國立交通大學

資訊科學與工程研究所

碩士論文

NCTU CStack：OpenStack 與 Ceph 的整合與應用

NCTU CStack:An integration and application of OpenStack and Ceph

研 究 生：蔡權昱

指導教授：蔡錫鈞 教授

中華民國一零一年八月

NCTU CStack：OpenStack 與 Ceph 的整合與應用

NCTU CStack:An integration and application of OpenStack and Ceph

研 究 生：蔡權昱　　Student:Chuan-Yu Tsai

指導教授：蔡錫鈞　　Advisor:Shi-Chun Tsai

國立交通大學

資訊科學與工程研究所

碩士論文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

November 2012

Hsinchu, Taiwan, Republic of China

中華民國一零一年十一月

摘　要

雲端運算是近年來非常熱門的話題，隨著需要處理的資料愈來愈龐大，各個機構所設置的資訊中心也愈來愈複雜。因此，關於資料如何安全的保存，以及硬體設備如何有效的利用，也變成非常重要的議題。

在資料安全性方面，Ceph 為一新崛起的分散式儲存系統，設計之初即把把底層的硬體離線維護視為常態，並不會因為小數量的機器故障而造成整個系統停擺，進而提供了很好的資料安全性。

在硬體利用率方面，虛擬化是一個很好的解決方式。傳統的虛擬化軟體可以在一台機器上開啟多台虛擬設備，但若要管理一堆虛擬機的集合，分配不同的虛擬機給不同的單位使用，以及處理虛擬機與實體機的網路連線等，則非常麻煩。因此，虛擬化的主要廠商皆推出了相對應的產品，如 VMware ESXi 搭配 vCenter 來建構虛擬化的資料中心，或是 Microsoft Windows Server 的 Hyper-V 以及 Citrix XenServer 等，都可以快速的提供企業自己的私有雲服務，但是昂貴的授權費用，以及對硬體設備的規格需求相對嚴苛的限制，也讓不少機構望之卻步。

OpenStack 為近年由 NASA 和 Rackspace 共同開發的開放原始碼軟體，可提供任何人建立免費的雲端運算服務。我們利用了 OpenStack 可快速建立私有雲的特性，搭配了 Ceph 分散式儲存系統，在交通大學資訊技術服務中心建立了一個名為 NCTU CStack 的系統，提供免費且高度彈性化的校園雲端運算和雲端儲存服務。

本文包含了基本的 OpenStack 和 Ceph 配置與部署。我們提出新的虛擬機網路模式 (PHA)，讓運算節點可以使用私有 IP 提供服務，大量節省 IP 的消耗，並且在正常使用情境下仍然保有 HA 模式的各種好處；另外使用 rgwauthAPI 來整合兩個系統上專案和帳號的權限，讓 OpenStack 可對外提供相容 Amazon S3 協定的物件儲存服務。

Abstract

Recent years, cloud computing becomes a hot topic. With the rapid growth of information day by day, the more complex data centers will be set by various institutions. Therefore, how to save data safely and utilize hardware devices efficiently are important issues.

Ceph is a new brand of distributed file system. At the design phase, Ceph regards the fault of underlying hardware as normal, and no need to shutoff the whole system when a few machine is scheduled for maintenance offline, thus Ceph can quite ensure the safety of data.

Virtualization is a good way to improve the hardware utilization. Traditional virtualization software can easily boot multiple virtual machines on a single server, however, it is hard to manage so many virtual machines, deal with the network connection between virtual machines and different host servers. Hence all the major virtualization companies release corresponding products, such as using VMware ESXi with vCenter to construct the virtualization data center. The other suits, for example Hyper-V role in Microsoft Windows Server and the Citrix XenServer, can quickly provide enterprise private cloud services, but many institutions may flinch from expensive licensing fees and stringent restrictions about hardware specifications.

OpenStack is an open source software which is developed by NASA and Rackspace at first. Anyone can use OpenStack to create a free cloud computing services quickly. At last, we establish a private cloud named NCTU CStack system in NCTU ITSC. NCTU CStack is integrated by OpenStack and Ceph storage backend, providing free, highly-flexible campus cloud computing and storage services.

This thesis includes the basic configuration and deployment flow about OpenStack and Ceph. We design the new networking mode(PHA), using PHA mode can greatly reduce the consumption of public IP, and it works like HA mode in most scenario. Moreover, we use rgwauthAPI to integrate the authentication between OpenStack and Ceph, and make our system support the object storage service which is Amazon S3 API compatible.

