.7
vm08	177	3952	38.9 %	62.7

Demo Version by ShieldUI

OpenStack Controller Overview
- 1.7% CPU
- 13.2% / 63GB Mem

OpenStack Computer1 Overview
- 15.7% CPU
- 11.1% / 47GB Mem

OpenStack Computer2 Overview
- 19.6% CPU
- 4.9% / 94GB Mem

OpenStack Computer3 Overview
- 0.3% CPU
- 1.5% / 63GB Mem

東海大學
高效能計算實驗室
High Performance Computing Lab
啟用 Windows
移至 [電腦設定] 以啟用 Windows。

FIGURE 3.5: Status monitoring on web site

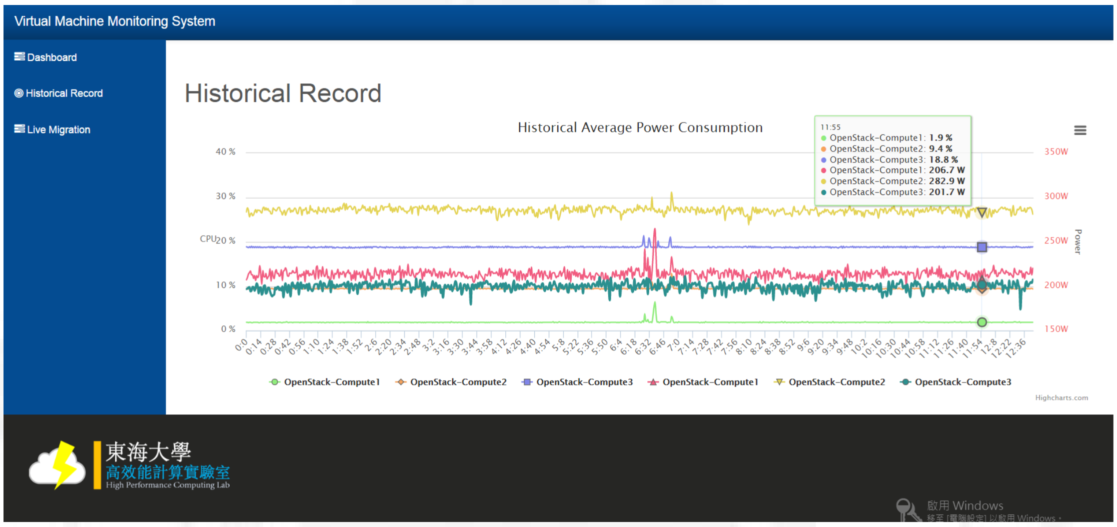

Virtual Machine Monitoring System
Dashboard
Historical Record
Live Migration
Historical Record
Historical Average Power Consumption
11:55
OpenStack-Compute1: 1.9 %
OpenStack-Compute2: 9.4 %
OpenStack-Compute3: 18.8 %
OpenStack-Compute1: 206.7 W
OpenStack-Compute2: 282.9 W
OpenStack-Compute3: 201.7 W
CPU
Power
OpenStack-Compute1
OpenStack-Compute2
OpenStack-Compute3
OpenStack-Compute1
OpenStack-Compute2
OpenStack-Compute3
Highcharts.com
東海大學
高效能計算實驗室
High Performance Computing Lab

FIGURE 3.6: Power consumption hitorical record

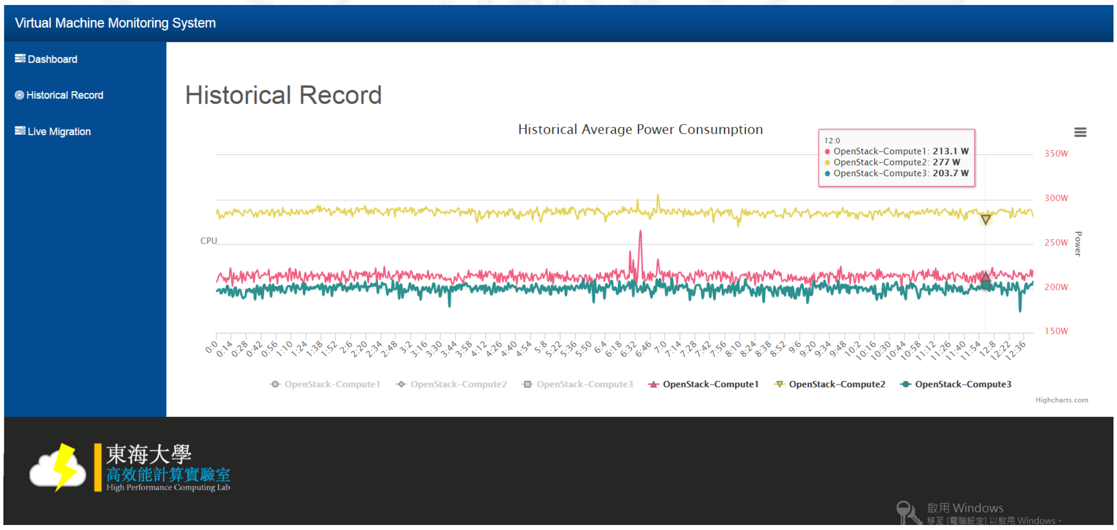


FIGURE 3.7: Power consumption hitorical record

Chapter 4

Experimental Results

In this chapter, we show the experimental environment and the experimental results. In section 4.1, we describe our experimental environment including hardware specification and software specification. The experimental results are shown in section 4.2. We test the two algorithms implemented in the same system and show the experimental results.

4.1 Experimental Environment

The experimental environment consists of four computers and their hardware specifications are listed in Table 4.1. Controller consist of 12-core CPU, 64 GB memory, 2 TB disk, two compute nodes consist of 64-core CPU, 48 GB memory, 2 TB disk, and one compute node consist 32-core CPU, 64 GB memory, 2 TB disk and four servers are with Ubuntu 14.04 as the operating system.

Software specifications are listed in Table 4.2. The OpenStack version is Kilo released on 30 April 2015. The PHP version is 5.5.9. The SNMP version is 5.7.2.

Table 4.1: Hardware specification

Host Name	CPU	RAM	HDD	OS
Controller node	12-core	64GB	2TB	Ubuntu14.04
Computing node1	64-core	48GB	2TB	Ubuntu14.04
Computing node2	64-core	48GB	2TB	Ubuntu14.04
Computing node3	32-core	64GB	2TB	Ubuntu14.04

Table 4.2: Software specification

Software	OpenStack	Python	PHP	SNMP	NFS
Version	Kilo	2.7.6	5.5.9	5.7.2	4

4.2 Experimental Results and Discussion

First we do the test of the relationship between power consumption of physical machine and CPU utilization of virtual machine. Show as figure, we can see that before 50% of CPU utilization of virtual machine is linear growth, and growth rate is larger, after 50%, growth rate is gentler. We use the result of this experiment to be the power consumption basis to our other experiment.

