東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

整合 OPENSTACK 和 DOCKER 建構出動態遷移的雲端虛擬化環境

On Construction of Cloud Virtualization for Live Migration by Integration of OpenStack and Docker

研究生: 許家勝

中華民國一〇五年一月

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 許 家 勝 所提之論文

整合 OPENSTACK 和 DOCKER 建構出動態遷移的雲端虛擬化環境

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 楊武 簽章

委 員 江振瑞

時文中

許慶賢

指 導 教 授 楊朝棟 簽章

中華民國 105 年 1 月 7 日

摘　要

在這資訊爆炸，講求便捷、迅速的年代。不管是政府或是企業組織甚或個人，都可能有建構雲端系統的需求，也因此雲端技術的應用和服務與日俱增，成為大數據甚或跨國企業、政府聯盟相繼追求技術的趨勢。為了讓一般使用者在無需花費高額的軟體授權費用下，也能擁有私有雲的服務，因此著重於運用開放原始碼的技術，發展更有效率的雲端技術。在電子計算機學中，虛擬化乃指的是邏輯群組的呈現抑或電腦資源的部分集合。而筆者所提到虛擬化所指的是平台虛擬化，也就是一般人所謂的虛擬機，在此，我的實驗內容是建構了一個雲端虛擬環境並根據開放原始碼的技術加以實驗。本研究主題是以實驗方式在雲端建立虛擬環境，並整合 Docker 和 OpenStack 等開放源始碼軟體，為用戶端提供雲端虛擬化環境。它有效地提供了組織或企業的私有雲解決方案。其重點在於雲端的基礎建設服務模式。就使用者界面而言，此系統可以有效減少使用者存取雲端資源的複雜度，簡化運作的程序。就使用者存取的操作而言，所建構出的網頁介面來管理虛擬機的遷移，是極為簡易的。筆者以虛擬機動態遷移與實體機器和虛擬機的性能等實驗項目，來加以實測，並根據數據分析其結果。顯然，在實驗環境中發現了動態遷移並非如商業廣告所說的不停機轉移，且我們也得到了全虛擬化技術相當貼近實機效能的結果。最終我們成功且完整的建構出一個開放原始碼的私有雲解決方案，並能執行雲端三大服務的結合。

關鍵字：雲端計算、基礎建設即服務;、集裝箱、即時遷移、虛擬機供應。

Abstract

Increasing cloud computing services, whether government or corporate, organizations and even individuals are likely to construct the cloud system itself. This paper focus on the cloud technology in open-source to achieve the private cloud for common people without high software license charge. In computer science, the virtualization is a process of manifestation of the logical group or subset of computer resources. But the virtualization mentioned in this paper is platform virtualization. The benefits of virtualization are numerous. It is usually to learn virtualization before using the cloud technology for enterprise. The constructed environment mentioned in this paper implement virtualization and to experiment.

The subject of this paper is how to construct virtualization in the cloud with integration of OpenStack LXD and Container for the user. It provided the private cloud solution for enterprise or organization. It's focus on the IaaS of three services in cloud. This system can reduce the complexity of the accessing the cloud resources by the user interface. It is easy to mange deployment the VMs by the web-based user interface. This paper measured the data of the live migration and the compare of the physical machine and virtual machine, then analyze results. In the experimental environment, we prove that live migration is not a non-stop service. In addition, we also prove the performance of full virtualization closer to the physical machine. Finally, we had completed the construction of the open source solution for private cloud . And we implement the combination of the three services of the cloud.

Keywords: Cloud Computing, IaaS, KVM, OpenStack, Live migration,LXD,LXC

致謝詞

研究者在東海大學進修時因研修了楊朝棟教授所開的課程「網格運算」，在這個課程中接觸到運算系統架構後感到相當有興趣，原以為所謂的電腦是受限於硬體科技所以是有限的，但透過軟體可以將多台不同電腦間整合為一為一個大資料作運算，也可以一台腦化為多台虛擬機，這讓我決定投入雲端運算的研究當中。

本研究論文我要感謝曾給了我協助的學長與同學，我特別要感謝我的指導教授楊朝棟教授，感謝教授對於一邊工作一邊進修的我們能在晚上特別為我們指導論文的進行，讓我在 HPC 實驗室獲益良多，雖然在這段期間工作與家庭的忙碌使得學習一度中斷，但感謝我的家人給了我支持並鼓勵我在創業之餘回來完成學業，我想表達的是沒有大家的支持與幫助，我無法堅持到最後。

Table of Contents

List of Figures

List of Tables

Chapter 1

簡介

1.1 研究背景

雲端運算是現今作熱門的議題，不管是軟體或是硬體廠商均開始將產品逐漸移轉到雲端，發展雲端相關產品。雲端運算為「使用無所不在、便利、隨需應變的網路，共享廣大的運算資源 (如網路、伺服器、儲存、應用程式與服務)，可透過最少的管理工作及服務供應者互動，快速提供各項服務」，依據美國國家標準與技術研究院 (National Institute of Standards and Technology, NIST) 於 2012 年 5 月發佈 SP 800-146 建議書，雲端模型由五個基本特徵、三個服務模式及四種佈署模型組成，基本三種模式分別為軟體即服務（SaaS）Software as a Service 號、平台即服務（PaaS）Platform as a Service、基礎設施即服務（IaaS）Infrastructure as a Service，要達成這三種服務，我們就必需花費許多軟體與硬體的設備以及相關的技術支援與維，以一般的中小企業或學術單位來說是一個難以獨立完成的設施。

雲端運算是一種運算模式，它提供客戶各種 IT 功能，解決客戶的問題，客戶依照需求訂閱自己需要的功能包含軟體和硬體的需求。對企業來說，採用雲端運算並不會儉省 IT 解決方案的成本，但會降低某些成本，例如硬體成本。此外，採用雲端服務的企業，為了不浪費資源，越來越多也兼作雲端服務的提供者，例如先預租一份大的運算資源 (比較便宜)，再把沒有用到的資源轉租給其

客戶和業務夥伴使用，藉此提供應用、資訊及業務流程服務。基本上雲端不是一種新技術的結合，而是商業模式的創新，雲端技術透過網路把多個成本相對較低的計算實體整合成一個具有強大運算能力的系統；透過運算能力標準化，並借助 IaaS、PaaS、SaaS 等先進的商業模式把強大的運算能力以服務的方式提供給使用者使用。

因此現今雲端運算快速蓬勃發展 [13,15]。有許多公司目前提供雲計算相關的服務，如 Google[30]，Amazon [29]，Yahoo 等公司，數以萬計的伺服器資源用於構建大規模的雲端計算資源的服務與平台，並提供各種服務，如：超大儲存空間，強大的運算能力，無需下載的在線功能，如編輯影像視訊功能。用戶端計算和存儲資源是有限的，用戶可以通過登入後使用他們所需要的雲端運算資源。

1.2 研究動機

在這資訊爆炸，講求便捷、迅速的年代。不管是政府或是企業組織甚或個人，都可能有建構雲端系統的需求，也因此雲端技術的應用和服務與日俱增，成為大數據甚或跨國企業、政府聯盟相繼追求技術的趨勢。

為了讓一般使用者在無需花費高額的軟體授權費用下，也能擁有私有雲的服務，因此著重於運用開放原始碼的技術，發展更有效率的雲端技術。在電子計算機學中，虛擬化乃指的是邏輯群組的呈現抑或電腦資源的部分集合。而筆者所提到虛擬化所指的是平台虛擬化，也就是一般人所謂的虛擬機，在此，我的實驗內容是建構了一個雲端虛擬環境並根據開放原始碼的技術加以實驗。

虛擬化指的是從實際提供服務的實體資源，如硬體、軟體、網路、儲存等軟硬體設備，將這些資源做邏輯上的分割，轉變為邏輯上可以管理的資源，常見的虛擬化如 VMWare、KVM、Xen 與 Hyper-V 等等是以邏輯分割出的系統環境上執行作業系統或系統服務的能力，在系這上面可以安裝任何作業系統，但缺點是不管如何一個虛擬機就背負著一個花費不好資源的作業系統，且開機與關機都與一般裝在實體機上的作業系統一樣需花費許多時間。

傳統虛擬化技術從作業系統層下手，目標是建立一個可以用來執行整套作業系統的沙箱獨立執行環境，習慣以虛擬機器（Virtual Machine）來稱呼。而 Container 技術則是直接將一個應用程式所需的相關程式碼、函式庫、環境配置檔都打包起來建立一個執行環境，為了和傳統虛擬化技術產生的虛擬機器區分，Container 技術產生的環境就稱為 Container。

Docker Container 是一個 Client-Server 架構的應用程式，在一個 Docker 執行環境中，包括了 Docker 用戶端程式、和在背景執行（Daemon）的 Docker 伺服器（也稱為 Docker Engine），另外還有將 Container 封裝後的 Docker 映象檔，用來儲存映象檔的 Registry 服務。官方提供的映象檔 Registry 服務就稱為 Docker Hub，這是類似 Github 程式碼 Repository 儲存服務的映象檔 Repository 儲存服務。

安裝 Docker 之後，會提供了一個命令列的用戶端程式來和在背景執行的 Docker 伺服器溝通。開發者可以直接從 Docker Hub 下載 Docker 映象檔（Docker pull 指令），再執行（Docker run 指令）就可以用這個映象檔來建立一個 Container。

本研究主要重點如下三個要點目的

- 如何使用開源碼 Open Stack LXD(The Legion of Extraordinary Dancers) 與 Docker Container 建立一個雲端運算的解決方案，為用戶端提供雲端虛擬化環境且驗證成功地提供了企業或組織的私有雲解決方案。
- Container 與 VM 以效能的比較。
- 以 Container 取代 VM 的可行性與適用時機。

Chapter 2

研究背景

雲端運算 (Cloud Computing) 在近年來的 IT 領域是一個非常重要的課題，並且開發了許多不同的服務。在實際上，雲計算並不是一個新的技術。這是一個新概念 [1,10,19,23,21,22,24,28,30]。雲端運算是一種電腦運算的概念，最初運用是在網格運算 (Grid Computing)，主要是在解決一些專業且複雜需大量運算的相關問題的解決，而一般人所用到雖然只是一些簡單的應用例如查詢，但因資料量非常的龐大，因此現今雲端運算快速蓬勃發展 [13,15]。有許多公司目前提供雲計算相關的服務，如 Google[30]，Amazon [29]，Yahoo 等公司，數以萬計的伺服器資源用於構建大規模的雲端計算資源的服務與平台，並提供各種服務，如：超大儲存空間，強大的運算能力，無需下載的在線功能，如編輯影像視訊功能。用戶端計算和存儲資源是有限的，用戶可以通過登入後使用他們所需要的雲端運算資源。本文集中在雲計算基礎設施，特別是虛擬機和即時遷移 (Live Migration)[3,4,5,6,7,8,9,11,12,14,18,33,34]。. 依照美國國家標準技術研究所 National Institute of Standards and Technology（NIST）所提出的基本三種模式分別為軟體即服務（SaaS）Software as a Service 號、平台即服務（PaaS）Platform as a Service、基礎設施即服務（IaaS）Infrastructure as a Service，本文以開源碼架設一 IaaS 平台，並在此平台上實現雲計算的三種服務。

2.1 雲端計算 (Cloud Computing)

雲端運算是一種基於網際網路的運算方式，通過這種方式來共享軟硬體資源和資訊可以按需求提供給電腦和其他裝置。雲端運算所提供的是服務而並非產品，通過共享資源和資訊作為設備將是通過網路以計量方式提供服務。[1] 依照 NIST 所定義雲端運算如下：「使用無所不在、便利、隨需應變的網路，共享廣大的運算資源 (如網路、伺服器、儲存、應用程式與服務)，可透過最少的管理工作及服務供應者互動，快速提供各項服務」這種雲模型由五個基本特徵，三個服務模式，四個部署模型。其部署的模式可分為公有雲 (Public Cloud)、私有雲 (Private Cloud)、社群雲端 (Community Cloud) 及混合雲 (Hybrid Cloud)，詳細來說明的話，則是：

五個基本特徵

- (1) 隨需自助服務 (On-demand Self-service)：可以依其需求索取計算資源 (例如伺服器或儲存空間)，且整個過程是單方面自動化的，無須與資源提供者互動。
- (2) 網路使用無所不在 (Broad Network Access)：服務是經由網路提供，且有標準機制能讓不同的客戶端平台 (如智慧型手機及筆電等) 都可以使用。
- (3) 多人共享資源池 (Resource Pooling)：服務者所提供的計算資源，例如儲存空間、網路頻寬、計算能力、虛擬機器數量等，可類比為一個大水池，能隨時依需要 (重新) 分配給不同平台的多個使用者，使用者不需了解資源的實體位置。
- (4) 快速重新部署靈活度 (Rapid Elasticity)：計算資源不僅可以快速且有彈性地被提供或釋放，且對客戶而言，資源是取之不盡且可以依自己的需求購買的。
- (5) 可被監控與量測的服務 (Measured Service)：計算資源可依其所提供的服務特性被自動控管及最佳化。提供者與使用者雙方都可透明地監控資源使用情形。

三個服務模式

- (1) 軟體即服務 (SaaS)：是屬於客戶端所使用的程式，指的是透過網際網路以提供商業應用軟體的一個模式，但沒辦法掌握作業系統，硬體及網路…等資源，例如：Salesforce.com。
- (2) 平台即服務 (PaaS)：是指消費者掌握應用程式的環境，是整合了設計、開發、測試、部署等功能的平台提供給用戶的一個服務，主要是讓開發人員可以透過網路撰寫程式，並依據使用量來進行收費，例如：Google App Engine。
- (3) 基礎架構即服務 (IaaS)：是指消費者跟廠商以租用的方式，使用處理器，儲存容量…等資源，也可以在上面安裝作業系統，架設屬於自己的環境，不需購買硬體及建置基硬設施。例如：Amazon AWS。

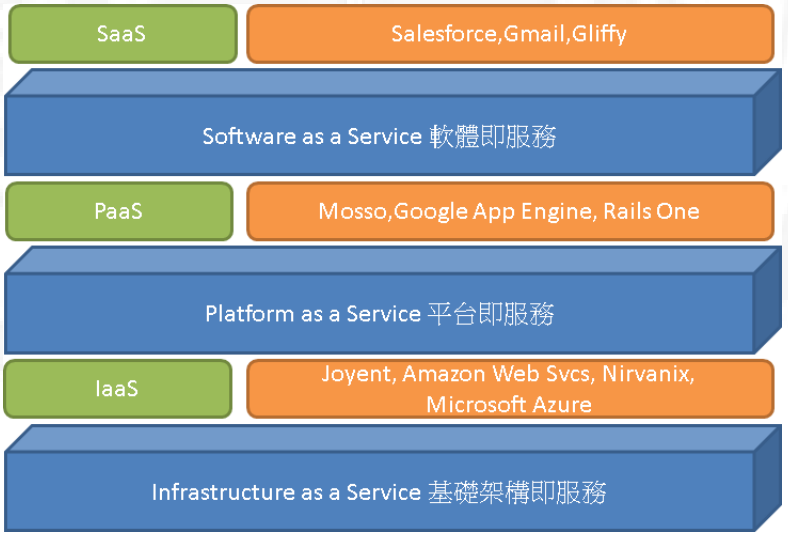

FIGURE 2.1: 雲端服務架構

四個部署模型

- (1) 公有雲 (Public Cloud)：是指第三方提供雲端基礎設施，平台，軟體公開來給使用者使用。
- (2) 私有雲 (Private Cloud)：將雲端基礎設施架構在企業內部，僅供企業內部員工做資料上的共享及使用，是目前企業內最廣泛的作法。