致　謝

時光荏苒，轉眼間碩士班生活已接近尾聲，從懵懂的大學生到有自己想法的研究生，這段時間內因為有許多人的支持與協助，我才得以完成碩士論文研究。首先要謝謝我的指導教授蔡錫鈞老師，老師的管理風格富有彈性，讓我自由選擇喜歡的研究主題，做實務不忘理論，是我在您身上學習到的精神。研究過程中難免會遇到瓶頸，對於任何研究上的需求，老師總是全力支援，從”要玩就玩真的”這句話中，我不斷被老師的熱忱所感染，您的鼓勵是我持續向前的動力。學業之外，老師也時常關心學生在異鄉的起居及家庭狀況，讓我這個外地遊子倍感溫馨。謝謝交大資訊技術服務中心俊憲學長及所有幫助維護系統運作的工程師，因為有你們的幫忙，這篇論文才能順利完成。

很感謝在交大兩年裡，曾經幫助我的人，特別是實驗室裡每天生活在一起的大家。特別謝謝名全學長聽我預報論文，教導我投影片製作技巧，並且處理實驗室的大小疑難雜症，讓實驗室得以順利運作；雖然相處的時間很短，但也謝謝昶諄學長熱心地替剛進實驗室的我指點迷津，開啟我的實驗室生活；謝謝李睿學長總是在我閱讀論文遭遇瓶頸時替我理清演算法細節；謝謝旻錚學長總是在我報告問題回答不出來時解救我，並且總是熱心的跟我討論及整理研究的可行方向。

也感謝一同互相幫忙兩年的宜謙，你總是可以跟我即時討論新的想法，給我許多不錯的點子，恭喜我們順利走過這兩年，完成學業。實驗室的方智誼與邱韜瑋以及其他碩一學弟們當然也不能忘記，你們的協助我銘記在心，真心祝福你們研究順利。

離家到新竹求學，婉馨的支持與陪伴是我前進的力量，因為有妳的歡笑聲，在我陷入低潮時才能鼓勵我並使我重拾研究的熱情。最後，謹以此文獻給我摯愛的雙親。

目　錄

表 目 錄

圖 目 錄

第一章　緒論

從第一台電腦被發明以來，人們使用電腦的習慣一直隨著科技演進而改變著。從早期的 IBM 大型電腦 (Mainframe) 時期，使用者利用終端機連線回大型電腦工作，所有工作都在遠端的大型電腦裡完成，僅有輸入輸出功能的終端機幾乎沒有任何運算能力。一直到積體電路的出現，電腦主機的體積逐漸縮小，價格也越來越便宜，人們才開始負擔得起購買自己的電腦。

二十世紀末，個人電腦 (Personal Compute) 運算能力越來越強，雖然各種軟體如雨後春筍般被開發出來，但是可以執行的程式和可以處理的資料規模，還是受限於個人電腦的規格等級。此外，這時的網路也尚在萌芽階段，頻寬與普及率仍不足以方便的使用，文件檔案的分享依然大量利用可移除式裝置，不同電腦之間檔案的同步與保存成為一個熱門的議題。

二十一世紀，情況又大大改變了，網路已成為大部分人日常生活中不可或缺的一部份。隨著高速網路的普及，網路徹底的顛覆了人們使用電腦的習慣，有些作業系統甚至直接與瀏覽器結合，讓使用者靠一個瀏覽器就可以在網路上完成文書、上網、遊戲、會議、多媒體等所有原本在本機端處理的工作。

處理資料的核心又再一次回到了遠端的機房，只是這一次負責服務的不再是一台大型的超強電腦，而是遍步世界各地的電腦群集 (cluster)。各種支援橫向擴展 (scale-out) 的軟體與系統開始被研究和使用，每天共同處理著高達數以 Pera Byte 計的資料，如何有效率的處理這些資料，以及如何有效率的管理處理這些資料的機器，其中不少研究方向值得我們去探討。

1.1　研究背景與動機

隨著網路速度的持續發展，使用者越來越習慣在網路上完成所有日常工作，而不用去擔心本地機器的運算能力強不強或是儲存空間夠不夠的問題，因此網路服務提供商相對的就必須持續增加自己的機器數量來維持一定的運算能力，以及購買更多的儲存

設備。

然而維護越來越多的機器設備除了會導致管理成本快速增加，機器的部署配置與相關的網路設定也會造成維護人員的負擔。另外，某些服務並非常態性的被使用 (如年節的網路訂票、報稅，學校的選課等)，這些需求如果使用專門的機器來提供服務，會導致離峰時段機器的閒置，電力的浪費，硬體使用沒有效率。

虛擬化提供了一個解決上述問題的途徑。近年來硬體虛擬化技術的成熟，讓虛擬化的效率獲得提升，一台實體機器上可以執行數台至數十台獨立的虛擬機來提供不同的服務。透過各種虛擬化軟體，可以隔離實體機器上的各種服務，經由虛擬機的動態遷移 (Migration) 可以隨時調整實體機器的負載，讓使用效率變高，確實解決了不少問題。