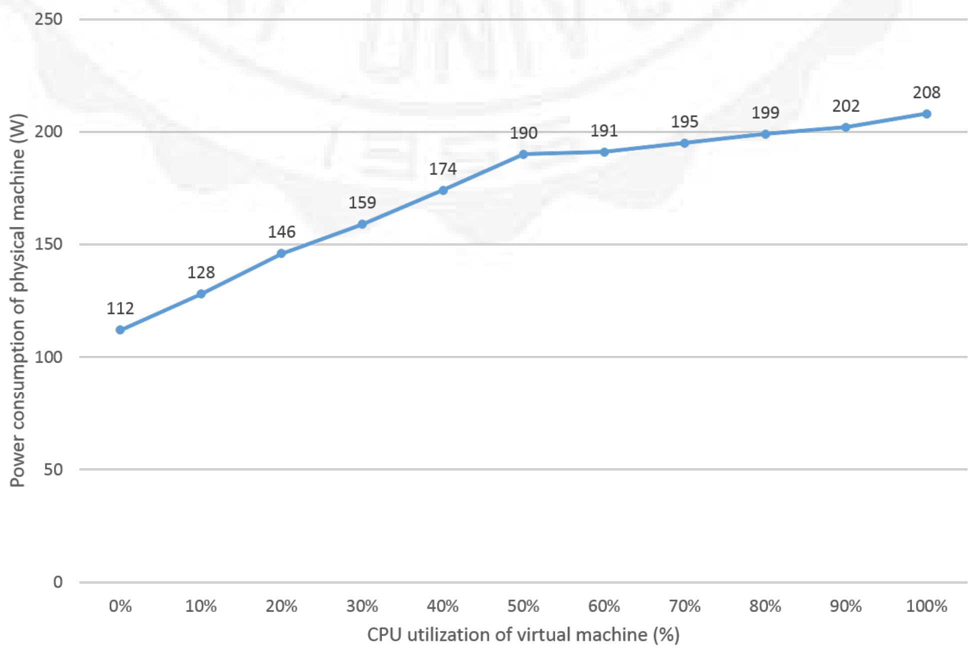


Figure 4.1: Relationship between power consumption of physical machine and CPU utilization of virtual machine

We test DLB method. We deploy some virtual machines on every compute nodes, totally eight virtual machines, all virtual machines consist of 4-core CPU, 4 GB memory, 20 GB disk, and with Ubuntu 14.04 as the operating system. Then run the program to occupy varying resource utilization to make different power consumption. To run DLB method after deploying, show as Figure 4.2 and Figure 4.3. Before DLB, the power consumption of every compute nodes is not equal, there have high consumption and low consumption, after DLB, the power consumption of every compute nodes is very similar. The result represent that it can achieve power load balancing by our method.Figure 4.2 is the power consumption after live migration will be like that theoretically, Figure 4.3 is the power consumption after live migration will be like that actually, there has a little different is because that the power consumption of every virtual machine on different physical machine will be a little different, but they are almost equal, to show that our method is feasible.

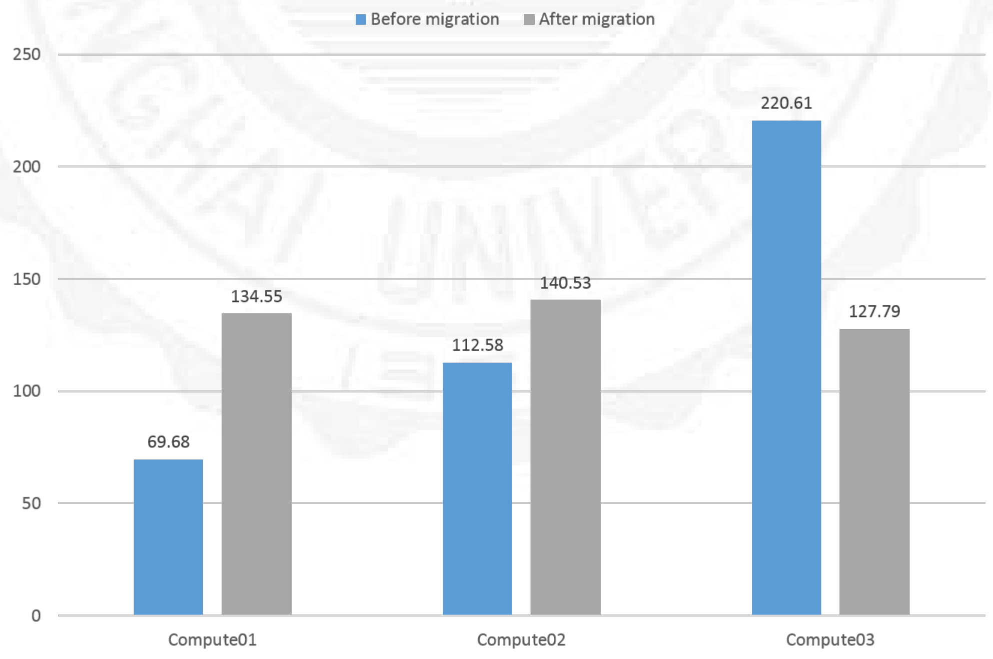


FIGURE 4.2: Power consumption of each nodes before DLB and after DLB (Theoretically)

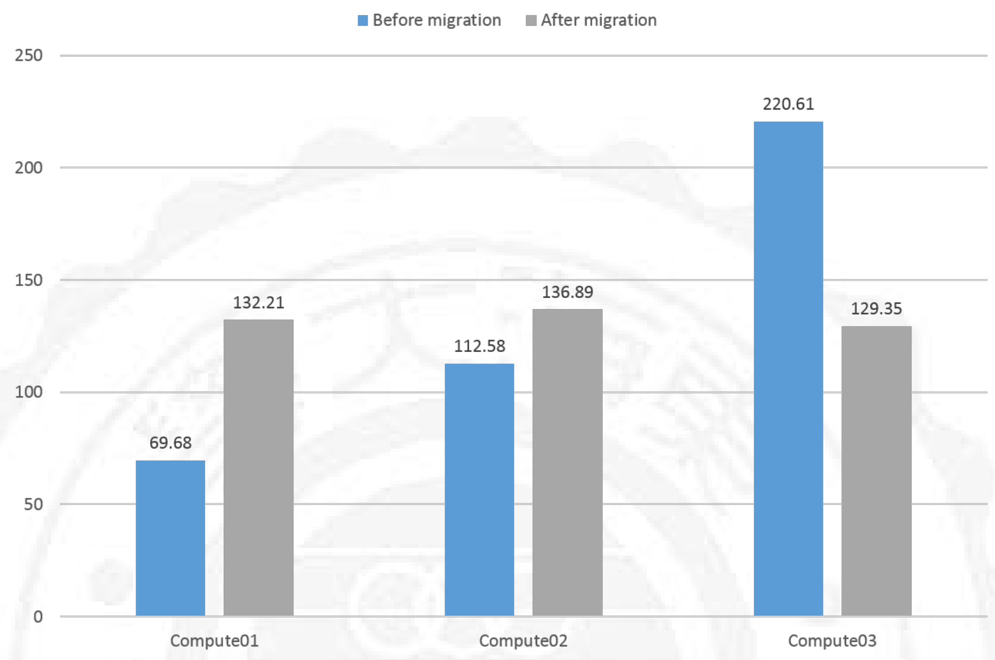


FIGURE 4.3: Power consumption of each nodes before DLB and after DLB (actually)

Then we test Power Saving method. We deploy some virtual machines on every compute nodes, totally eight virtual machines, all virtual machines consist of 4-core CPU, 4 GB memory, 20 GB disk, and with Ubuntu 14.04 as the operating system. Then run the program to occupy varying resource utilization to make different power consumption. To run Power Saving method after deploying, and we choose to shutdown compute3.

Figure 4.4 shows that there have two virtual machines on compute1, two virtual machines on compute2, and four virtual machines on compute3 before Power Saving method, after we run power saving method, it moves out all virtual machines on compute3 then distribute on other two compute nodes.