- (3) 社群雲 (Community Cloud): 社群雲由眾多利益相仿的組織掌控及使用，例如特定安全要求、共同宗旨等。社群成員共同使用雲端資料及應用程式。通常社群雲主要使用者大多以學術為主。

- (4) 混合雲 (Hybrid Cloud)：是指一個環境上公有雲及私有雲兩個架構，在公有雲上方便公眾存取使用，同時有保有資料隱密的私有雲，所以保有兩者的特性。

2.2 VM 虛擬化 (Virtualization)

虛擬化指的是從實際提供服務的實體資源，如硬體、軟體、網路、儲存等軟硬體設備，將這些資源做邏輯上的分割，轉變為邏輯上可以管理的資源，並提供給需求者的技術，讓使用者更合理、更充分的控制與管理這些資源。具體而言，虛擬化提供了在邏輯分割出的系統環境上執行作業系統或系統服務的能力，並且無關於特定的實體電腦系統。顯然地，所有這些都必須在任何特定的時間上，在實際的電腦系統上運行，而虛擬化提供了一個邏輯上的抽象層，將應用程式、系統服務，甚至作業系統，提供它們與一個特定的硬體區塊進行綁定。虛擬化，專注於邏輯上的運作環境，但實體上，能讓應用、服務，以及在作業系統中的實體，易於在不同的實體系統上搬移。虛擬化是能夠執行許多作業系統下的應用程式，更有效率地管理 IT，並且與其他電腦分配計算資源。基礎架構虛擬化的好處：

- 網路虛擬化：虛擬機器在任何地方都可以存取。虛擬機器可以移動至任何區域機房 (使用公有雲)。

- 儲存虛擬化：儲存資源集中化，整合異質儲存設備。提升現有儲存設備的可用度與使用率。

- 作業系統虛擬化/虛擬機器：用虛擬化硬體支援舊有的作業系統。集中化管理。備份備援。快速佈建。避免敏感資料外洩。

虛擬化架構，如下圖所示

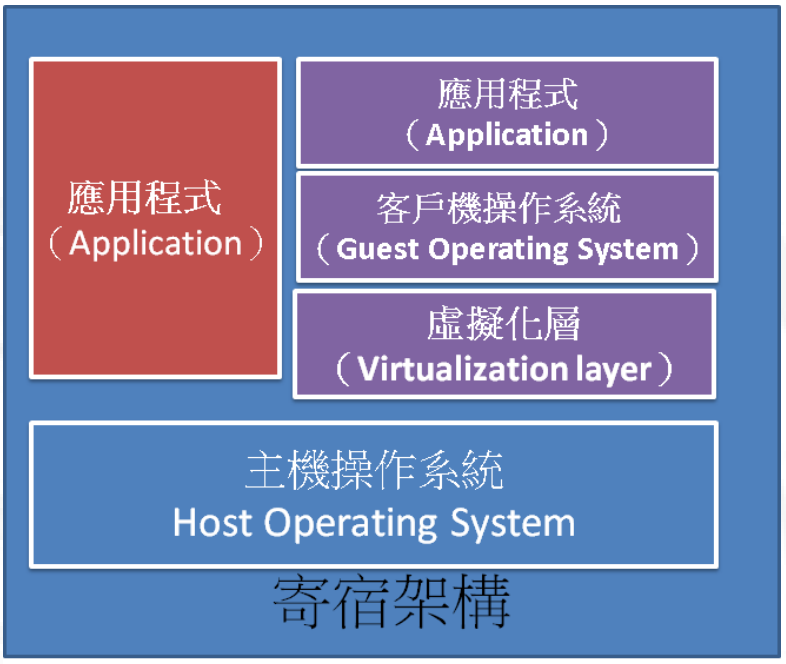

FIGURE 2.2: 虛擬化架構

- 實體機 (Host)：實際存在的電腦硬體。PB（Petabyte）等級。
- 主機操作系統 (Host OS)：實際上存在的電腦作業系統，也就是執行 VM 的電腦作業系統。
- 虛擬化層 (Virtualization Layer)：將電腦的主要資源抽象化，提供 OS 透過抽象層存取。
- 客戶機操作系統 (Gest OS)：客戶機的作業系統，可依需求安裝。
- 應用程式 (Application)：最終要使用的應用程式，例如 SQL Server 或 Web 服務的應用。

VM 虛擬化一般分為二種

- 全虛擬化：以 VMWare 技術而言，它是採用二進位轉譯 (Binary Translation) 技術達成，將因為虛擬化而降級的作業系統 (Ring 1 特權模式) 存取硬體資源時，由 VM Manager 將作業系統發出的 CPU 指令透過二進位轉

譯技術進行轉換進而順利存取硬體資源，簡言之作業系統並不知道自己被調降到 Ring 1 特權模式中。優點：不需要修改作業系統的核心，因此可運作大部份的作業系統種類。缺點：透過二進位轉譯會消耗較多的硬體資源。全虛擬化應用軟體：Virtual PC、VMWare、Virtual Box，KVM。

- 半虛擬化：半虛擬化技術必須修改作業系統核心來植入 Hypercall，使得作業系統不用因為虛擬化而將 CPU 特權等級被調降到 Ring 1 (保持在 Ring 0)，並且透過 Hypercall 來存取硬體資源。優點：此方式對於硬體資源消耗相對較少。缺點：因為必須修改作業系統核心，因此可於半虛擬化平台上運作的作業系統種類較少。半虛擬化應用軟體：Xen。

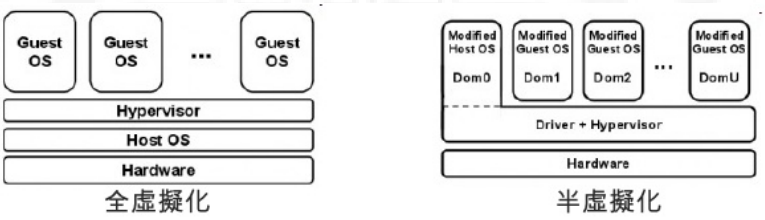

FIGURE 2.3: 全虛擬化與半虛擬化架構比較

2.3 作業系統虛擬化 (Operating system–level virtualization)

一種虛擬化技術，這種技術將作業系統內核虛擬化，可以允許使用者空間軟體物件（Instances）被分割成幾個獨立的單元，在內核中運行，而不是只有一個單一物件運行。於原作業系統上模擬出一個行程，所有的 CPU/RAM/IO 等資源，全部都共用原生的 Host OS，幾乎沒有虛擬硬體的負擔，所以跟在原機上執行的效能幾乎一樣，大約只差 1 - 3 作業系統層虛擬化應用軟體：OpenStack LDX。

2.4 OpenStack

OpenStack 是一個美國國家航空暨太空總署和 Rackspace 合作研發的雲端運算軟體，經由 Apache 許可認證授權，並且是一個開放源始碼的自由軟體，OpenStack 內部包含許多模組，其中包含有運算模組、網路通訊模組、儲存模組以及負責管理以上三個模組的儀表板模組，是一套可以提供 IaaS 服務的軟體，可以提供虛擬機器的方式，供運算資源調度的上彈性利用如下圖

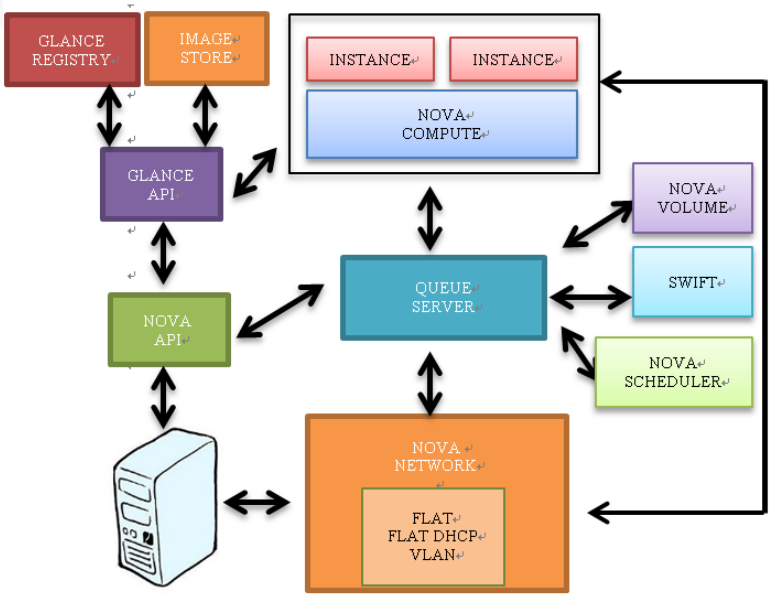

FIGURE 2.4: OpenStack 模組

OpenStack 是一個開放源始碼的雲計算管理平台，主要由幾個元件組合起來完成具體管理平台共享服務，並且 OpenStack 通過各種互補的服務提供了基礎設施即服務（IaaS）的解決方案，每個服務提供 API 以進行集成。可以提供虛擬機器的方式，對外提供運算資源以便彈性擴充或調度。應用程式的開發者，只要透過 API 的呼叫就可以和 OpenStack 溝通，例如用 API 來調整虛擬機器的數量、開機或關機等等，OpenStack 內部再負責運算資源的管理還有各元件內部溝通與協調。OpenStack 支援幾乎所有類型的雲環境目標是提供實施簡單、可大規模擴展、豐富、標準統一的雲計算管理平臺。

OpenStack 雲計算平臺，說明服務廠商和企業內部實現類似於 Amazon EC2 和 S3 的雲基礎架構服務 (Infrastructure as a Service, IaaS)。OpenStack 包含兩個主要模組：Nova 和 Swift，前者是 NASA 開發的虛擬伺服器部署和業務計算模組；後者是 Rackspace 開發的分散式雲存儲模組，兩者可以一起用，也可以分開單獨用。OpenStack 是一個為公共及私有雲的建設與管理提供軟體的開放源始碼的專案。它的社區擁有超過 130 家企業及 1350 位開發者，這些機構與個人都將 OpenStack 作為基礎設施即服務（IaaS）資源的通用前端。OpenStack 專案的首要任務是簡化雲的部署過程並為其帶來良好的可擴展性，前端來設置及管理自己的公共雲或私有雲。 OpenStack 就是一套雲端作業系統（Cloud OS）。就像是在實體電腦上，作業系統可以用來控制實體電腦上的硬體模式，讓底層實體硬體與應用程式完全隔離。作業系統提供共通執行環境，一個可以跨硬體的，並且不讓應用程式受制於不同廠牌、規格的硬體功能。同樣的邏輯，雲端作業系統作為雲端應用程式和實體資料中心的中間層，雲端應用也不再受制於實體資料中心內各種硬體設備的侷限，同樣可以提供一個共通的雲端執行環境，也可以打破了 IT 廠商壟斷 OS 平臺的局面，提供了商用軟體產品以外的另一個選擇。OpenStack 對雲端平臺的價值，是可以讓使用者造出一套免費的雲端平臺，來實現上雲端的目標，不論是公有雲、私有雲，都可以使用 OpenStack 來建置。OpenStack 創建自己的雲計算服務。有三個主要元件下面。

- Open Stack Compute: 提供和管理虛擬機器的大型網路。
- Open Stack Object Store: 創建使用標準硬體的安全、可靠的存儲海量。

- Open Stack Glance: 目錄和管理大量的伺服器映射庫。

本文所使用的部份是由 OpenStack 的 NOVA API 支援 Docker 的模組，是一個作業系統內核虛擬化，利用此模組達到 IaaS 不同於一般虛擬化的平台管理功能。

2.5 OpenNebula

OpenNebula 是數據中心虛擬化的行業標準的開源產品，提供多種完整功能最，定制的解決方案的基礎上的 Xen，KVM 和 VMware 的虛擬化部署的企業級數據中心和私有雲基礎架構，並提供雲消費者的選擇界面，從開放式雲事實上的標準，像 EC2 API。[16,32]

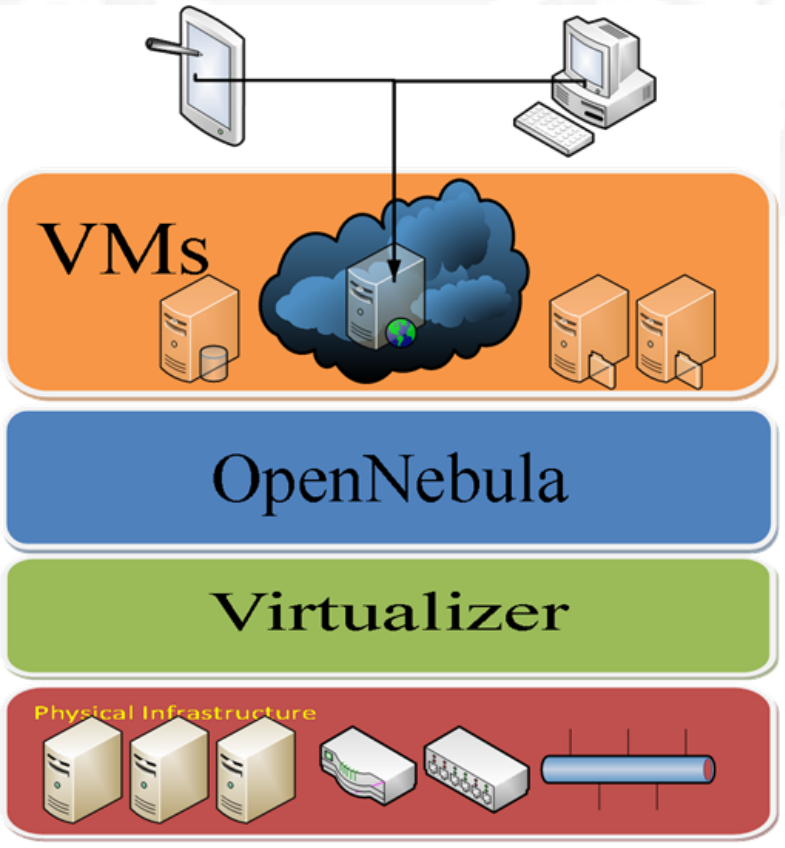

FIGURE 2.5: The diagram of OpenNebula

OpenNebula 是一個虛擬基礎引擎，使動態部署和重新分配的虛擬機中的物理資源池。OpenNebula 提供的功能，在數據中心虛擬化和雲基礎架構的兩個主要層次：數據中心虛擬化管理。許多用戶都使用 OpenNebula 來管理數據中心虛擬化，服務器整合和現有 IT 資產的計算，存儲和網絡集成。在這種部署模式，OpenNebula 直接與虛擬機管理程序集成（如 KVM，Xen 或 VMware 的 ESX），並完全控制虛擬和物理資源，提供先進的功能，容量管理，資源優化，高可用性和業務連續性。其中用戶還可以享受 OpenNebula 的雲管理和配置功能。OpenNebula 包含一個前端和多個後端。前端向用戶提供訪問的接口和管理功能。在後端安裝在 Xen 的服務器，其中的 Xen 虛擬機管理程序的啟動和虛擬機可以支持。前端和後端使用 SSH 之間的通信。該 OpenNebula 為用戶提供了一個單一的接入點上的本地分佈式基礎架構部署虛擬機。OpenNebula 編排存儲，網絡，虛擬化，監控和安全技術的分佈式基礎設施啟用的多層服務的動態放置，結合兩者的數據中心資源和遠程雲資源，根據分配策略