可是隨著虛擬化軟體的使用，管理人員必須同時管理實體機器與虛擬機器之間，以及虛擬機器與虛擬機器使用者之間的關係，系統複雜程度反而比以前來的高。隨著服務規模的擴大、機器的增加，這些軟體昂貴的授權費用也額外造成不少的負擔。

另外一個解決辦法是使用現有的雲端運算服務，如 Amazon AWS EC2、Microsoft Azure Platform 或是 Hinet Hicloud 等。利用租用雲端運算能力 (虛擬機) 與儲存空間來架設所需的伺服器，或是直接利用現有雲端廠商提供的平台 (如 Google App Engine、Heroku) 來撰寫程式，付出可接受的費用將底層的維護管理成本外包，自己只需要專心在服務的開發及宣傳，並且享有快速擴展規模，根據使用量付費等好處。

可是此方式仍然有相當大的隱憂，常見的則是因為資料隱私性的關係，許多單位組織不願意將資料放到遠端的公有雲上，而且根據使用量的增加，相關的費用也會逐漸提高，我們也無法針對雲端系統要求客製化的動作，這些原因也是許多人轉而使用雲端運算軟體來架設私有雲的原因。

1.2 系統發展之目標與成果

我們希望在不用添購太多最新的硬體的前提之下，利用交大資訊技術服務中心現有的硬體設備，以及搭配免費的 Open Source 雲端運算軟體－ OpenStack Essex 來建立交通大學的私有雲，提供校內師生一個可快速使用，且可彈性擴展的雲端運算服務。

考量到 OpenStack Swift 僅能提供物件儲存的服務，我們整和了 OpenStack 以及另一個發展中的分散式檔案系統－ Ceph，來取代 Swift。除了提供原本的物件儲存功能之外，進一步藉由 Ceph 可以提供多種網路儲存解決方案的特性，當作 OpenStack 各個單元的儲存後端，讓 OpenStack 裡需要額外儲存空間的服務例如：Image、Volume、虛擬機的 shared storage 等，都能與 Ceph 做整合。使用 Ceph 來負責所有的儲存工作，管理者就不用擔心是否有哪個服務的可用空間即將用罄，也不需要依據不同服務的特性而

為 OpenStack 架設不同的網路儲存服務。

除了參考先前 Sébastien[21] 對於 OpenStack 裡的部分服務和 Ceph 的整合過程之外，我們另外撰寫了一個 Python 模組－ rgwauthAPI，來負責同步 OpenStack 與 Ceph 的專案和帳號的權限；Ceph RADOS 是 Ceph 上提供物件儲存的服務名稱，我們修改 OpenStack Dashboard(Horizon) 使其可以直接使用 Ceph RADOS Gateway 所提供的 Swift API，並使用相同的權限認證對外提供相容於 Amazon S3 的物件儲存服務。

另外考量到日漸枯竭的 IPv4 位址，我們修改了 OpenStack 的 network 子系統，發展了第三種網路模式：PHA 模式，讓運算節點可以使用私有 IP 提供服務，大量節省 IP 的消耗，並且在正常使用情境下仍然保有 HA 模式的各種好處 (如避免單點錯誤)，而且在這種模式之下，比傳統 HA 模式能提供更好的安全性，可以避免為了支援虛擬機動態遷移而產生的連接端口暴露。