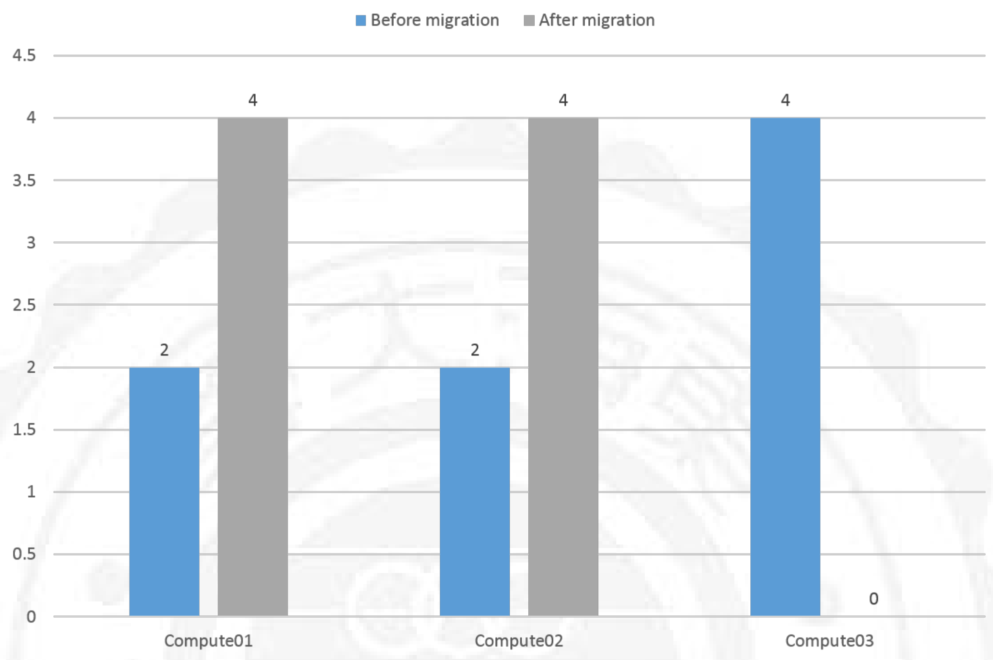


FIGURE 4.4: Number of VMs on each compute nodes

Figure 4.5 shows that the power consumption on each compute node, the power consumption of compute1 and compute2 has be increased because the virtual machines on compute3 have distributed on compute1 and compute2, and the power consumption of compute3 become zero because we shutdown compute3.

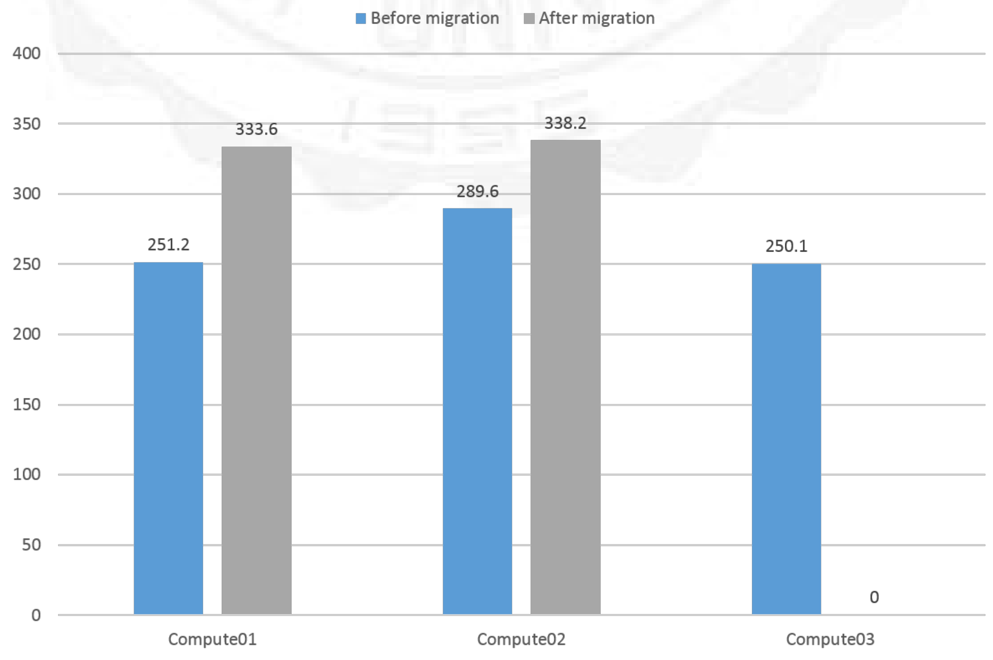


FIGURE 4.5: Power consumption of each compute nodes

Figure 4.6 shows the total power consumption of three compute nodes, it reduce almost 120W because we shutdown compute3, and it shows that it really can save power by our Power Saving method.

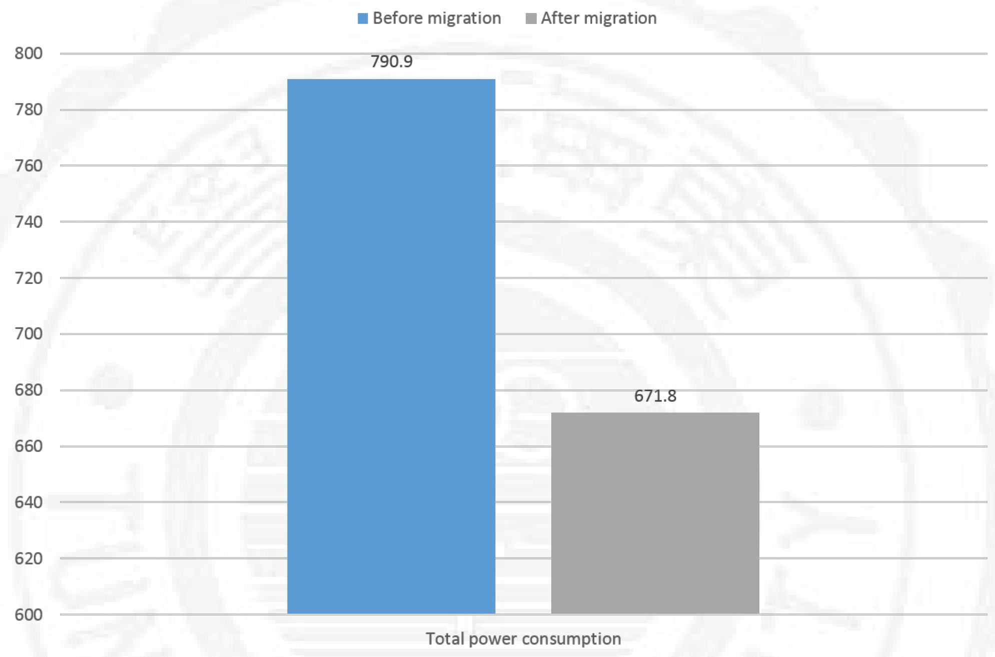
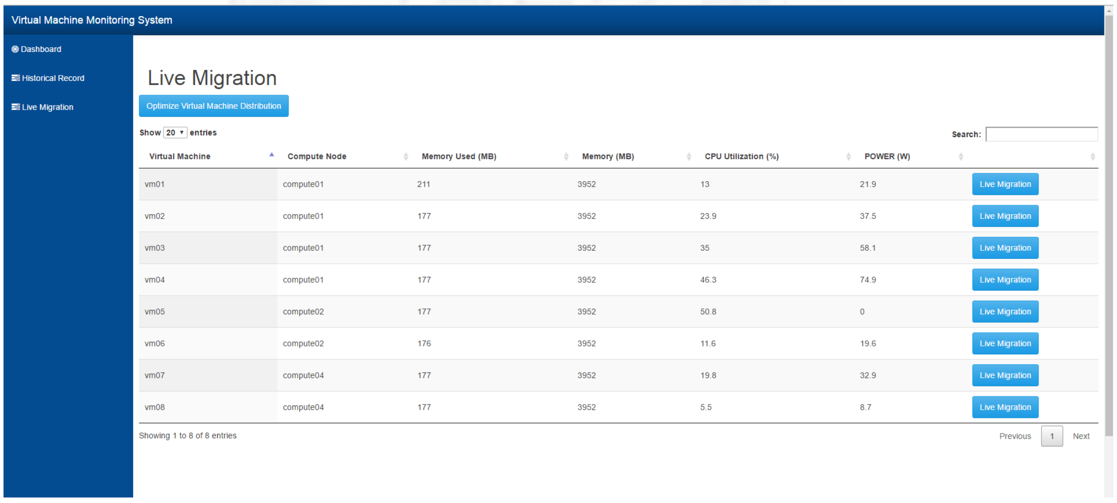


FIGURE 4.6: Total Power consumption of compute nodes

Figure show the interface of live migration on web site, it can choose that just live migrate one VM or auto live migration by the two method. It can choose which method that user wants to use by click Optimize Virtual Machine Distribution.