2.6 Docker

Docker 是一個開源專案，是一種 Container 的技術概念，Container 技術則是直接將一個應用程式所需的相關程式碼、函式庫、環境配置檔都打包起來建立集裝箱執行環境，因為與虛擬機（Virtual Machines）的實做功能差不多，但事實上確架構上是有很大的差別的，但其結構與所占的資源都比一般虛擬機少很多，因此又稱為羽量級的作業系統虛擬機，下圖比較了 Container 與傳統虛擬化不同之處。

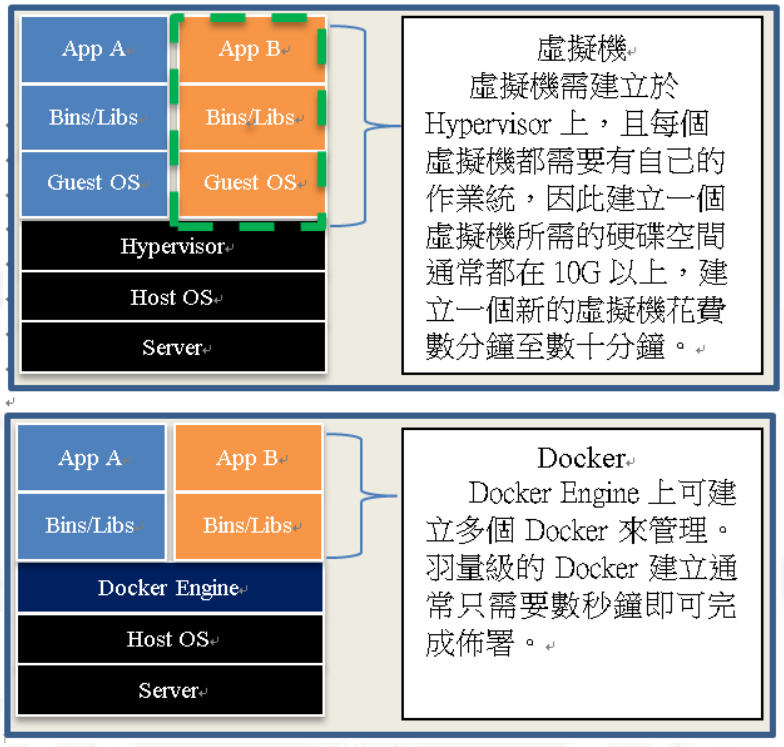

FIGURE 2.6: VM 與 Container 比較

Container 與 Virtual Machines 兩者雖然都屬於虛擬化的技術，其目標都是為了將一個完整服務的應用程式所需的執行環境打包起來，建立一個獨立的環境，方便在不同的實體機中移動，但兩者的運作思維截然不同。簡單來說，常見的傳統虛擬化技術如 vSphere 或 Hyper-V 是以作業系統為中心，而 Container 技術則是一種以應用程式為中心的虛擬化技術。

2.7 Live Migration

透過一套管理模組使資源有效的整合與調節，以更貼近不同需求的實際運用提供的服務，Live Migration 是其中一項重要的技術，可實現在虛擬機在不關機的狀態下遷移到其它實體機上達到負載平衡。

即時遷移是一個虛擬機從一個物理主機到另一處的同時連續地通電的運動。當正確運行時，這個過程没有任何明顯的影響，從使用者來看（圖 1-5）。即時遷移允許管理員採取虛擬機離線進行維護或升級不受該系統使用者停機時間。當資源被虛擬化，虛擬機的額外的管理需要建立，終止，複製或從主機移動到虛擬機主機。虛擬機遷移可以做到離線（客戶在虛擬機已關閉）或上線（正在運行的虛擬機到另一台主機即時遷移）。

即時遷移最為顯著的優點在於，它有助於預防性維護的事實。即將出現故障，服務中斷發生之前可以解決潛在的問題。即時遷移也可用於負載平衡，其中工作的實體機之間共享，以優化實體機可用的 CPU 資源。

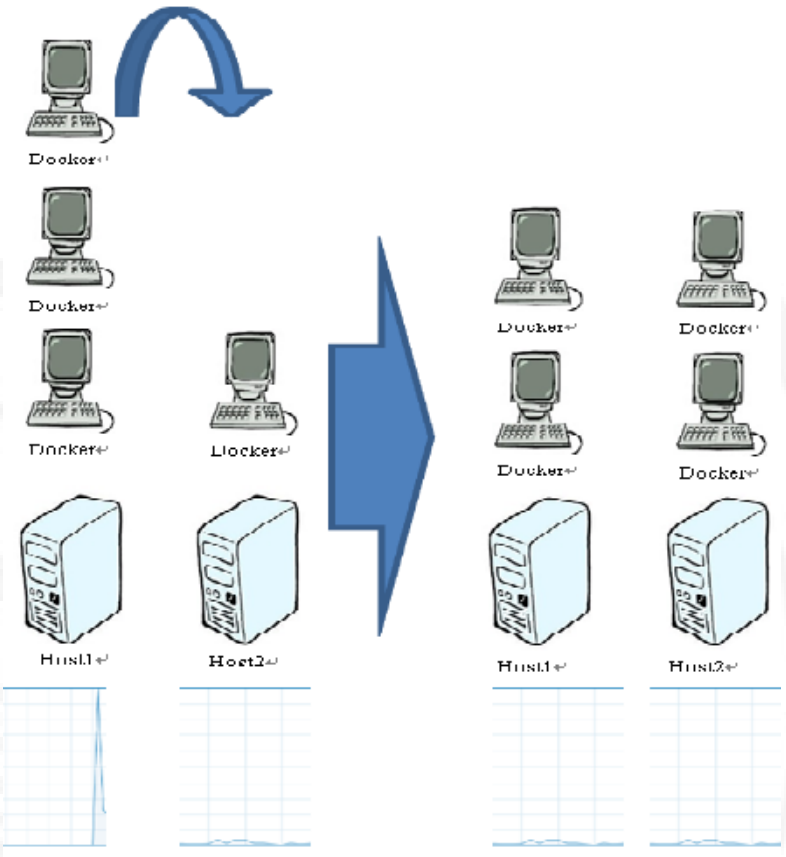

FIGURE 2.7: The concept of Live Migration

Live Migration 可將一個完整執行中的虛擬機從一部實體伺服器移到另一部，卻不會造成停機時間。虛擬機能保存其網路身分識別及連線，以確保順暢的移轉程序。虛擬機的使用中記憶體與精確的執行狀態會透過高速網路迅速傳輸，讓原本在來源 A 主機上執行的虛擬機，可立刻切換到目的地 B 主機上執行。透過 Gigabit 乙太網路，整個移轉過程只需在數秒內執行完畢。

2.8 Live Migration Mechanism of Docker Container

Live Migration 必需做到在不中斷服務的情況下在實體機間將正在執行的處理程序的 Docker Container 作轉移，這是 Container 可以取代 Hypervisors 的一個重要技術，因為在此之前 Container 都著重在容器裡的應用程式與羽量級環境的獨立運行，OpenStack 的 LXD 套件於 2015 年 4 月 29d 發佈的 LXD 0.8.1 版本 (https://linuxcontainers.org/lxd/news/) 已經證實可達到 Live Migration，此技術與以往的 Live Migration 不同之處在於解決 Docker 只能針對其 Container 裡的 APP 做快速佈署但無法做到 Live Migration 的功能，在其羽量級的快速佈署下顯的不足，而 LXD 是建立在 LXC 的標準化之下的產品，因此在 Live Migration 的運作時只需約 0 3 秒即可完成。

2.9 LXC

LXC 的名稱來自 Linux 軟體內核容器（Linux Containers）的縮寫，其使用的是一種作業系統層面的虛擬化技術（Operating system–level virtualization）的技術，其 Linux 內核容器功能的一個使用者空間介面，LXC 將應用軟體系統打包成一個軟體的容器也就是（Container），其中包含應用軟體本身的程式碼以及其所需要的作業系統核心和函式庫，因為只是提供基本的核心與函式庫，所以可以說是簡易版瘦身後的作業系統，再透過統一的命名空間與共用的 API 來配置不同軟體容器的可用實體資源，建構出服務應用程式的獨立 Container 空間的報行環境，使 Linux 的使用者可以容易的建置和管理系統或 Container。

在 Linux 內核中提供了 cgroups 功能來達到資源獨立的區隔化，並且也提供了名稱空間區隔化的功能，這種方式可使得應用程式所見的作業系統環境被區隔成了獨立區間，包括行程樹（tree of process）、網路、使用都識別與掛載的檔案系統，但是 cgroups 並不一定需要啟動任何虛擬機器來執行服務。

LXC 使用了 cgroups 與名稱空間的功能，提供了應用軟體獨立的作業系統環境，LXC 不需要使用 Hypervisor 這個軟體層，因為其使用了先前提到的使用相同的命名空間，因此就不需要再使用一個軟體管理層來管理，軟體容器（Container）本身極為輕量化，提升了建立虛擬機器的速度。軟體 Docker 被用來管理 LXC 的環境。

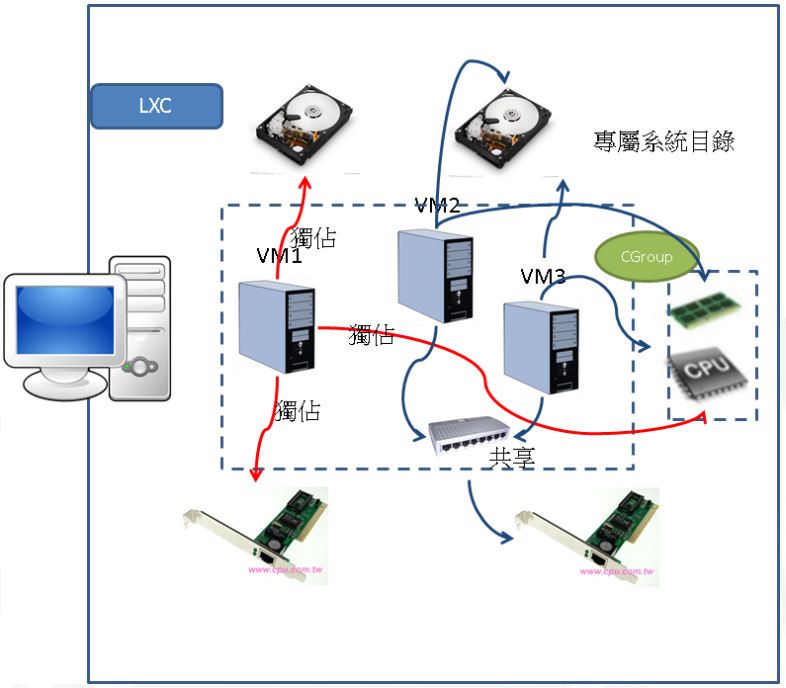

FIGURE 2.8: LXC

2.10 LXD

LXD 是 Container 技術概念而開發的一套程式，核心功能為提供一個後端程式 REST API，應用運行於每台實體主機 (Host) 與用戶端工具，使其能管理這些容器的運行、搬遷與複製等。

LXD 目前建立於 LXC 的之上，LXC 不是一個新的技術，在此之前常被用來做模擬多台 Linux 作業系統，這在需要模擬很多台電腦時，就會很方便。它採用了穩定的 LXC API 來完成所有的容器管理的幕後，增加頂部的 REST API，並提供一個更簡單，更一致的使用。

Chapter 3

系統設計與實作

3.1 系統概觀

以往要架構一個雲端基礎架構（IaaS）為什麼會是困難的一件事，因為我們需要以下幾個步驟

- 第一：使用需要龐大資源的虛擬機軟體如 KVM。
- 第二：建立一個互信的網路登入，類似的組件例如 SSH Setup。
- 第三：網路文件的共享如 NFS。
- 第四：用來管理體機與邏輯虛擬機的 Hypervisor 例如 4. OpenNebula。
- 第五：安裝管理套件如 SunStone。

目前所有雲端平台的架構主流都離不開這五個步驟來完成雲端基礎設施服務（IaaS），這也是目前雲端基礎設施服務（IaaS）當前主流的架設方式。

有別以往的是以往用來做系統壓力測試的技術 LXC 是透過 cgroup 實現資源隔離功能，因為在開發完系統後於系統上線前因為需要提供許多使用者的登入使用，且要模擬出不同使用者所使用的功能與不同時刻使用的人數等等，

LXC 可以在短時間內置做出十台百台千台以上的虛擬機做系統的壓力測試，因為模擬出實際的環境所以每個虛擬機都是獨立乾淨的環境，此獨立乾淨的環境正好符合雲端虛擬機所需要的資料安全性，

本研究提出一套結合 OpenStack 與 Container 等開源碼的集群系統來完成雲端的建制、Live Migration 的功能與效能是本研究的動點，因此 Live Migration 效能評估是本論文著重的部分，故效能評估有三個階段分別是效能模型 (Performance Model)、效能測試 (Performance Test)、效能分析 (Performance Analyzer)。

三種雲的服務模式 Figure 3.1 所示指出本文重點處，系統具有基於網路界面來管理虛擬機，系統會顯示 CPU 目前負載程度，記憶體使用量等來知道資源的利用率。

Figure 3.1顯示了三種雲模型的組成部分，並指出了本文的重點。該系統具有基於網絡的界面進行管理虛擬機。而系統顯示的 CPU 利用率，主機負載，內存利用率和虛擬機信息等。

除了管理虛擬機運行的正常立外，本研究還設計了一個網頁伺服器的服務的部署，這類的服務通常包插一組互相關聯的組件（Web 服務器和資料庫），需要數個虛擬機來完成，因此我們將使用 LXD 來管理部署所有的 Container。

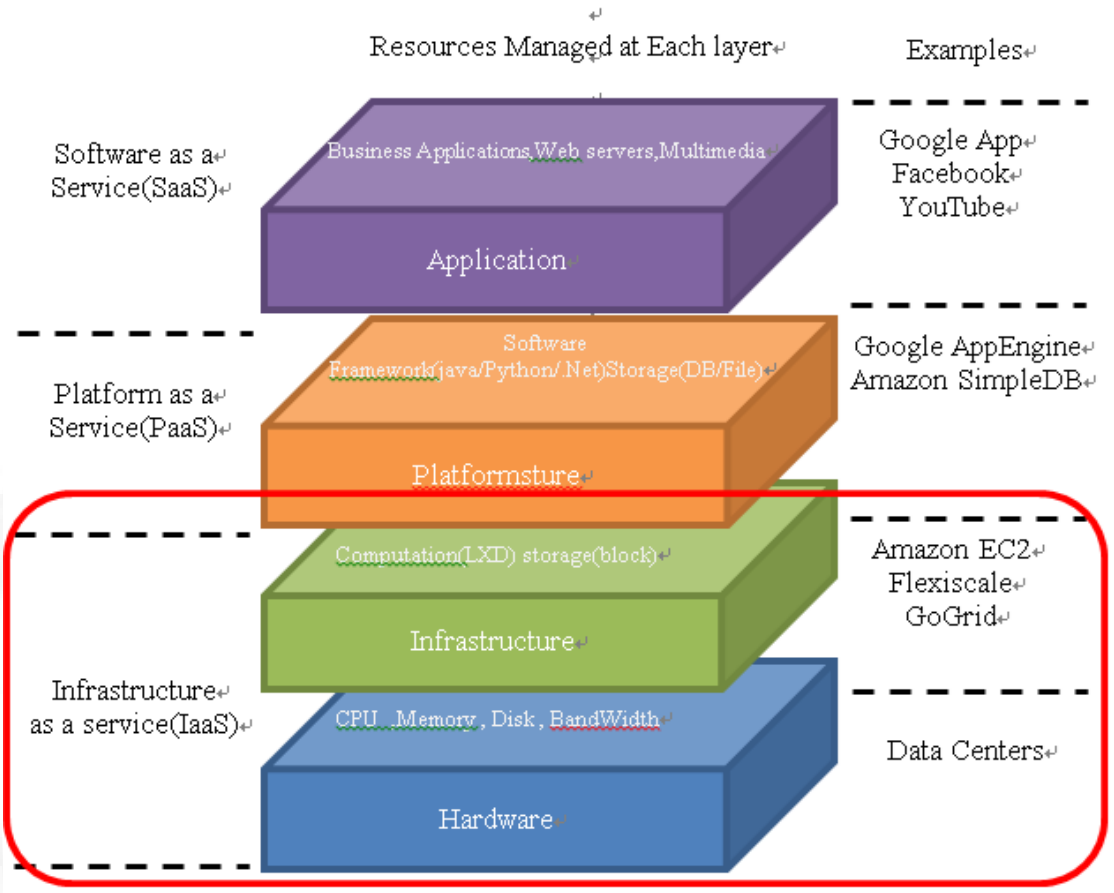

FIGURE 3.1: IaaS 的定義

本研究核心為整合一套完整的雲端服務系統，利用 OpenStack LXD 管理模組、儀表版模組與 Docker Container 等的相互連結來建構 IaaS 的環境，LXD 所提供的是從一個單一的實體資源到另一個實體資源使資源得以延伸的平台，他提供了獨力的儲空間分配各個 Container 所需的資源，我們將利用此系統架構一個可動態佈置的環境 Figure 3.2。

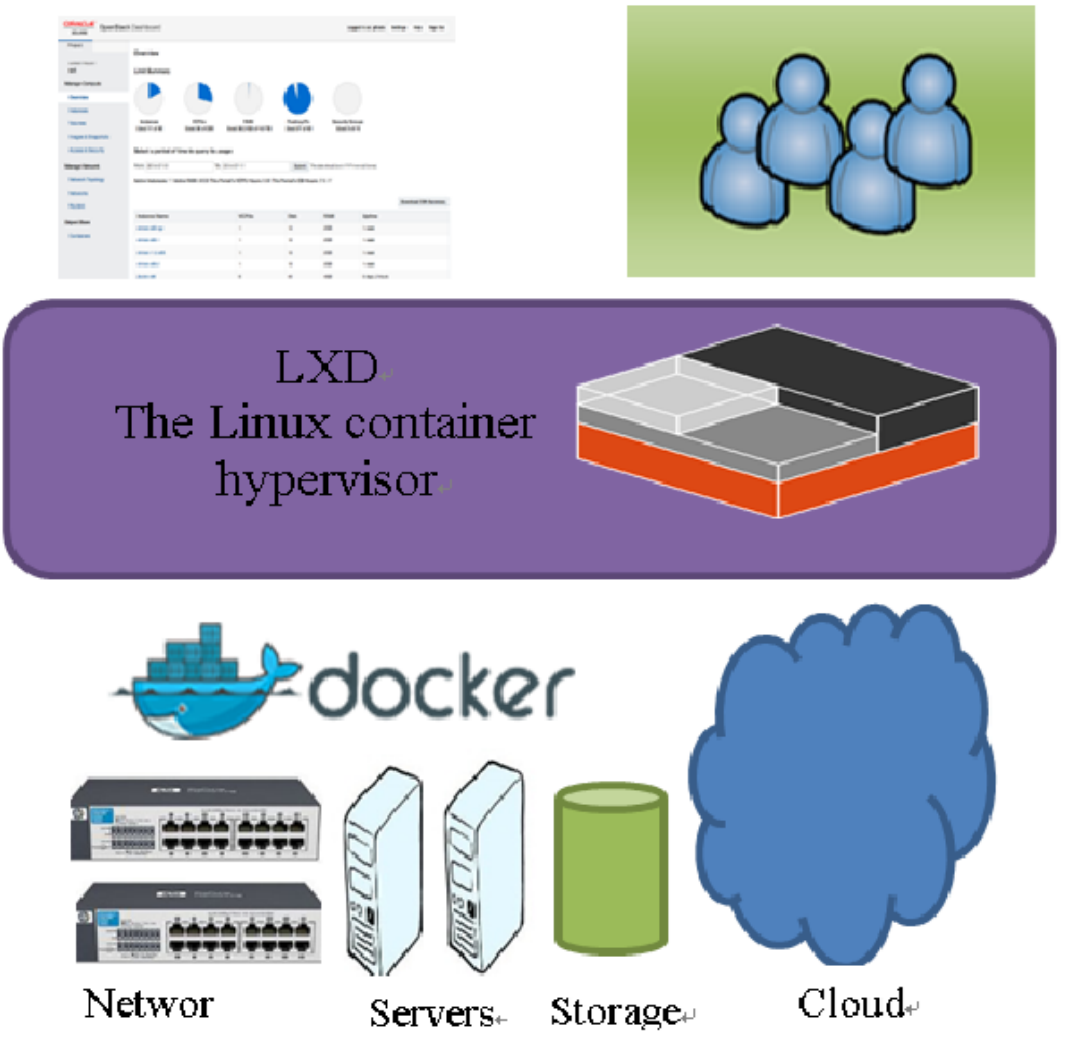

FIGURE 3.2: LXD 系統概念

要完成此研究項目首先要分成幾個階段來一一完成。

- 第一階段：建構環境，建構一個可以建立 Container、移除 Container 與刪除 Container 的環境平台。
- 第二階段：效能測試是將 Container 轉移到另一台實體機上作業，並確認傳送之時間。
- 第三階段：效能分析是將執行中的 Container 動態轉移到另一實體機上，產出評估報告。

- LXC：

- 第二階段：效能測試是將 Container 轉移到另一台實體機上作業，並確認傳送之時間。

- 第三階段：效能分析是將執行中的 Container 即時遷移到另一實體機上，產出評估報告。

3.2 系統架構

本研究為完成一基礎建設即服務（IaaS）設計如 Figure 3.3 為本系統架構，一般的使用者使用個人電腦、平版電腦以及智慧型手機等等使用者介面登入後通過用戶簡單的連結可直接連結到使用的網站，雖然目前還有一些非 linux 作業系統無法產生 Container 但相信不久後的將來會有更多的作業系統與雲端管理系統會相繼支援 LXD。