Virtual Machine Monitoring System
Dashboard
Historical Record
Live Migration
Optimize Virtual Machine Distribution
Show 20 entries
Search:

Live Migration

Virtual Machine	Compute Node	Memory Used (MB)	Memory (MB)	CPU Utilization (%)	POWER (W)	
vm01	compute01	211	3952	13	21.9	Live Migration
vm02	compute01	177	3952	23.9	37.5	Live Migration
vm03	compute01	177	3952	35	58.1	Live Migration
vm04	compute01	177	3952	46.3	74.9	Live Migration
vm05	compute02	177	3952	50.8	0	Live Migration
vm06	compute02	176	3952	11.6	19.6	Live Migration
vm07	compute04	177	3952	19.8	32.9	Live Migration
vm08	compute04	177	3952	5.5	8.7	Live Migration

Showing 1 to 8 of 8 entries

FIGURE 4.7: Live migration interface on the web site

Chapter 5

Conclusions and Future Work

In this thesis, we implement an energy saving cloud infrastructure with virtual machine monitoring and live migration with Distributed Load Balancing method and Power Saving method.We will describe our conclusion in section 5.1 and future work in section 5.2.

5.1 Conclusions

In this thesis, we implement an energy saving cloud infrastructure with virtual machine monitoring and live migration with Distributed Load Balancing method and Power Saving method. We will describe our conclusion in section 5.1 and future work in section 5.2.

In this work, we implement an infrastructure platform base on OpenStack. We achieve our goal by using a DLB method to balance the power consumption of physical machines and an energy saving method to save the energy consumption. The DLB method not only can avoid idle and waste of resources but also can avoid reducing machine life because of the physical machines always keep in high usage. The energy saving method works mainly through shutting physical machines to save energy consumption.

Through the DLB method we can find the most suitable power load for the physical machines. After running DLB method, we will distribute all VMs to physical machines to achieve the suitable power load. The energy saving method can distribute the VMs on all compute nodes to some compute nodes and shut down one compute node to achieve the energy saving purpose.

In this thesis, we achieve our goal to distribute the power load on the physical machines. The experimental results show that we save 120 W by the energy saving method.

5.2 Future Work

The DLB method and the Power Saving method still have some work to do. We plan to combine OpenStack with Docker to achieve speed up the time of deploying VM. On the web site, we plan to have more functions, i.e building virtual machine on the web site, have more detail of virtual machine, and management of compute node. In the future work, we will not only continue studying the DLB method and the Power Saving method to improve our system, but also apply our system to a larger environment.

References

[1] Cloud computing, 2015. `https://www.ibm.com/cloud-computing/what-is-cloud-computing`.

[2] What is cloud computing?, 2015. `http://www.ibm.com/cloud-computing/us/en/what-is-cloud-computing.html`.

[3] Cloud open lab, 2012. `http://www.cloudopenlab.org.tw/ccipo_industryDefinition.do`.

[4] Rajkumar Buyya, Christian Vecchiola, and S. Thamarai Selvi. *Mastering Cloud Computing: Chapter 4 – Cloud Computing Architecture.* MORGAN KAUFMANN, 2013.

[5] Rajkumar Buyya, Christian Vecchiola, and S. Thamarai Selvi. *Mastering Cloud Computing: Chapter 3 – Virtualization.* MORGAN KAUFMANN, 2013.

[6] Xiaofei Liao, Hai Jin, Shizhan Yu, and Yu Zhang. A novel memory allocation scheme for memory energy reduction in virtualization environment. *Journal of Computer and System Sciences*, 81:3 – 15, 2015.

[7] Yaozu Dong, Xiantao Zhang, Jinquan Dai, and Haibing Guan. Hyvi: A hybrid virtualization solution balancing performance and manageability. *Parallel and Distributed Systems*, 25:2332 – 2341, 2014.

[8] *Chapter 7 –Multicore Virtualization.* Multicore Software Development Techniques, 2016.

[9] Hypervisor, 2015. https://technet.microsoft.com/zh-tw/magazine/hh802393.aspx.

[10] Aristide Fattori, Andrea Lanzi, Davide Balzarotti, and Engin Kirda. Hypervisor-based malware protection with accessminer, 2015.

[11] Openstack open source cloud computing software, 2015. http://www.openstack.org/.

[12] What is openstack?, 2015. http://opensource.com/resources/what-is-openstack.

[13] Openstack, 2015. http://docs.openstack.org/kilo/install-guide/install/apt/content/.

[14] Zhaojun Li, Haijiang Li, Xicheng Wang, and Keqiu Li. A generic cloud platform for engineering optimization based on openstack. *Advances in Engineering Software*, 75:42 – 57, 2014.

[15] Luca Abeni, Csaba Kiraly, Nanfang Li, and Andrea Bianco. On the performanc of kvm-based virtual routers. *Computer Communications*, 70:40 –53, 2015.

[16] Hai Jin, Li Deng, Song Wua, Xuanhua Shia, Hanhua Chena, and Xiaodong Panc. Mecom: Live migration of virtual machines by adaptively compressing memory pages. *Future Generation Computer Systems*, 38:23 – 25, 2014.

[17] Mattias Forsman, Andreas Glad, Lars Lundberg, and Dragos Ilie. Algorithms for automated live migration of virtual machines. *Journal of Systems and Software*, 101:110 – 126, 2015.

[18] Muhammad Atif and Peter Strazdins. Adaptive parallel application resource remapping through the live migration of virtual machines. *Future Generation Computer Systems*, 37:148 – 161, 2014.

[19] Hai Jin, Wei Gao, Song Wu, Xuanhua Shi, Xiaoxin Wu, and Fan Zhou. Optimizing the live migration of virtual machine by cpu scheduling. *Journal of Network and Computer Applications*, 34:1088 – 1096, 2011.

[20] Kejiang Ye, Xiaohong Jiang, Ran Ma, and Fengxi Yan. Vc-migration: Live migration of virtual clusters in the cloud. In *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, pages 209–218, Sept 2012.

[21] Shaoming Guo, Wang Yang, and Guojun Wang. Nfs protocol performance analysis and improvement for mobile transparent computing. *High Performance Computing and Communications 2013 IEEE International Conference on*, pages 1878 – 1883, 2013.

[22] Network file system, 2015. `http://nfs.sourceforge.net/`.

[23] Alex Osadzinski. *The Network File System (NFS).* Computer Standards and Interfaces, 1988-1989.

[24] Power distribution unit, 2015. `http://en.wikipedia.org/wiki/Power_distribution_unit`.

[25] What is power distribution unit?, 2013. `http://searchdatacenter.techtarget.com/definition/power-distribution-unit-PDU`.

[26] Jung-Chun Liu Yi-Wei Su Chao-Tung Yang, Kuan-Lung Huang and William Cheng-Chung Chu. Implementation of a power saving method for virtual machine management in cloud. *2013 International Conference on Cloud Computing and Big Data*, 2013.

[27] Jung-Chung Liu Chien-Chih Chen Chao-Tung Yang, Chih-Liang Chuang and William C. Chu. Implementation of cloud infrastructure monitor platform with power saving method. *2015 29th International Conference on Advanced Information Networking and Applications Workshops*, 2015.

[28] Jung-Chun Liu Shuo-Tsung Chen Chien-Chih Chen, Chao-Tung Yang. *Implementation of a Cloud Energy Saving System with Virtual Machine Dynamic Resource Allocation Method Based on OpenStack.* 2015.

[29] Kuan-Lung Huang Fuu-Cheng Jiang Chao-Tung Yang, Jung-Chun Liu. A method for managing green power of a virtual machine cluster in cloud. *Future Generation Computer Systems, Volume 37, July 2014*, pages 26 – 36, 2014.