管理者或使用都可以利用簡單的 WEB 介面登入並管理 Container 如圖 Figure 3.4 Figure 3.5 Figure 3.6

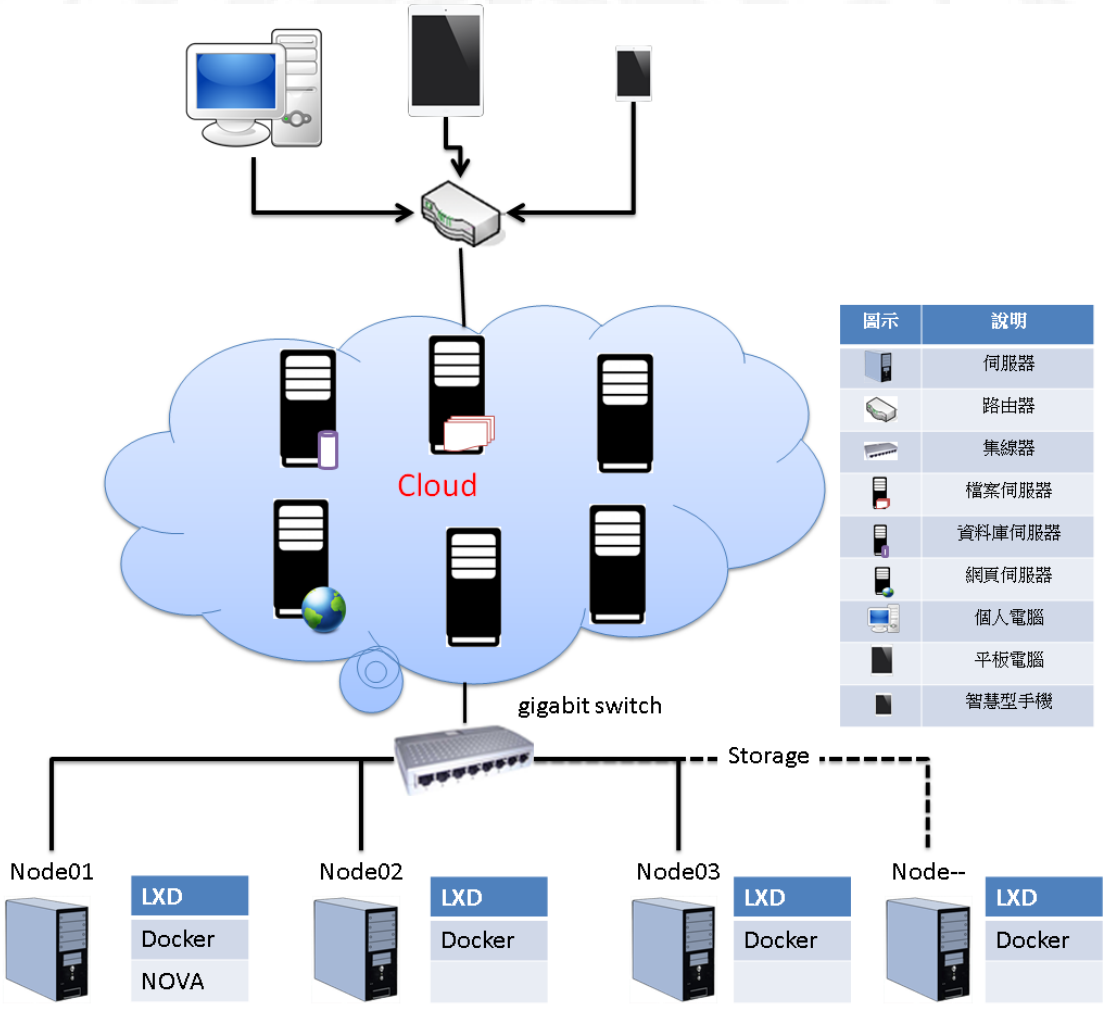

FIGURE 3.3: 系統架構

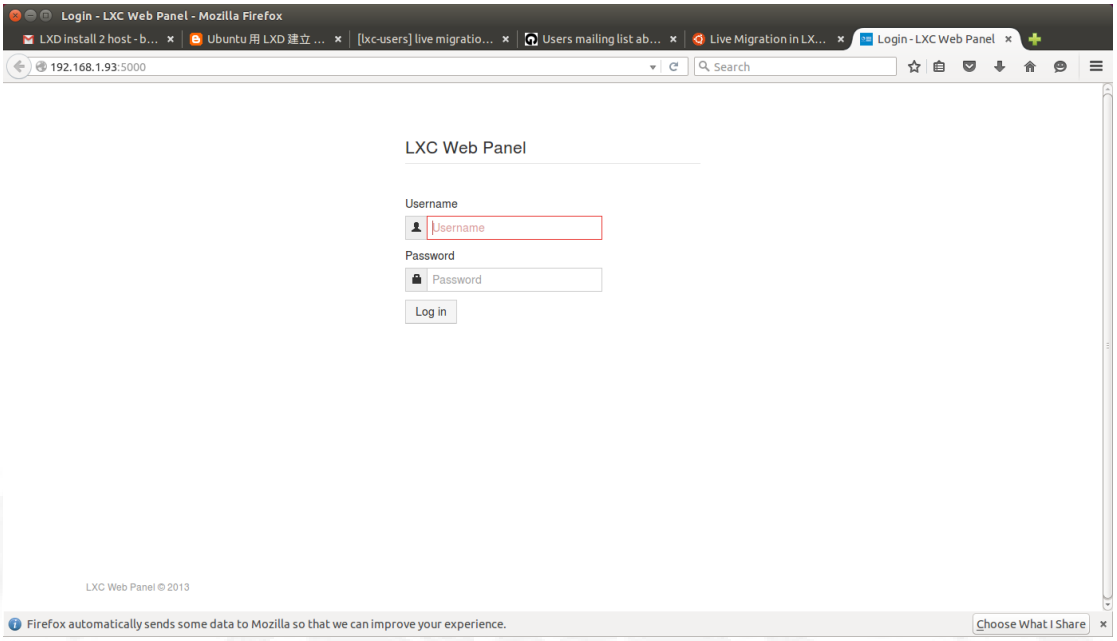

FIGURE 3.4: Container 管理介面登入

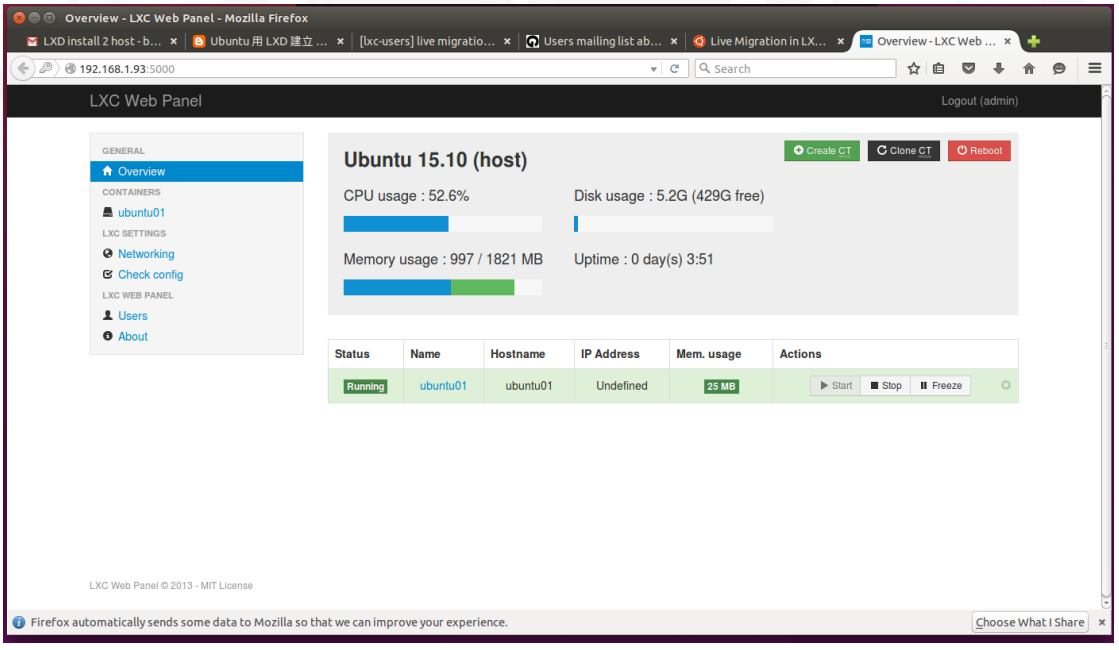

FIGURE 3.5: Container 管理介面總覽

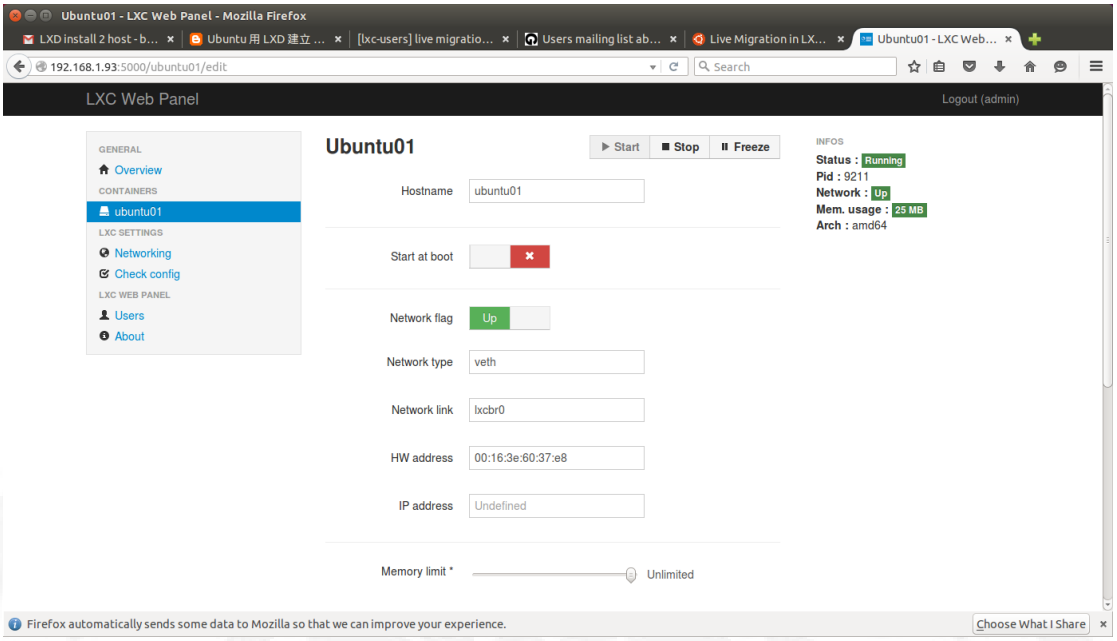

FIGURE 3.6: 系 Container 管理介面設定

3.3 即時遷移 Live Migration

即時遷移的目的是為了保持虛擬機持續的運行達到服務的不中斷來進行虛擬機在不同的實體伺服器作搬移的工作。在虛擬機上運行的服務始終能對正在使用服務的用戶使用者能夠不間斷的提供服務，在搬運的過程中虛擬機會在目的實體機中做類似複製的動作來達到記體上的同步，在完成前服務仍是由原實體位置的虛擬機做服務，在遷移完成時達到同步的狀況之下在網路上會做切斷舊服務與連上新服務的動作，這個動作會中斷服務但時間相當的短，幾乎不會被使用都感受到的時間內完成，在 Figure 3.7 中我們將可以清楚的分辨出每一步即時遷移在運作的過程。

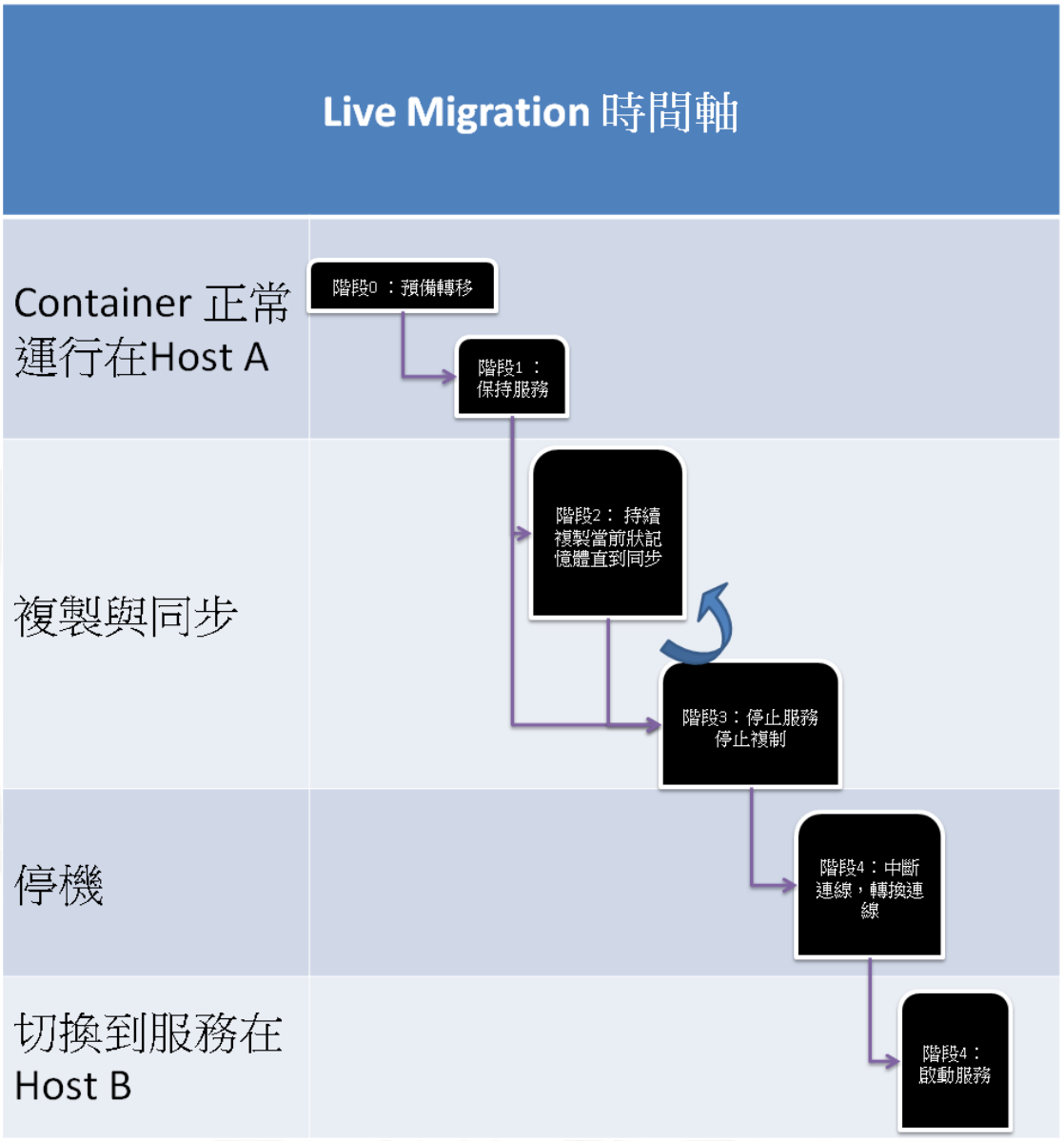

FIGURE 3.7: Live Migration 時間軸

3.4 應用（XAMPP）

XAMPP 是易於安裝的 Apache 發行版本，這是一套結合網頁（PHP）、資料庫（MySQL）與通用的腳本語言（Perl），可以快速建立一套網頁服務系統，本研究利用此架站工具實際運用於強視科技股份有限公司網站架設上

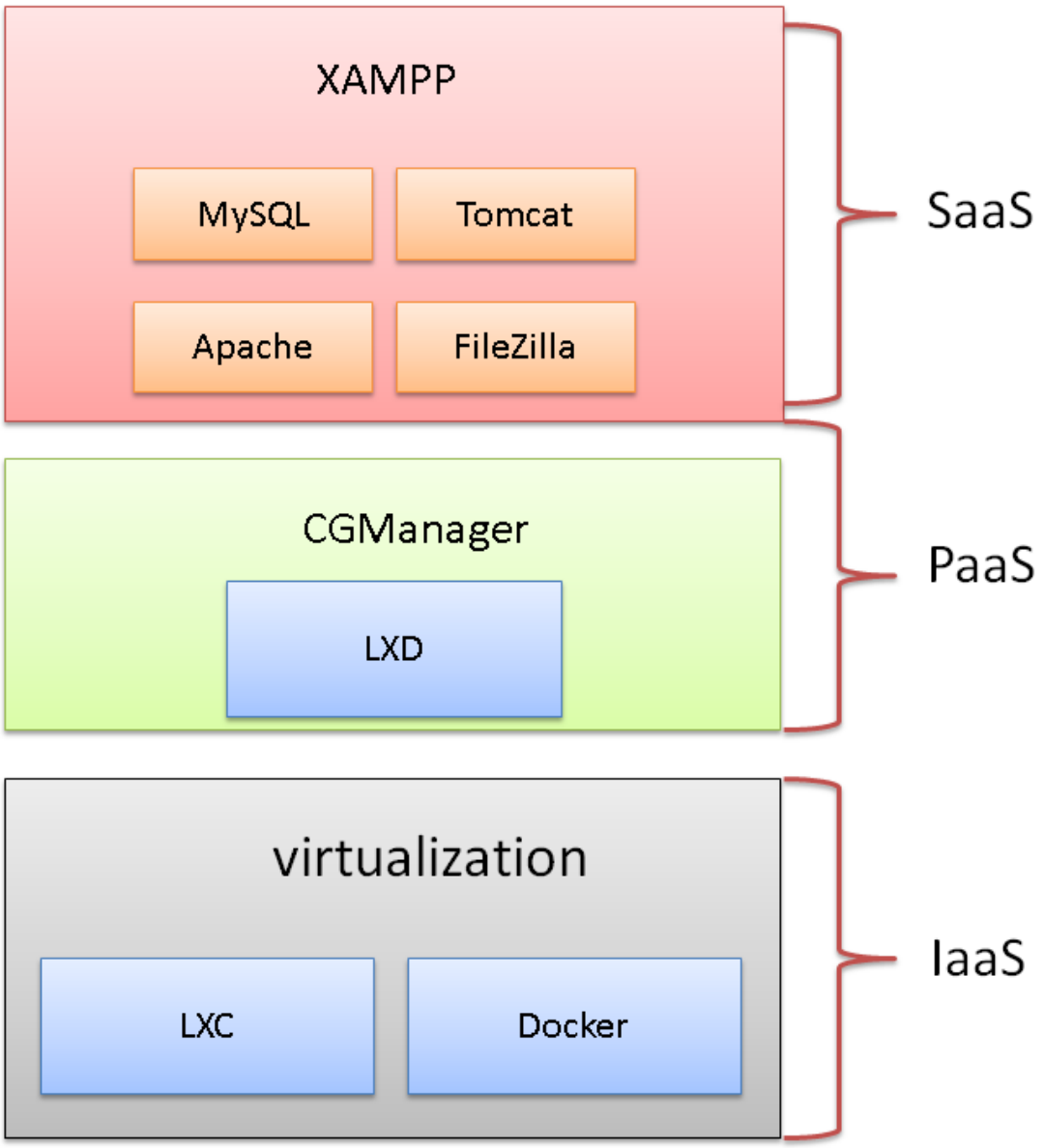

FIGURE 3.8: XAMPP 組成架構

在使用 XAMPP 加設網站上可以看到 Figure 3.8詳細描述各個組件，用此來畫分整個系統在雲端三個服務的模式。

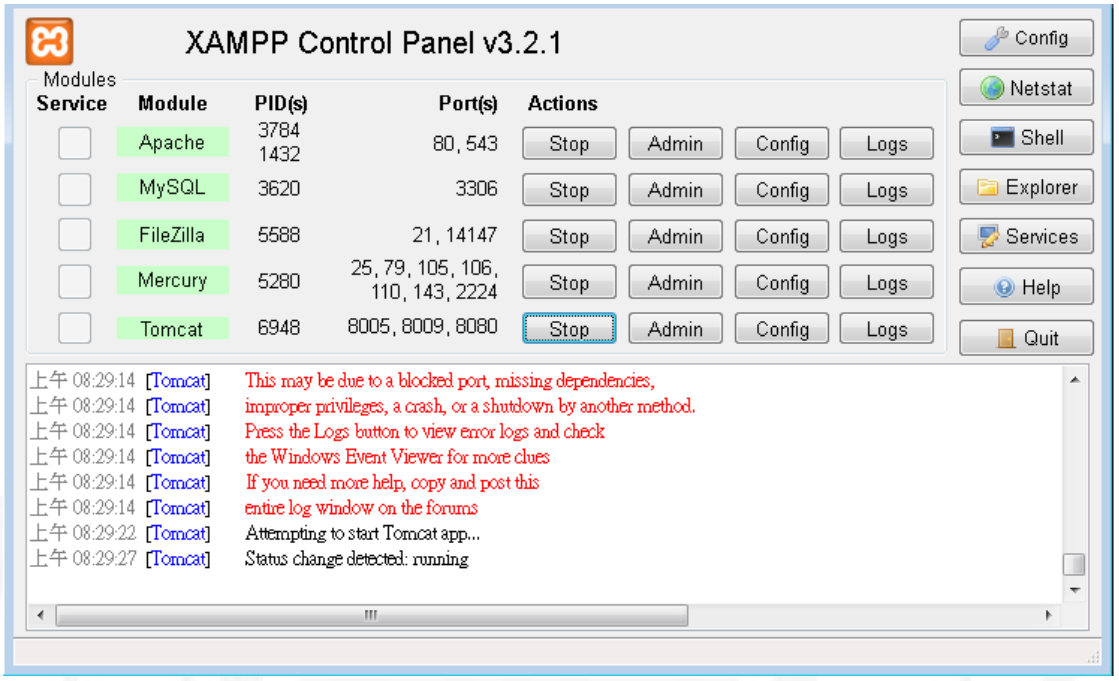

FIGURE 3.9: XAMPP 套件

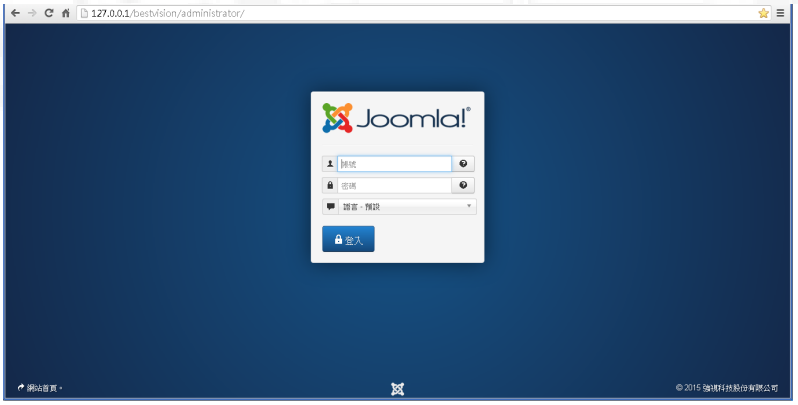

FIGURE 3.10: XAMPP 管理登入

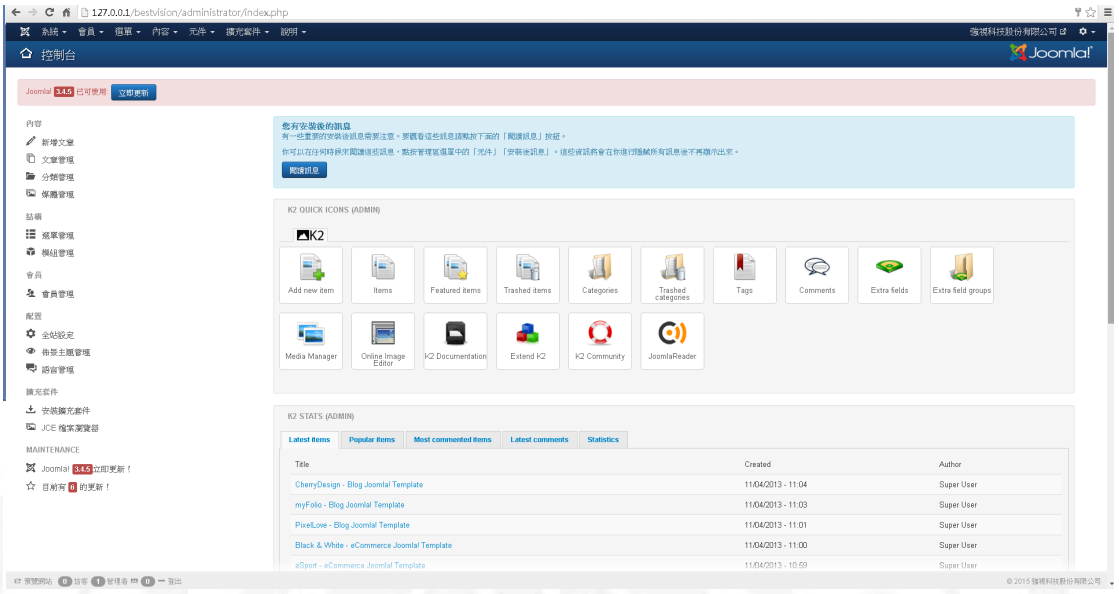

FIGURE 3.11: XAMPP 管理介面

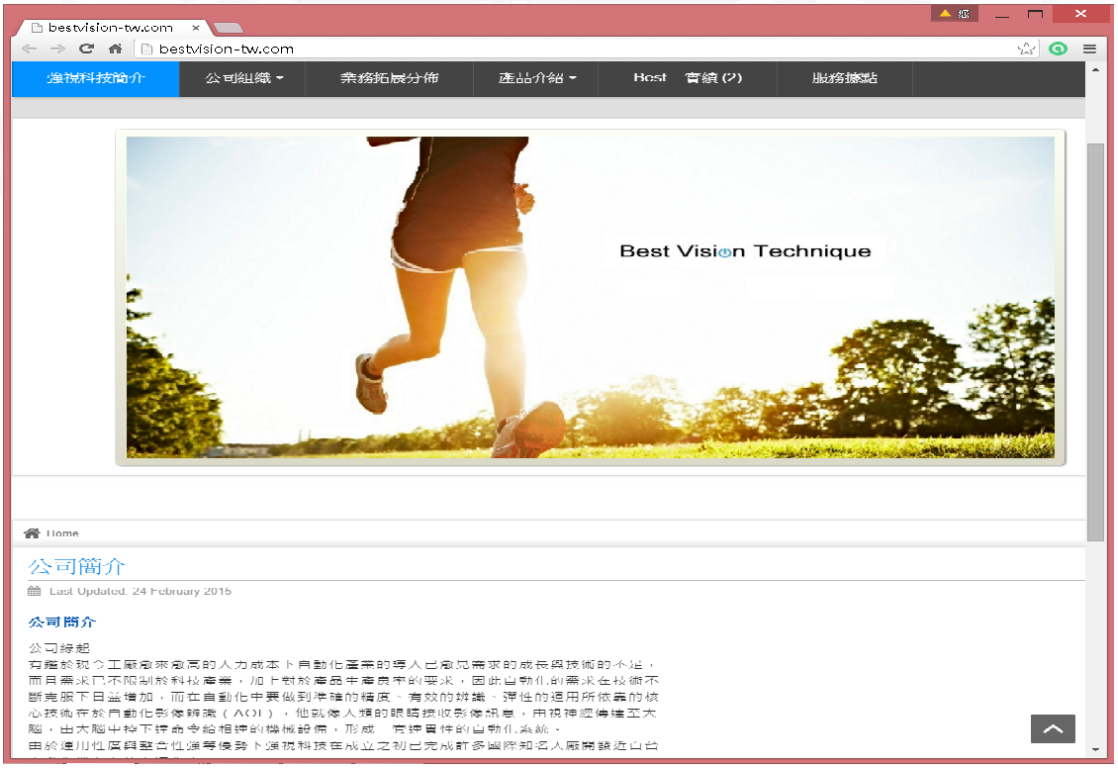

FIGURE 3.12: XAMPP 架設強視科技網站

- SaaS：在軟體即服務層面我們提供用號或管理員可以管理 Web 介面的新增、刪除、修改與移動 Figure 3.8，另外當然也包含了部署 Apache 和 MySQL 等的服務。

- PaaS：在平台即服務上具是套過套件 CGManager，主要功能是在統一命名空間使 LXD 在不同硬體上實作虛擬機的平台。
- IaaS：在基礎建設即服務上我們套過 LXC 與 Docker 的 Container 建立，使在一個實體機上可以生成許多羽量及的虛擬機容器。

借此真正落實在我們的雲平台三大服務模式（SaaS 的，PaaS 和 IaaS 的）的組合。

Chapter 4

實驗環境與結果

4.1 實驗實作

本實驗研究在使用 LXD 與 Container 作為雲端基礎架構即服務 (IaaS) 平台，兩種平台上均使用 Ubuntu 14.10 Desktop 為作業系統，再安裝 XAMPP 各套件系統。

本研究側重於即時遷移功能和 Container 虛擬機的效率測試，包括實體機和虛擬機之間的計算性能，在一般 IaaS 的服務提供者在平常維護硬體上除了追求服務速度上之外，為了服務不中斷與靈活與充分運用硬體資都會使用 Live Migration，事實上 Live Migration 也確實被當為理所當然應有的技術之一，因此我們的實驗內容大多都與 Live Migration 的測試為主，其實驗問題列示如下：

- 虛擬機不間斷服務下遷移到不同實體機

  實做 Live Migration 過程中為了了解虛擬機是否保持在持續運作（服務），我們執行連續丟測試封包 ping 給虛擬機。

- 不同大小的虛擬機遷移比較

測試要遷移的 Container 其內存大小影響？我們實驗內存不同大小的 Container 作 Live Migration，分別為 512MB、1GB、2GB、4GB 來佈署虛擬機，分別各自搬移，我們想知道這是否會影響即時遷移的時間。

- 遷移中作檔案下載

  測試要 Live Migration 中的 Container 其在下載檔案時是否會因搬遷時中斷，並比較與一般下載時的速度是否有差異。

4.2 實驗環境

在實驗環境的部份我們例用三台主機來加設 IaaS 的雲端服務，三台電腦的硬體規格是不同，因此我們在實驗時以 Note2 作為標準，再以 Note2 遷移到 Note3 做為遷移的標準，因為 LXD 目前可以支援的作業系統並不多，官方指定測試的版本為 Ubuntu 14.04 ，但目前最新版本為 Ubuntu 15.04，因此我們所有的 Host 皆安裝同樣的作業系統 Ubuntu 15.04 版本 Table 4.1。

Table 4.1: 伺服器規格表

	CPU	Memory	Disk	Network	OS	SoftWare
Host01	Intel(R) Celeron(R) CPU G1610 2.60GHz	4GB	1T	Gigabit	Ubuntu 15.04	LXD 0.25
Host02	Intel(R) Celeron(R) CPU G1610 2.60GHz	4GB	1T	Gigabit	Ubuntu 15.04	LXD 0.25
Host03	Intel(R) Celeron(R) CPU G1610 2.60GHz	4GB	1T	Gigabit	Ubuntu 15.04	LXD 0.25

4.3 結果

實驗部份都以 Live Migration 為核心測試，因為在雲端運算架構上虛擬化獨立作業與服務上如果沒有 Live Migration 的功能那系統將無法靈活調度資源的分配，而要達到 Live Migration 則需要雲端系統一套完整的整合環境從 IaaS 到 PaaS 的運用。

- 在實驗中，我們實作 Live Migration 由 Node2 的虛擬機移到 Node3，過程中我們持續的 Ping Docker Container 的虛擬機服務其 IP 為 192.168.238.134，過程中如 Figure 4.1Figure 4.2Figure 4.3所示，幾乎感受不到停頓的時間，但我們整理資料不難發現在穩定的封包傳輸過程中的跳動停頓，這是因為 CRIU 正在調配網路瞬間切換，這一切的運作以此實驗來說算是十分快速的一個切換過程。

```
[root@Node01 ~]# ping 192.168.238.134
PING 192.168.238.134 (192.168.238.134) 56(84) bytes of data.
64 bytes from 192.168.238.134: icmp_seq=1 ttl=64 time=0.052 ms
64 bytes from 192.168.238.134: icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from 192.168.238.134: icmp_seq=3 ttl=64 time=0.034 ms
64 bytes from 192.168.238.134: icmp_seq=4 ttl=64 time=0.064 ms
64 bytes from 192.168.238.134: icmp_seq=5 ttl=64 time=0.057 ms
64 bytes from 192.168.238.134: icmp_seq=6 ttl=64 time=0.056 ms
64 bytes from 192.168.238.134: icmp_seq=7 ttl=64 time=0.061 ms
64 bytes from 192.168.238.134: icmp_seq=8 ttl=64 time=0.048 ms
64 bytes from 192.168.238.134: icmp_seq=9 ttl=64 time=0.057 ms
```

FIGURE 4.1: Ping of Live Migration

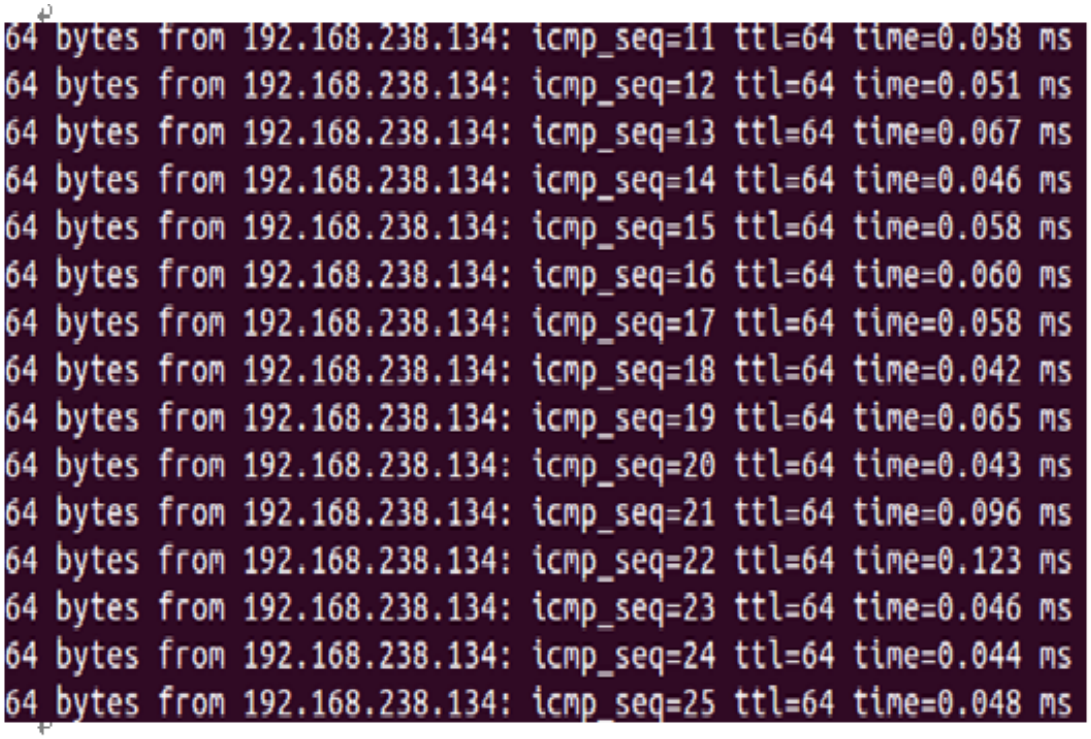

FIGURE 4.2: Ping during Live Migration

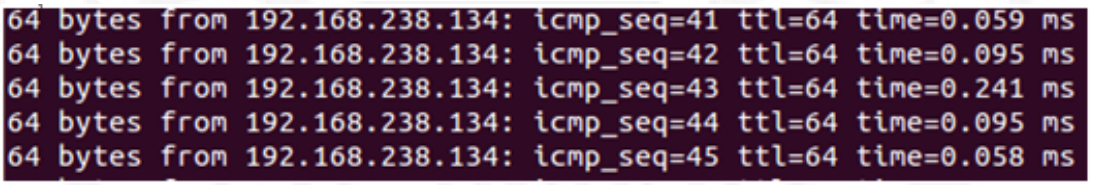

FIGURE 4.3: The ping result after Live Migration finish

- 由於 IaaS 所提供的服務會有許多不同大小的虛擬機 Container，而這些搬移都是透過網路來達成搬移作，不同大小內儲的 Container 理論上應該會有搬遷上時間的差別，但這與其大小是否為等比，我們實驗內存不同大小的 Docker Container 作 Live Migration，分別為 512MB、1GB、2GB、4GB 來佈署虛擬機，分別各自搬移，此次實驗為 Host3 從未有載入與搬移過這些 Docker Container 的情況下作搬移的動作，實驗結果我們發現其搬移時差異甚小，甚至誤差在人為下指令的誤差範圍時間內如 Figure 4.4，因此在內儲較大的 Container 其效率較好。