Appendix A

OpenStack Installation

I. Network Time Protocol (NTP)

```
$ apt-get install ntp
$ service ntp restart
```

II. Database (Controller node setup)

```
$ apt-get install -y mariadb-server python-mysqldb
#===== MySQL configure =====
[mysqld]
...
bind-address = Your_IP
[mysqld]
...
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8

$ service mysql restart
$ mysql_secure_installation
Set root password? [Y/n] Y
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

III. Database (Compute node setup)

```
$ apt-get install python-mysqldb
```

IV. MySQL Setting

```
$ mysql -u root -p
CREATE DATABASE keystone;
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
exit
$ mysql -u root -p
CREATE DATABASE glance;
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
exit
$ mysql -u root -p
CREATE DATABASE nova;
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
exit
```

V. Messaging Server

```
$ apt-get install rabbitmq-server
$ rabbitmqctl change_password guest YOUR_RABBIT_PASS
```

VI. Identity Service Install and Configure

```
$ apt-get install -y keystone python-openstackclient apache2 libapache2-mod-wsgi
memcached python-memcache
$ openssl rand -hex 10
#===== KeyStone configure =====
#Edit /etc/keystone/keystone.conf
[DEFAULT]
```

```
...
admin_token = ADMIN_TOKEN
...
verbose = True
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://keystone:KEYSTONE_DBPASS@controller/keystone
...
[memcache]
# Memcache servers in the format of "host:port". (list value)
servers = localhost:11211
[token]
provider = keystone.token.providers.uuid.Provider
driver = keystone.token.persistence.backends.memcache.Token
[revoke]
...
driver = keystone.contrib.revoke.backends.sql.Revoke

$ su -s /bin/sh -c "keystone-manage db_sync" keystone
#===== Wsgi-KeyStone configure =====
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone
     display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /var/www/cgi-bin/keystone/main
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    <IfVersion >= 2.4>
      ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    LogLevel info
    ErrorLog /var/log/apache2/keystone-error.log
    CustomLog /var/log/apache2/keystone-access.log combined
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone
     display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /var/www/cgi-bin/keystone/admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    <IfVersion >= 2.4>
      ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    LogLevel info
    ErrorLog /var/log/apache2/keystone-error.log
    CustomLog /var/log/apache2/keystone-access.log combined
</VirtualHost>

$ ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled
$ mkdir -p /var/www/cgi-bin/keystone
$ curl http://git.openstack.org/cgit/openstack/keystone/plain/httpd/keystone.py?
h=stable/kilo | tee /var/www/cgi-bin/keystone/main /var/www/cgi-bin/keystone/admin
$ chown -R keystone:keystone /var/www/cgi-bin/keystone
$ chmod 755 /var/www/cgi-bin/keystone/*
$ service apache2 restart
$ rm /var/lib/keystone/keystone.db


$ export OS_SERVICE_TOKEN=ADMIN_TOKEN
$ export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0
$ openstack service create \
  --name keystone --description "OpenStack Identity" identity
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Identity               |
| enabled     | True                             |
| id          | 4beecffff2554875bb21e36a3e9732cc |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+

$ openstack endpoint create \
  --publicurl http://CONTROLLER_IP:5000/v2.0 \
  --internalurl http://CONTROLLER_IP:5000/v2.0 \
  --adminurl http://CONTROLLER_IP:35357/v2.0 \
  --region RegionOne \
  identity
+--------------+-----------------------------------+
| Field        | Value                             |
+--------------+-----------------------------------+
| adminurl     | http://192.168.245.132:35357/v2.0 |
| id           | 427c9f134e56499bbb43aaef910d4951  |
| internalurl  | http://192.168.245.132:5000/v2.0  |
| publicurl    | http://192.168.245.132:5000/v2.0  |
| region       | RegionOne                         |
| service_id   | 4beecffff2554875bb21e36a3e9732cc  |
| service_name | keystone                          |
| service_type | identity                          |
+--------------+-----------------------------------+
$ openstack project create --description "Admin Project" admin
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Admin Project                    |
| enabled     | True                             |
| id          | c56fe7a817ab4abfa18342b142121b5f |
| name        | admin                            |
+-------------+----------------------------------+

$ openstack user create --password-prompt admin
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | 1999e067fd75481a8a6dd00a030ece12 |
| name     | admin                            |
| username | admin                            |
+----------+----------------------------------+

$ openstack role create admin
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 86b5d0acb87e4b85b3250b23365932a4 |
| name  | admin                            |
+-------+----------------------------------+

$ openstack role add --project admin --user admin admin
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 86b5d0acb87e4b85b3250b23365932a4 |
| name  | admin                            |
+-------+----------------------------------+

$ openstack project create --description "Service Project" service
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| enabled     | True                             |
| id          | c503c03fc76f4916a04a1fefecc8bb61 |
| name        | service                          |
+-------------+----------------------------------+

$ unset OS_TOKEN OS_URL
$ openstack --os-auth-url http://CONTROLLER_IP:35357 \
  --os-project-name admin --os-username admin --os-auth-type password \
  token issue
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2015-09-04T07:41:45Z             |
| id         | 4e6a719c159f493c87052c3e1c6fb21c |
| project_id | c56fe7a817ab4abfa18342b142121b5f |
| user_id    | 1999e067fd75481a8a6dd00a030ece12 |
+------------+----------------------------------+

$ openstack --os-auth-url http://CONTROLLER_IP:35357 \
  --os-project-domain-id default --os-user-domain-id default \
  --os-project-name admin --os-username admin --os-auth-type password \
  token issue
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2015-09-04T07:43:23.197410Z      |
| id         | 1e5e605514d644bd89e59301ba77c1d8 |
| project_id | c56fe7a817ab4abfa18342b142121b5f |
| user_id    | 1999e067fd75481a8a6dd00a030ece12 |
+------------+----------------------------------+

$ openstack --os-auth-url http://CONTROLLER_IP:35357 \
  --os-project-name admin --os-username admin --os-auth-type password \
  project list
+----------------------------------+---------+
| ID                               | Name    |
+----------------------------------+---------+
| c56fe7a817ab4abfa18342b142121b5f | admin   |
| c503c03fc76f4916a04a1fefecc8bb61 | service |
+----------------------------------+---------+

$ openstack --os-auth-url http://CONTROLLER_IP:35357 \
  --os-project-name admin --os-username admin --os-auth-type password \
  user list
+----------------------------------+-------+
| ID                               | Name  |
+----------------------------------+-------+
| 1999e067fd75481a8a6dd00a030ece12 | admin |
+----------------------------------+-------+

$ openstack --os-auth-url http://CONTROLLER_IP:35357 \
  --os-project-name admin --os-username admin --os-auth-type password \
  role list
+----------------------------------+-------+
| ID                               | Name  |
+----------------------------------+-------+
| 86b5d0acb87e4b85b3250b23365932a4 | admin |
+----------------------------------+-------+

# edit admin-openrc.sh

export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://CONTROLLER_IP:35357/v3

$ source admin-openrc.sh
$ openstack token issue
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2015-09-04T07:55:15.597454Z      |
| id         | 4f83041d35204fc281c89530a7bec5c5 |
| project_id | c56fe7a817ab4abfa18342b142121b5f |
| user_id    | 1999e067fd75481a8a6dd00a030ece12 |
+------------+----------------------------------+
```

VII. Add the Image service

```
$ mysql -u root -p

MariaDB [(none)]> CREATE DATABASE glance;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
    ->   IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
    ->   IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> exit

$ source admin-openrc.sh

$ openstack user create --password-prompt glance
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | ef1c53829ba74308ba6c3f7ff6d685a2 |
| name     | glance                           |
| username | glance                           |
+----------+----------------------------------+

$ openstack role add --project service --user glance admin
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | cd2cb9a39e874ea69e5d4b896eb16128 |
| name  | admin                            |
+-------+----------------------------------+

$ openstack service create --name glance \
   --description "OpenStack Image service" image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image service          |
| enabled     | True                             |
| id          | 0264a9d23f864ca4b7421dd490d69965 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+

$ openstack endpoint create \
  --publicurl http://CONTROLLER_IP:9292 \
  --internalurl http://CONTROLLER_IP:9292 \
  --adminurl http://CONTROLLER_IP:9292 \
  --region RegionOne \
  image
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| adminurl     | http://192.168.245.132:9292      |
| id           | 1d0d28b9ff404899aed319a5d75eb55b |
| internalurl  | http://192.168.245.132:9292      |
| publicurl    | http://192.168.245.132:9292      |
| region       | RegionOne                        |
| service_id   | 0264a9d23f864ca4b7421dd490d69965 |
| service_name | glance                           |
| service_type | image                            |
+--------------+----------------------------------+