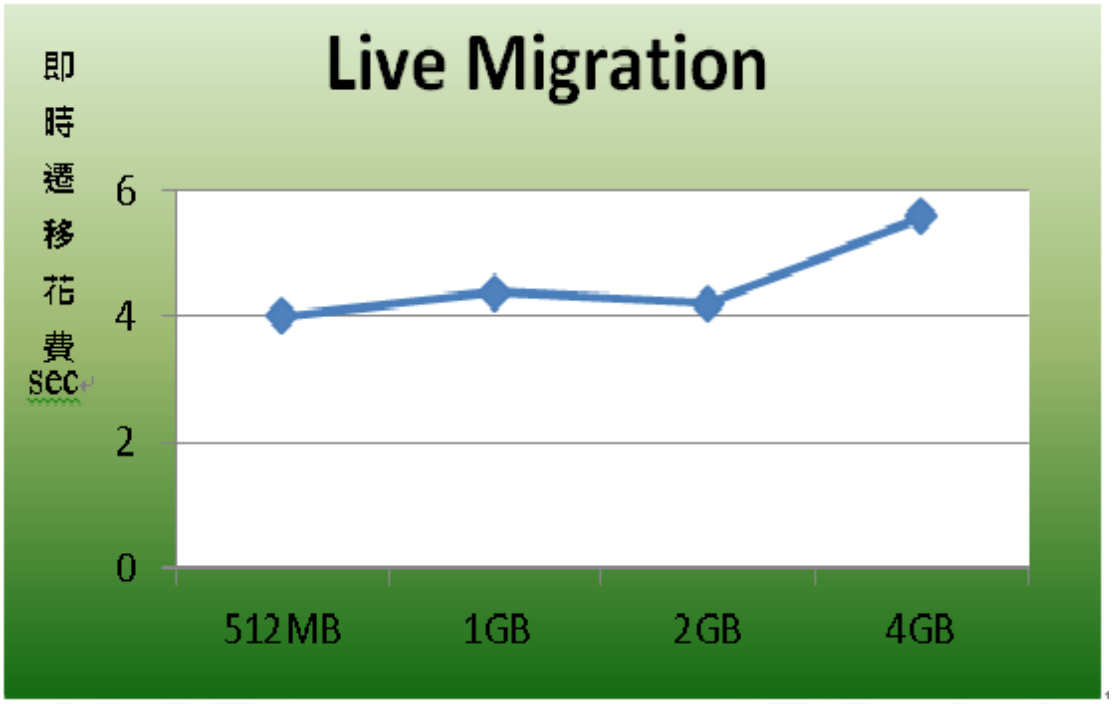

FIGURE 4.4: 不同容量 Live Migration 的變化

- 因搬移次數之間的頻繁會影響平日維護的工作量與效率，實驗重覆搬移其 Live Migration 時間：實驗 512MB、1GB、2GB、4G 的 Docker Container 在 Host2 與 Host3 之間重覆搬移時間之變化，明顯快了許多如 Figure 4.5

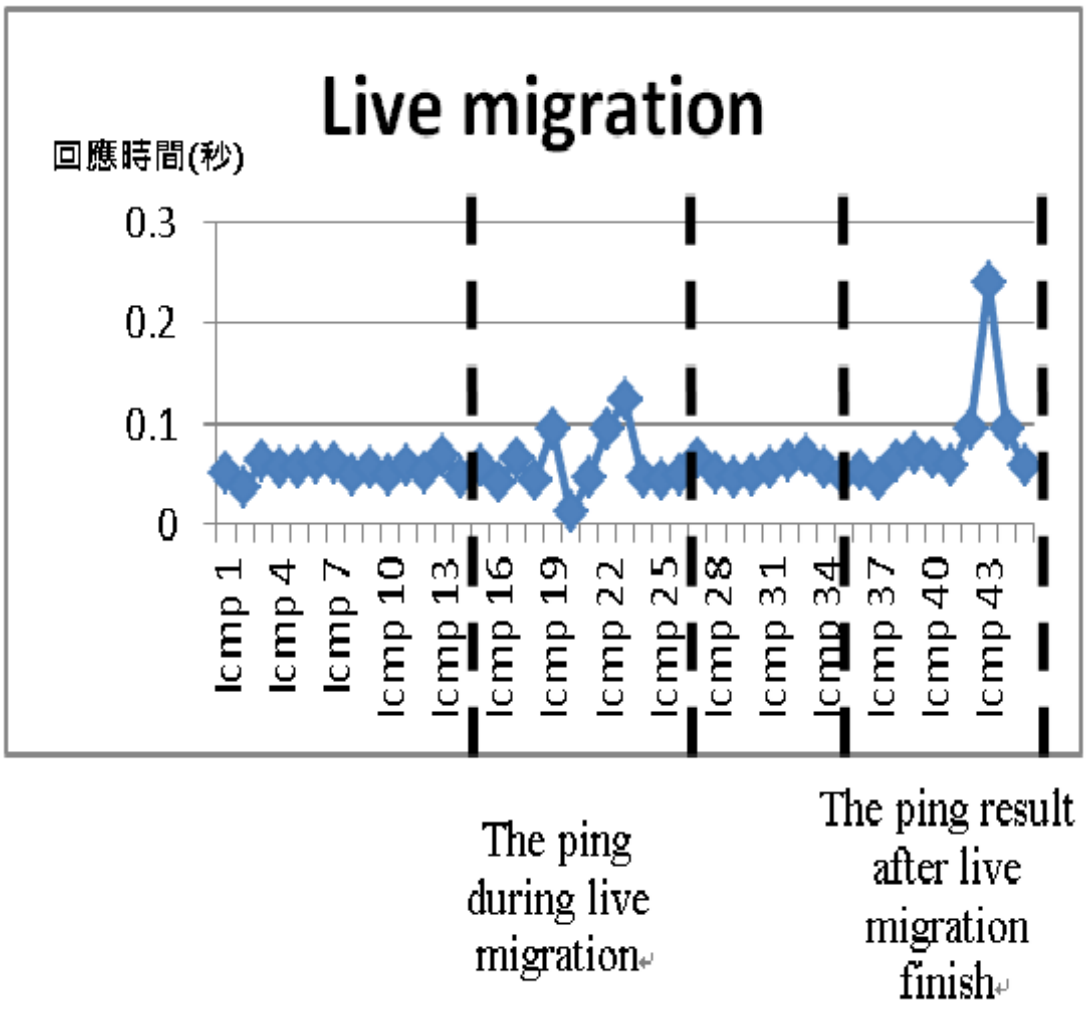

FIGURE 4.5: Live Migration 時間變化

- 實驗在 Container 中下載網路上的檔案的同時作 Live Migration 可以看到在實驗中是被正常下載的 Figure 4.6，我們再與未做 Live Migration 下載同樣的檔案 Figure 4.7做速度上的比較 Figure 4.8，結果顯示在下載檔案中 Live Migration 會使下載檔案的時間明顯增多 Figure 4.9。

```
rm: cannot remove 'ubuntu-14.04.3-desktop-amd64.iso': No such file or directory
root@first:~# wget http://ftp.ubuntu-tw.org/mirror/ubuntu-releases/14.04.3/ubunt
u-14.04.3-desktop-amd64.iso
--2015-12-31 03:11:06--  http://ftp.ubuntu-tw.org/mirror/ubuntu-releases/14.04.3
/ubuntu-14.04.3-desktop-amd64.iso
Resolving ftp.ubuntu-tw.org (ftp.ubuntu-tw.org)... 163.22.3.70, 2001:e10:6840:3:
:70
Connecting to ftp.ubuntu-tw.org (ftp.ubuntu-tw.org)|163.22.3.70|:80... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 1054867456 (1006M) [application/x-iso9660-image]
Saving to: 'ubuntu-14.04.3-desktop-amd64.iso'

100%[====================================>] 1,054,867,456 1.84MB/s   in 14m 26s

2015-12-31 03:25:36 (1.16 MB/s) - 'ubuntu-14.04.3-desktop-amd64.iso' saved [1054
867456/1054867456]

root@first:~#
```

FIGURE 4.6: 下載檔案時做 Live Migration 正常下載了檔案

```
brady@ubuntu:~$
brady@ubuntu:~$ ls
current  Documents  examples.desktop  Pictures  Templates
Desktop  Downloads  Music             Public    Videos
brady@ubuntu:~$ sudo lxc exec first -- /bin/bash
[sudo] password for brady:
root@first:~# wget http://ftp.ubuntu-tw.org/mirror/ubuntu-releases/14.04
u-14.04.3-desktop-amd64.iso
--2015-12-31 07:48:23--  http://ftp.ubuntu-tw.org/mirror/ubuntu-releases
/ubuntu-14.04.3-desktop-amd64.iso
Resolving ftp.ubuntu-tw.org (ftp.ubuntu-tw.org)... 163.22.3.70, 2001:e10
:70
Connecting to ftp.ubuntu-tw.org (ftp.ubuntu-tw.org)|163.22.3.70|:80... c
.
HTTP request sent, awaiting response... 200 OK
Length: 1054867456 (1006M) [application/x-iso9660-image]
Saving to: 'ubuntu-14.04.3-desktop-amd64.iso.1'

 2% [                                      ] 24,280,968  2.36MB/s  eta
```

FIGURE 4.7: 直接進入 Container 中下載檔案不做搬移的動作

```
100%[====================================>] 1,054,867,456 1.92MB/s   in 11m 21s

2015-12-31 03:03:53 (1.48 MB/s) - 'ubuntu-14.04.3-desktop-amd64.iso' saved [1054
867456/1054867456]

root@first:~# ls
ubuntu-14.04.3-desktop-amd64.iso
```

FIGURE 4.8: 載入檔案完成的時間

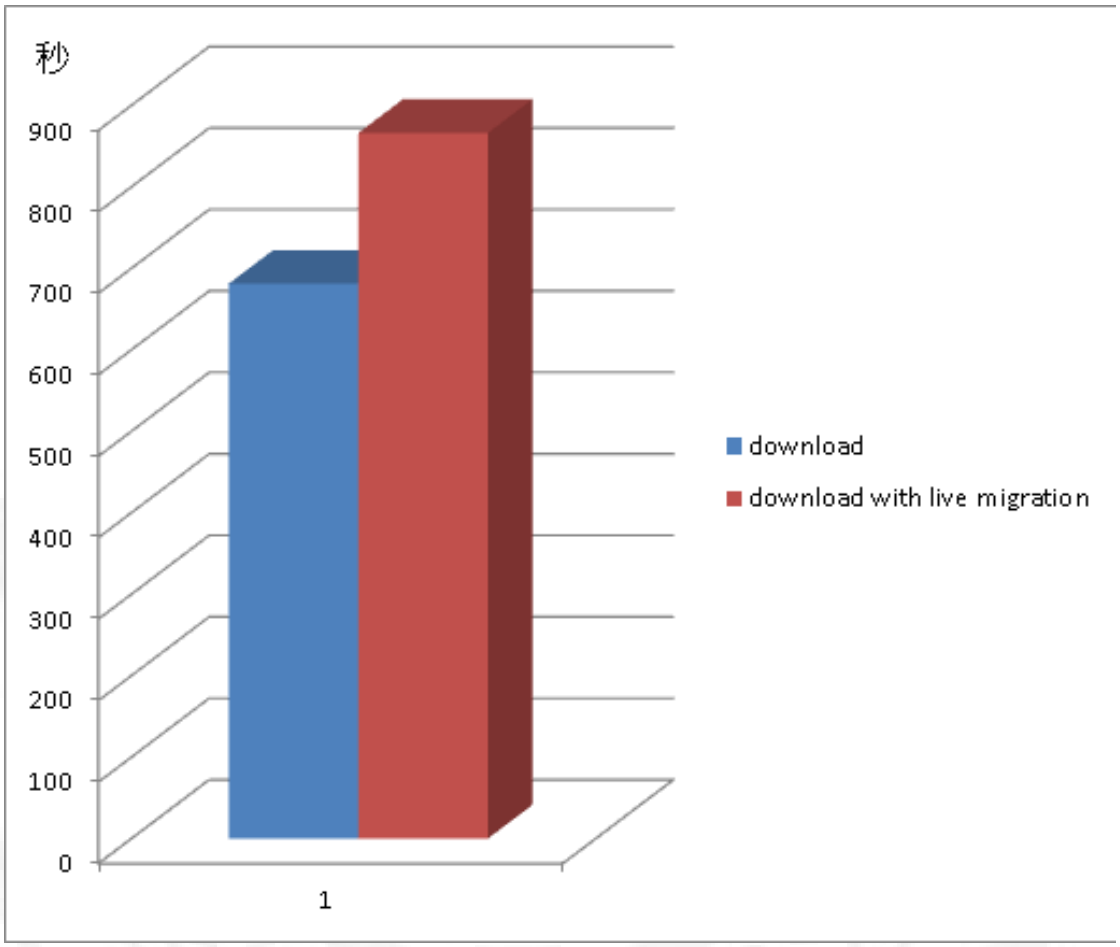

FIGURE 4.9: Download and Download with Live Migration

Container 管理器是 LXD，而 LXD 是建立在 LXC 的基礎上的管理器，被視為 hypervisor 同等的角色，LXD 也支援 Docker Container 的掛載運行，因此我們實驗上所使用的管理命令與工具皆是透過 LXD 來完成，這個實驗可以實現我們所想知道即時遷移在本研究中所得到的成果與數據的比較。

- 在實驗 1 中

  我們可以感受到“Live Migration”在實際操作的過程中是不間斷服務的，在 Figure 4.2中持續 Ping 的過程中維有在轉移連線的瞬間有明顯的波峰，這是發生在實體機與實體機間交換的過程，所以“Live Migration”運用中是可以減少停機時間至毫秒等級，這毫秒等級的轉換時間可以讓使用者就算是在密集的使用下也幾忽感不到其服務的機器已作了轉移的動作。

- 在實驗 2 中

我們嘗試作各個內存大小不同的 Container 來測試 “Live Migration” 對於遷移時間的影響實驗，實驗中發現不同內存大小的 Contaner 在遷移的時間上與其內存大小幾忽沒有關聯，那是因為其內存的空間在實際上並非存在單一實體機上來需要搬移，實際遷移的是”dirty pages”。

- 在實驗 3 中

  我們下載了 ubuntu14.04.3-desktop-amd64.iso 的檔案，實驗中證實了在 Live Migration 中下載檔案是可以正常運作的，但因透過實驗也發現當沒有做 Live Migration 時下載速度明顯的快過有做 Live Migration 的下載，原因是因為 Live Migration 是運作是透過 CRIU 運作透過 socket 網路傳輸，因此速度上才會有明顯的差異，在運作的過程中我們也發現到當 Live Migration 在執行時下載時的速度由原本的 2.38MB/s Figure 4.10明顯的下降到了 0.557MB/s Figure 4.11。

```
u-14.04.3-desktop-amd64.iso
--2015-12-31 04:01:38--  http://ftp.ubuntu-tw.org/mirror/ubuntu-releases/14.04.3
/ubuntu-14.04.3-desktop-amd64.iso
Resolving ftp.ubuntu-tw.org (ftp.ubuntu-tw.org)... 163.22.3.70, 2001:e10:6840:3:
:70
Connecting to ftp.ubuntu-tw.org (ftp.ubuntu-tw.org)|163.22.3.70|:80... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 1054867456 (1006M) [application/x-iso9660-image]
Saving to: 'ubuntu-14.04.3-desktop-amd64.iso'

 5% [=>                                      ] 59,943,482  2.38MB/s  eta 6m 56s
```

FIGURE 4.10: 未執行 Live Migration 時下載速度

```
u-14.04.3-desktop-amd64.iso
--2015-12-31 04:01:38--  http://ftp.ubuntu-tw.org/mirror/ubuntu-releases/14.04.3
/ubuntu-14.04.3-desktop-amd64.iso
Resolving ftp.ubuntu-tw.org (ftp.ubuntu-tw.org)... 163.22.3.70, 2001:e10:6840:3:
:70
Connecting to ftp.ubuntu-tw.org (ftp.ubuntu-tw.org)|163.22.3.70|:80... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 1054867456 (1006M) [application/x-iso9660-image]
Saving to: 'ubuntu-14.04.3-desktop-amd64.iso'

18% [======>                                 ] 190,424,600  557KB/s  eta 7m 31s
```

FIGURE 4.11: 執行 Live Migration 時下載速度

Chapter 5

結論與未來方向

5.1 結論

在本文中，我們提供了一個成功的開源免費的私有雲解決方案，而我們研究探討的即時遷移“Live Migration”協助了在日常運作上的實際運用。

另一方面在我們的架構下佈署 XAMPP 成功的架設了企業網頁，這代表了三種雲端服務模式的實現，該系統可以視為一個成功的私有雲服務系統。

在現今雲端運算的應用最為廣泛的是提供軟體的服務，而能提供雲端基礎建設服務 IaaS 的確不多，原因就在於能夠提供所需要加設的技術與投資門檻過高，目前開放式軟體能提供的解決方案不多，且大多數都以虛擬機 VM 的方式虛擬化，此方法所占硬體資源與本研究所提出的 Container 容器所占的硬體資源相差甚多為最特別的特色。

5.2 未來方向

雖然我們成功的架設時遷移，也從驗中了解遷移的過程中不會造成服務的中斷，但實際運用在工作中還是有幾個地方需要詳加探討與追蹤。

我們探討的是與一般虛擬化不同的雲端 IaaS 架構方式，羽量級的虛擬機空間大量結省了硬體資源的佔用，雖然如此但所支援的作業系統有受限是目前最大的困難點，如果一個虛擬容器其作業系統 OS 無法全面性的提供想必會是提供服務上的一大缺失這也是日後需努力的一個方向。

Container 的作業系統除了無法全面性的提供之外，另一點也需未來期待能夠有所突破的是更新的速度，目前作業系統要能跟著改版有賴於版本的釋出，其時間一般無法與新版 OS 同期推出，因此受制於作業系統商開發的速度。

因此未來方向我們大概整理出以下幾點在未來研究上可以在雲端運算服務上提供更完整的解決方案如下：

- 大量資料交換下是否會造成數據的遺失。

  之所以使用雲端服務除了期提供的便利性之外，更重要的一點在於提供正確無損的資料，在 Container 雖然是一個經過封裝打包的容器，但在 Live migration 中其搬遷失敗或多種未知的情況下所造成的錯誤是否能保證大量資料交換的過程驗證資料的完整性。

- 虛擬機的帳戶管理缺乏擴展與管理的方便性。

  目前 LXD 已釋出的版本仍處於測試階斷，尚未提供一個完整正式版本，因此帳戶管理上仍有許多不便與限制，這在實際運用在業界來說仍是一個不確定性。

- 虛擬機作業系統的限制性。

  如先前所說的，Container 是經過作業系統羽量化過的容器空間，如作業系統程式碼是封閉的將無法提供該作業系統的服務，但本研究的同時已有如微軟等作業系統提出將支援 Docker Container。

- 未來在不同雲之間交換的適應性。

  本研究所提出的虛擬化是不同於傳統虛擬機 VM 的方式運作，而雲端服務運用上佈署私有雲與混合雲都有資訊交換與相容的問題，這不同的虛擬機是否有相容適應性的問題將也是未來需探討的方向。

Container 目前已證實在雲端 IaaS 實作上的可行性，但其適應性與其作業系統支援上將是由怎麼樣的一個方式去制訂使其更完整我想是未來研究可以多加著墨的地方。

參考文獻

[1] Rongfeng Lai Liang Yang Zhenlin Wang Yingwei Luo Binbin Zhang, Xiaolin Wang and Xiaoming Li. Evaluating and optimizing i/o virtualization in kernel-based virtual machine(kvm). In *NPC'10 Proceedings of the 2010 IFIP international conference on Network and parallel computing*, pages 220–231, 2010.

[2] Yvonne Coady Chris Matthews. Virtualized recomposition: Cloudy or clear? In *ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 38–44, 2009.

[3] Keng-Yi Chou Chien-Hsiang Tseng Chao-Tung Yang and Shyh-Chang Tsaur. Design and implementation of a virtualized cluster computing environment on xen. In *presented at the The second International Conference on High Performance Computing and Applications*, 2009.

[4] C. A. Waldspurger. Memory resource management in vmware esx server. In *SIGOPS Oper. Rev., vol. 36, no. SI*, pages 181–194, 2002.

[5] F. Bellard. Qemu, a fast and portable dynamic translator. In *in Proceedings of the USENIX 2005 Annual Technical Conference, FREENIX Track*, pages 41–41, 2005.

[6] Peter Kacsuk Zsolt Nemetha Gabor Kecskemeti, Gabor Terstyanszky. An approach for virtual appliance distribution for service deployment. In *Future Generation Computer Systems, Volume 27 Issue 3*, pages 280–289, 2011.

[7] H. Raj and K. Schwan. High performance and scalable i/o virtualization via self-virtualized devices. In *in the proceedings of HPDC*, pages 179–188, 2007.

[8] Jean-Marc Menaud Hien Nguyen Van, Fr´ed´eric Dang Tran. Autonomic virtual resource management for service hosting platforms. In *ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 1–8, 2009.

[9] Hitoshi Oi and Fumio Nakajima. Performance analysis of large receive offload in a xen virtualized system. In *in Proceedings of 2009 International Conference on Computer Engineering and Technology (ICCET 2009)*, page 475–480, 2009.

[10] J. E. Smith and R. Nair. The architecture of virtual machines. In *Computer, vol. 38, no.5*, pages 32–38, 2005.

[11] Aravind Menon Scott Rixner Alan L. Cox J. S. Paul Willmann, David Carr and Willy Zwaenepoel. Concurrent direct network access for virtual machine monitors. In *The second International Conference on High Performance Computing and Applications*, pages 306–317, 2007.

[12] Kacsuk. P. Kertesz. A. Grid interoperability solutions in grid resource management. In *Systems Journal, IEEE, Volume 3, Issue:1*, pages 131–141, 2009.

[13] K. Adams and O. Agesen. A comparison of software and hardware techniques for x86 virtualization. In *in ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, page 2–13, 2006.

[14] Victor Gil Fermín Galán Javier Fontán Rubén S. Montero Ignacio M. Llorente Luis Rodero-Merino, Luis M. Vaquero. From infrastructure delivery to service management in clouds. In *Future Generation Computer Systems, Volume 26 Issue 8*, pages 1226–1240, 2010.

[15] Llorente Ignacio M. Montero Ruben S. Milojičić, Dejan. Opennebula: A cloud management tool. In *Internet Computing, IEEE, volume 15, issue 2*, pages 11–14, 2011.

[16] Umesh Deshpande Michael R. Hines and Kartik Gopalan. Post-copy live migration of virtual machines. In *Computer Science, Binghamton University (SUNY)*.

[17] Judith Kelner Djamel Sadok Patrícia Takako Endo, Glauco Estácio Gonçalves. A survey on open-source cloud computing solutions. In *VIII Workshop em Clouds, Grids e Aplicações*, pages 3–16, 2011.

[18] Qumranet. White paper: Kvm kernel-based virtualization driver. In *in SOSP ' 03: Proceedings of the nineteenth ACM symposium on Operating systems principles. New York, NY, USA: ACM Press*, page 164–177, 2003.

[19] K. Fraser S. Hand T. Harris A. Ho R. Neugebauer I. Pratt P. Barham, B. Dragovic and A. Warfield. Xen and the art of virtualization. In *in SOSP ' 03: Proceedings of the nineteenth ACM symposium on Operating systems principles. New York, NY, USA: ACM Press*, page 164–177, 2003.

[20] Qumranet. White paper: Kvm kernel-based virtualization driver. 2006.

[21] Ian Foster R. S. M. Borja Sotomayor, Ignacio M. Llorente. Virtual infrastructure management in private and hybrid clouds. In *IEEE Internet Computing, vol. 13*, pages 16–23, 2009.

[22] M. E. Fiuczynski A. Bavier S. Soltesz, H. Potzl and L. Peterson. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In *in EuroSys*, pages 275–287, 2007.

[23] W. v. Hagen. Professional xen virtualization. In *Wrox Press Ltd. Birmingham*, 2008.

[24] W. Emeneker and D. Stanzione. Hpc cluster readiness of xen and user mode linux. In *in 2006 IEEE International Conference on Cluster Computing*, pages 1–8, 2006.

[25] Naode Ma Liang Zhou Yajun Li, Yuhang Yang. A hybrid load balancing strategy of sequential tasks for grid computing environments. In *Future Generation Computer Systems*, pages 819–828, 2009.

[26] Yaozu Dong Xiantao Zhang. Optimizing xen vmm based on intel virtualization technology. In *2008 International Conference on Internet Computing in Science and Engineering (ICICSE 2008)*, pages 367–374, 2008.

[27] Asit Mallick Jun Nakajima Kun Tian Xuefei Xu Fred Yang Yaozu Dong, Shaofan Li and Wilfred Yu. Extending xen with intel virtualization technology. In *Journal, ISSN, Core Software Division, Intel Corporation*, pages 1–14, 2006.

[28] et al. Z. Hai. An approach to optimized resource scheduling algorithm for open-source cloud systems. In *in ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*, pages 124–129, 2010.

[29] Amazon http://aws.amazon.com/ec2/.

[30] Cloud computing http://en.wikipedia.org/wiki/Cloudcomputing.

[31] Openstack http://openstack.org.

[32] Opennebula http://www.opennebula.org.

[33] Kvm http://www.linux-kvm.org/page/MainPage.

[34] Xen. http://www.xen.org/.

[35] Lxd. https://linuxcontainers.org/lxd/.

[36] Monjeaud C. Sallou O. Go-docker: A batch scheduling system with docker containers. In *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*, pages 514 – 515, 2015.

[37] Libin Zhao Di Liu. The research and implementation of cloud computing platform based on docker. In *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*, pages 475 – 478, 2015.

[38] Hashim S.J.B. Memari N., Samsudin. K.B. Towards virtual honeynet based on lxc virtualization. In *Region 10 Symposium, 2014 IEEE*, pages 496 – 501, 2014.

附錄 A

環境安裝

1. 硬體規格與軟體版本

   CPU: Intel Celeron(R) G1610 2.60GHz

   RAM: 4 GB

   HD: 1TB

   Network: 100M/1000M bps Ethernet

   OS: Ubuntu 14.10

   LXD 版本：LXD 0.20

   LXC Contain Guest OS: ubuntu trusty amd64 –alias ubuntu

2. 本實驗須建立三台虛擬伺服器 (VM):

   一台 Namenode(Master): 主機名稱 host01: 192.168.1.93

   一台 Datanode(Slave): 主機名稱 host02: 192.168.1.47

   一台 Datanode(Slave): 主機名稱 host03: 192.168.1.52

3. 建立金鑰

   sudo apt-add-repository -y ppa:ubuntu-lxc/daily

   sudo apt-add-repository -y ppa:ubuntu-lxc/lxd-git-master

4. 安裝 LXD

   sudo apt-get update

   sudo apt-get install lxd

sudo apt-get remove lxcfs

sudo service lxd start

5. 檢查 LXD 群組

cat /etc/group | grep lxd

6. 將使用者加入 lxd 群組，以使用者帳號是 arthurtoday 為例

sudo adduser arthurtoday lxd

7. 建立遷移的容器

lxc init ubuntu migratee

lxc profile create migratable(Ubuntu 14.04 指令需加 config)

lxc profile edit migratable(Ubuntu 14.04 指令需加 config)

8. 修改連線設定

ame: migratable

config:

raw.lxc: |

lxc.console = none

lxc.cgroup.devices.deny = c 5:1 rwm

lxc.start.auto =

lxc.start.auto = proc:mixed sys:mixed

security.privileged: ”true”

devices:

9. 加入連線設定

lxc config profile apply migratee migratable

10. 加入 remote host

host01:sudo lxc config set core.httpsaddress 192.168.1.93

host01:sudo lxc config set core.trustpassword ***** 192.168.1.93

host02:sudo lxc config set core.httpsaddress 192.168.1.47

host02:sudo lxc config set core.trustpassword ***** 192.168.1.47

host03:sudo lxc config set core.httpsaddress 192.168.1.52

host03:sudo lxc config set core.trustpassword***** 192.168.1.52

host01:sudo lxc config set core.httpsaddress 192.168.1.93(Ubuntu 14.04 指

令 lxc remote add lxd 192.168.1.93:8443)

host01:sudo lxc config set core.httpsaddress 192.168.1.47

host01:sudo lxc config set core.httpsaddress 192.168.1.52

11. 複製連線方式到其它 host

lxc config profile copy migratable host02:

lxc config profile copy migratable host03:

12. 啟動虛擬機 Containers

lxc start migratee

13. 移動虛擬機

lxc move host01:migratee host02:migratee

完成