$ apt-get install -y glance python-glanceclient

# edit /etc/glance/glance-api.conf

[DEFAULT]
```

```
verbose = True
notification_driver = noop

[database]
...
connection = mysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]
auth_uri = http://CONTROLLER_IP:5000
auth_url = http://CONTROLLER_IP:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
flavor=keystone

[glance_store]
default_store = file
filesystem_store_datadir = /var/lib/glance/images/

# edit /etc/glance/glance-registry.conf

[DEFAULT]
verbose = True
notification_driver = noop

[database]
...
connection = mysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]
auth_uri = http://CONTROLLER_IP:5000
auth_url = http://CONTROLLER_IP:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
flavor=keystone

$ su -s /bin/sh -c "glance-manage db_sync" glance
$ service glance-registry restart
$ service glance-api restart
$ rm -f /var/lib/glance/glance.sqlite

$ echo "export OS_IMAGE_API_VERSION=2" | tee -a admin-openrc.sh demo-openrc.sh
$ source admin-openrc.sh
$ mkdir /tmp/images
$ wget -P /tmp/images http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64
  -disk.img

$ glance image-create --name "cirros-0.3.4-x86_64" \
  --file /tmp/images/cirros-0.3.4-x86_64-disk.img \
  --disk-format qcow2 --container-format bare \
  --visibility public --progress
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | ee1eca47dc88f4879d8a229cc70a07c6     |
| container_format | bare                                 |
| created_at       | 2015-09-04T07:44:30Z                 |
| disk_format      | qcow2                                |
| id               | bd9943e7-e341-48fb-8fc3-5b462b009379 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | cirros-0.3.4-x86_64                  |
| owner            | c56fe7a817ab4abfa18342b142121b5f     |
| protected        | False                                |
| size             | 13287936                             |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2015-09-04T07:44:30Z                 |
| virtual_size     | None                                 |
| visibility       | public                               |
+------------------+--------------------------------------+

$ glance image-list
+--------------------------------------+---------------------+
| ID                                   | Name                |
+--------------------------------------+---------------------+
| bd9943e7-e341-48fb-8fc3-5b462b009379 | cirros-0.3.4-x86_64 |
+--------------------------------------+---------------------+

$ rm -r /tmp/images
```

VIII. Add the Compute service (Controller node)

```
$ mysql -u root -p
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
    ->   IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
    ->   IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> exit

$ source admin-openrc.sh

$ openstack user create --password-prompt nova
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | 83bf24b81f194937bcf79d83bcf211ee |
| name     | nova                             |
| username | nova                             |
+----------+----------------------------------+

$ openstack role add --project service --user nova admin
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 86b5d0acb87e4b85b3250b23365932a4 |
| name  | admin                            |
+-------+----------------------------------+

$ openstack service create --name nova \
  --description "OpenStack Compute" compute
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Compute                |
| enabled     | True                             |
| id          | 3ce194e9f14142c1beb86fdd21f77a54 |
| name        | nova                             |
| type        | compute                          |
+-------------+----------------------------------+

$ openstack endpoint create \
  --publicurl http://CONTROLLER_IP:8774/v2/%\(tenant_id\)s \
  --internalurl http://CONTROLLER_IP:8774/v2/%\(tenant_id\)s \
  --adminurl http://CONTROLLER_IP:8774/v2/%\(tenant_id\)s \
  --region RegionOne \
  compute
+--------------+-------------------------------------------+
| Field        | Value                                       |
+--------------+---------------------------------------------+
| adminurl     | http://192.168.245.132:8774/v2/%(tenant_id)s |
| id           | e2ee483a875f498fad3e8e6893c67a8d            |
| internalurl  | http://192.168.245.132:8774/v2/%(tenant_id)s |
| publicurl    | http://192.168.245.132:8774/v2/%(tenant_id)s |
| region       | RegionOne                                   |
| service_id   | 3ce194e9f14142c1beb86fdd21f77a54            |
| service_name | nova                                        |
| service_type | compute                                     |
+--------------+---------------------------------------------+

$ apt-get install -y nova-api nova-cert nova-conductor nova-consoleauth \
  nova-novncproxy nova-scheduler python-novaclient

# edit /etc/nova/nova.conf
[DEFAULT]
verbose = True
...
rpc_backend = rabbit
auth_strategy = keystone
my_ip = CONTROLLER_IP
vncserver_listen = CONTROLLER_IP
vncserver_proxyclient_address = CONTROLLER_IP

[database]
connection = mysql://nova:NOVA_DBPASS@controller/nova

[oslo_messaging_rabbit]
rabbit_host = CONTROLLER_IP
rabbit_userid = openstack
rabbit_password = RABBIT_PASS

[keystone_authtoken]
auth_uri = http://CONTROLLER_IP:5000
auth_url = http://CONTROLLER_IP:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = NOVA_PASS

[glance]
host = CONTROLLER_IP

[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

```
$ su -s /bin/sh -c "nova-manage db sync" nova
# edit nova-restart.sh
service nova-api restart
service nova-cert restart
service nova-consoleauth restart
service nova-scheduler restart
service nova-conductor restart
service nova-novncproxy restart

$ sh nova-restart.sh
$ rm -f /var/lib/nova/nova.sqlite
```

IX. Add the Compute service (Compute node)

```
$ apt-get -y install nova-compute sysfsutils
# edit /etc/nova/nova.conf
[DEFAULT]
verbose = True
...
rpc_backend = rabbit
auth_strategy = keystone
my_ip = COMPUTE_HOST_IP
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = COMPUTE_MANAGEMENT_IP
novncproxy_base_url = http://CONTROLLER_IP:6080/vnc_auto.html

[oslo_messaging_rabbit]
rabbit_host = CONTROLLER_IP
rabbit_userid = openstack
rabbit_password = RABBIT_PASS

[keystone_authtoken]
auth_uri = http://CONTROLLER_IP:5000
auth_url = http://CONTROLLER_IP:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = NOVA_PASS

[glance]
host = CONTROLLER_IP
```

```
[oslo_concurrency]
lock_path = /var/lib/nova/tmp

$ egrep -c '(vmx|svm)' /proc/cpuinfo
$ rm -f /var/lib/nova/nova.sqlite
```

X. Legacy Networking (nova-network)

```
#===== Network configure (Controller node) =====
#Edit /etc/nova/nova.conf file
[DEFAULT]
...
network_api_class = nova.network.api.API
security_group_api = nova

$ service nova-api restart
$ service nova-scheduler restart
$ service nova-conductor restart

#===== Network configure (Compute node) =====
$ apt-get install -y nova-network nova-api-metadata
#Edit /etc/nova/nova.conf file
[DEFAULT]
...
network_api_class = nova.network.api.API
security_group_api = nova
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver
network_manager = nova.network.manager.FlatDHCPManager
network_size = 254
allow_same_net_traffic = False
multi_host = True
send_arp_for_ha = True
share_dhcp_address = True
force_dhcp_release = True
flat_network_bridge = br100
flat_interface = INTERFACE_NAME
public_interface = INTERFACE_NAME


$ service nova-network restart
$ service nova-api-metadata restart

#===== Create initial network =====
$ source admin-openrc.sh
$ nova network-create demo-net --bridge br100 --multi-host T \
  --fixed-range-v4 203.0.113.24/29
```

```
$ nova net-list
```

XI. Dashboard Installation

```
$ apt-get install -y openstack-dashboard
http://CONTROLLER_IP/horizon
Login : admin / ADMIN_PASS
```

Appendix B

NFS Installation

I. NFS Installation and Configuration (Controller node)

```
$ apt-get install nfs-kernel-server
$ mkdir -p /var/openstack/nfs-storage
#Edit /etc/exports file
/var/openstack/nfs-storage  *(insecure,rw,sync,no_root_squash)

$ service nfs-kernel-server restart
#Edit /etc/nova/nova.conf file
live_migration_flag=VIR_MIGRATE_UNDEFINE_SOURCE,VIR_MIGRATE_PEER2PEER,VIR_MIGRATE_LIVE

$ service nova-api restart
$ service nova-cert restart
$ service nova-consoleauth restart
$ service nova-scheduler restart
$ service nova-conductor restart
$ service nova-novncproxy restart
```

II. NFS Installation and Configuration (Compute node)

```
$ mkdir /var/lib/nova/instances
$ chown nova:nova instances
$ apt-get install nfs-common
$ mount Controller_IP:/var/openstack/nfs-storage /var/lib/nova/instances/
#Edit /etc/fstab file
Controller_IP:/var/openstack/nfs-storage /var/lib/nova/instances/   nfs   defaults
0    0
```

```
#Edit /etc/passwd file
nova:x:107:114::/var/lib/nova:/bin/bash

$ su nova
$ ssh-keygen
# do the free password to login between compute nodes

#Edit /etc/libvirt/libvirtd.conf file
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"

#Edit /etc/init/libvirt-bin.conf file
env libvirtd_opts="-d -l"

#Edit /etc/default/libvirt-bin file
libvirtd_opts=" -d -l"

$ uuidgen
# through the uuidgen will get the code
#Edit /etc/libvirt/libvirtd.conf file
host_uuid=code

$ service libvirt-bin restart

#Edit /etc/nova/nova.conf file
[DEFAULT]
...
live_migration_bandwidth=0
live_migration_retry_count=30
live_migration_flag=VIR_MIGRATE_UNDEFINE_SOURCE,VIR_MIGRATE_PEER2PEER,VIR_MIGRATE_LIVE

$ service nova-compute restart
```

Appendix C

Programming Codes

I. Distributed Load Balancing method

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from novaclient import client
import sys, json, time
import subprocess

nova = client.Client(2, "admin", "password", "admin", "http://IP:5000/v2.0")

sev = nova.servers.list()
hyp = nova.hypervisors.list()


def migration(instance=None, hostname=None):

  if instance != None or hostname != None:
    host_info = []
    host_info.append( ['compute01', 'COMPUTE01_ID'] )
    host_info.append( ['compute02', 'COMPUTE02_ID'] )
    host_info.append( ['compute03', 'COMPUTE03_ID'] )


    lStart = time.time()

    for i in range( len(host_info) ):
      if host_info[i][1] == instance.hostId:
        #print ''
        if host_info[i][0] == hostname:
          print ""

        else:
          instance.live_migrate(hostname, block_migration=False, disk_over_commit=False)
          name = instance.id
          ninstance = nova.servers.get(name)
          while ninstance.status != "ACTIVE":
            ninstance = nova.servers.get(name)
            time.sleep(1)
          lEnd = time.time()

          print ""

        break
  else:
    print 'migration error'



def bubble_sort( input_obj ):

  for i in range( len(input_obj['vm_name']) ):
    for j in range( len(input_obj['vm_name'])-1-i ):
      if input_obj['vm_power'][j] < input_obj['vm_power'][j+1]:
        tmp_n = input_obj['vm_name'][j]
        tmp_p = input_obj['vm_power'][j]
        input_obj['vm_name'][j] = input_obj['vm_name'][j+1]
        input_obj['vm_power'][j] = input_obj['vm_power'][j+1]
        input_obj['vm_name'][j+1] = tmp_n
        input_obj['vm_power'][j+1] = tmp_p
  return input_obj

def distribution( json_data ):

  pm_num = 3
  pm_name = []
  power_sum = []
  power_sum_tmp = []

  pm_name.append('compute01')
  pm_name.append('compute02')
  pm_name.append('compute04')

  for n in range(pm_num):
    #pm_name.append('compute0'+str(n+1))
    power_sum.append(0)
    power_sum_tmp.append(0)
    #print pm_name[n]
  obj_data = json.loads( json_data )
  pm_dist = {'vm_name': [], 'vm_host': []}


  avg_total = sum( obj_data['vm_power'] ) / pm_num

  obj_data = bubble_sort( obj_data )
  '''
  for i in range( len(obj_data['vm_name']) ):
    print obj_data['vm_name'][i], obj_data['vm_power'][i]
  '''


  for i in range( len(obj_data['vm_name']) ):
    flag = 0
    for j in range( pm_num ):
      power_sum_tmp[j] = power_sum[j]
      power_sum_tmp[j] = power_sum_tmp[j] +  obj_data['vm_power'][i]
      if power_sum_tmp[j] <= avg_total:
        power_sum[j] = power_sum[j] + obj_data['vm_power'][i]
        #print power_sum[j]
        pm_dist['vm_name'].append( obj_data['vm_name'][i] )
        pm_dist['vm_host'].append( pm_name[j] )
        break
      else:
        flag = flag + 1


    if flag == pm_num:

      for j in range( pm_num ):
        power_sum_tmp[j] = power_sum[j]
        power_sum_tmp[j] = power_sum_tmp[j] + obj_data['vm_power'][i]
        power_sum_tmp[j] = abs( power_sum_tmp[j] )

      for j in range( pm_num ):
        if min( power_sum_tmp ) == power_sum_tmp[j]:
          power_sum[j] = obj_data['vm_power'][i]
          pm_dist['vm_name'].append( obj_data['vm_name'][i] )
          pm_dist['vm_host'].append( pm_name[j] )

  return json.dumps( pm_dist )



if sys.argv[1]:
  m_list =  distribution( sys.argv[1] )
  print m_list
  m_list = json.loads( m_list )



  for i in range( len(m_list['vm_name']) ):
    for j in range( len(sev) ):
      if m_list['vm_name'][i] == sev[j].name:
        migration( sev[j], m_list['vm_host'][i] )
        break

else:
  print 'no input'
```

II. Power Saving method

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from novaclient import client
import sys, json, time
import subprocess

nova = client.Client(2, "admin", "password", "admin", "http://IP:5000/v2.0")

sev = nova.servers.list()
hyp = nova.hypervisors.list()


def migration(instance=None, hostname=None):

  if instance != None or hostname != None:
    host_info = []
    host_info.append( ['compute01', 'COMPUTE01_ID'] )
    host_info.append( ['compute02', 'COMPUTE02_ID'] )
    host_info.append( ['compute03', 'COMPUTE03_ID'] )

    lStart = time.time()

    for i in range( len(host_info) ):
      if host_info[i][1] == instance.hostId:

        if host_info[i][0] == hostname:
          print instance.name+";"+hostname+";"+ instance.name+" is on "+hostname+",
          doesn't need to Live migration."
        else:
          instance.live_migrate(hostname, block_migration=False, disk_over_commit=False)
          name = instance.id
        ninstance = nova.servers.get(name)
        while ninstance.status != "ACTIVE":
          ninstance = nova.servers.get(name)
          time.sleep(1)
        lEnd = time.time()
        print instance.name+";"+hostname+";Live migration"+instance.name+"to"+hostname
      break
  else:
    print 'migration error'


def bubble_sort( input_obj ):

  for i in range( len(input_obj['vm_name']) ):
    for j in range( len(input_obj['vm_name'])-1-i ):
      if input_obj['vm_power'][j] < input_obj['vm_power'][j+1]:
        tmp_n = input_obj['vm_name'][j]
        tmp_p = input_obj['vm_power'][j]
        input_obj['vm_name'][j] = input_obj['vm_name'][j+1]
        input_obj['vm_power'][j] = input_obj['vm_power'][j+1]
        input_obj['vm_name'][j+1] = tmp_n
        input_obj['vm_power'][j+1] = tmp_p
  return input_obj

def distribution( json_data, input_hosts ):

  hosts_data = json.loads( input_hosts )

  pm_num = len( hosts_data['host_name'] )

  pm_name = []
  power_sum = []
  power_sum_tmp = []

  for i in range(pm_num):
    pm_name.append( hosts_data['host_name'][i] )

  for n in range(pm_num):

    power_sum.append(0)
    power_sum_tmp.append(0)



  obj_data = json.loads( json_data )
  pm_dist = {'vm_name': [], 'vm_host': []}



  avg_total = sum( obj_data['vm_power'] ) / pm_num
  obj_data = bubble_sort( obj_data )
  '''
  for i in range( len(obj_data['vm_name']) ):
    print obj_data['vm_name'][i], obj_data['vm_power'][i]
  '''


  for i in range( len(obj_data['vm_name']) ):
    flag = 0
    for j in range( pm_num ):

      power_sum_tmp[j] = power_sum[j]

      power_sum_tmp[j] = power_sum_tmp[j] +  obj_data['vm_power'][i]


      if power_sum_tmp[j] <= avg_total:
        power_sum[j] = power_sum[j] + obj_data['vm_power'][i]

        pm_dist['vm_name'].append( obj_data['vm_name'][i] )
        pm_dist['vm_host'].append( pm_name[j] )
        break
      else:

        flag = flag + 1


    if flag == pm_num:

      for j in range( pm_num ):
        power_sum_tmp[j] = power_sum[j]
        power_sum_tmp[j] = power_sum_tmp[j] + obj_data['vm_power'][i]
        power_sum_tmp[j] = abs( power_sum_tmp[j] )

      for j in range( pm_num ):
        if min( power_sum_tmp ) == power_sum_tmp[j]:
          power_sum[j] = obj_data['vm_power'][i]
          pm_dist['vm_name'].append( obj_data['vm_name'][i] )
          pm_dist['vm_host'].append( pm_name[j] )

  return json.dumps( pm_dist )



if sys.argv[1] and sys.argv[2]:
  m_list =  distribution( sys.argv[1], sys.argv[2] )
  print m_list
  m_list = json.loads( m_list )
  for i in range( len(m_list['vm_name']) ):
    for j in range( len(sev) ):
      if m_list['vm_name'][i] == sev[j].name:
        migration( sev[j], m_list['vm_host'][i] )
        break
else:
  print 'no input'
```

Appendix D

Monitor Codes

I. Server and Virtual Machines Information Monitor program

```
# -*- coding: utf-8 -*-
import sys
import os
import atexit
import time
import psutil

print "Loading..."
time.sleep(3)

line_num = 1
ShowInfo="On"
def print_line(str):
 if ShowInfo=='On':
  print str

#function of Get CPU State
def getCPUstate(interval=1):
    return (str(psutil.cpu_percent(interval)))

#function of Get Memory
def getMemorystate():
    phymem = psutil.phymem_usage()
    buffers = getattr(psutil, 'phymem_buffers', lambda: 0)()
    cached = getattr(psutil, 'cached_phymem', lambda: 0)()
    used = phymem.total - (phymem.free + buffers + cached)
    line = " Memory: %5s%% %6s/%s" % (
        phymem.percent,
        str(int(used / 1024 / 1024)) + "M",
        str(int(phymem.total / 1024 / 1024)) + "M"
    )
    return line

def bytes2human(n):
    """
    >>> bytes2human(10000)
    '9.8 K'
    >>> bytes2human(100001221)
    '95.4 M'
    """
    symbols = ('K', 'M', 'G', 'T', 'P', 'E', 'Z', 'Y')
    prefix = {}
    for i, s in enumerate(symbols):
        prefix[s] = 1 << (i+1)*10
    for s in reversed(symbols):
        if n >= prefix[s]:
            value = float(n) / prefix[s]
            return '%.2f %s' % (value, s)
    return '%.2f B' % (n)

def poll(interval):
    """Retrieve raw stats within an interval window."""
    tot_before = psutil.network_io_counters()
    pnic_before = psutil.network_io_counters(pernic=True)
    diskio_before=psutil.disk_io_counters()
    # sleep some time
    time.sleep(interval)
    tot_after = psutil.network_io_counters()
    pnic_after = psutil.network_io_counters(pernic=True)
    diskio_after=psutil.disk_io_counters()
    # get cpu state
    cpu_state = getCPUstate(interval)
    # get memory
    memory_state = getMemorystate()
    return (tot_before, tot_after, pnic_before, pnic_after,\\
        cpu_state,memory_state,diskio_before,diskio_after)

def refresh_window(tot_before, tot_after, pnic_before, pnic_after,\\
        cpu_state,memory_state,diskio_before,diskio_after):
    os.system("cls")
    """Print stats on screen."""
    print_line(time.asctime()+" | "+cpu_state+" | "+memory_state)

    #CPU USED INFO
    CPU_USED=cpu_state
    #RAM USED INFO
    phymem = psutil.phymem_usage()
    buffers = getattr(psutil, 'phymem_buffers', lambda: 0)()
    cached = getattr(psutil, 'cached_phymem', lambda: 0)()
    used = phymem.total - (phymem.free + buffers + cached)
    RAM_PA=phymem.percent
    RAM_USED=str(int(used / 1024 / 1024))
    RAM_ALL=str(int(phymem.total / 1024 / 1024))
    #DISK IO INFO
    DISK_READ=str(diskio_after.read_bytes - diskio_before.read_bytes)
    DISK_WRITE=str(diskio_after.write_bytes - diskio_before.write_bytes)

    #SEND-TO-DB SERVER
    import httplib,urllib
    httpClient = None
    try:
        ServerID="1"
        params = urllib.urlencode({'CPU_USED': CPU_USED, 'RAM_PA': RAM_PA,\\
         'RAM_ALL': RAM_ALL,'RAM_USED':RAM_USED,'SID':ServerID})
        headers = {"Content-type": "application/x-www-form-urlencoded", "Accept":
        "text/plain"}
        httpClient = httplib.HTTPConnection('IP', 80, timeout=3)
        httpClient.request('POST', '/power/API/ServerInfo.php',params,headers)

        #response HTTPResponse
        response = httpClient.getresponse()
        print response.status
        print response.reason
        print response.read()

    except Exception, e:
        print e
    finally:
        if httpClient:
            httpClient.close()
try:
    interval = 0
    while 1:
        args = poll(interval)
        refresh_window(*args)
        interval = 1
except (KeyboardInterrupt, SystemExit):
    pass
```

II. PDU Information program

```
<?php
function get_server_info($host, $community, $objectid) {
$a = snmpget($host, $community, $objectid);

$tmp = explode(":", $a);
if (count($tmp) > 1) {
$a = trim($tmp[1]);
}
return $a;
}
$host="IP";
$community="public";

for($i=1;$i<=8;$i++){
$Power = get_server_info($host,$community,".1.3.6.1.4.1.13742.4.1.2.2.1.7.".$i);
echo $i."-Power:".$Power."<br>";

$I = get_server_info($host,$community,".1.3.6.1.4.1.13742.4.1.2.2.1.4.".$i);
echo $i."-I:".$I."<br>";

$V = get_server_info($host,$community,".1.3.6.1.4.1.13742.4.1.2.2.1.6.".$i);
echo $i."-V:".$V."<br>";

$PF = get_server_info($host,$community,".1.3.6.1.4.1.13742.4.1.2.2.1.9.".$i);
echo $i."-PF:".$PF."<br>";
$sql="INSERT INTO `ServerMonitor`.`Power` (`ID`, `TIME`, `SID`, `V`, `C`, `P`, `PF`) \\
 VALUES (NULL, CURRENT_TIMESTAMP, '".$i."', '".$V."', '".$I."', '".$Power."','".$PF."')
  ;";
//echo $sql;
mysql_query($sql) or die('MySQL query error');
$sql="UPDATE `ServerMonitor`.`PowerRealTime` SET `TIME` = CURRENT_TIMESTAMP(), \\
`V` = '".$V."', `C` = '".$I."', `P` = '".$Power."',\\
 `PF` = '".$PF."' WHERE `PowerRealTime`.`ID` = ".$i.";";
mysql_query($sql) or die('MySQL query error');
}
?>
```