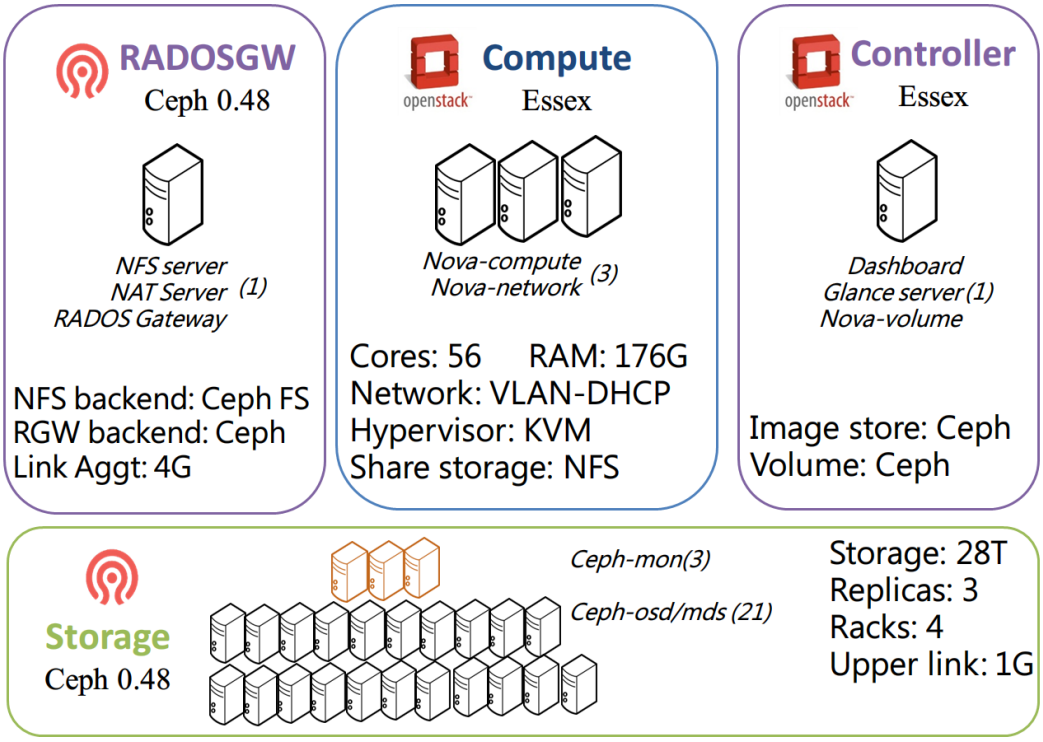

圖 1.1: CStack 部屬示意圖

我們最終在交大資訊技術服務中心部署了一個完整的雲端運算和儲存系統 (圖 1.1)，我們利用 20 台由台達電子 (Delta Electronics, Inc) 捐贈的客製化機器和 4 台 HP DL360-G5 來運行 Ceph 檔案系統，提供約 28TB 的儲存空間 (圖 1.3)；另外使用 1 台 HP DL380-G5 以及兩台 Acer 380 F2 運算節點共擁有 56 個核心跟 176GB 的記憶體來提供虛擬機使用，兩台 HP DL380-G5 當作控制節點以及提供物件儲存服務 (圖 1.2)，整個系統已經上線供中心工程師以及有相關課程需要的老師及學生們使用。

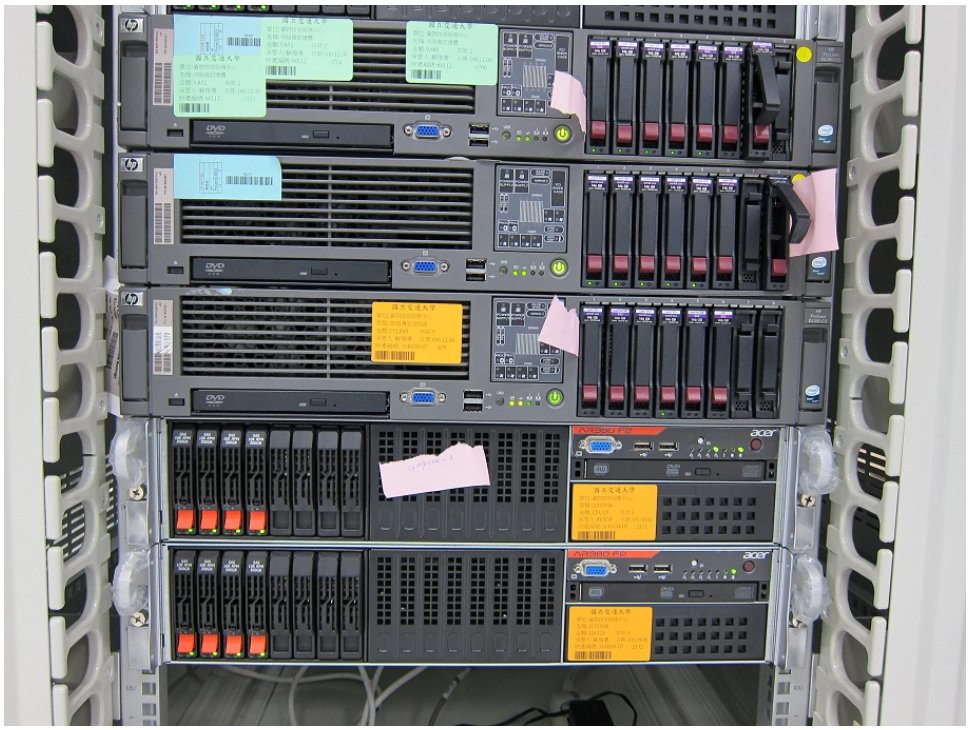

圖 1.2: 運算節點示意圖

1.3 各章節介紹

第二章一開始介紹了雲端運算的相關定義，並說明我們的系統是歸類為哪一種類型的雲端服務；另外我們也介紹了 NCTU CStack 的兩大子系統－ OpenStack、Ceph 的相關背景細節，和子系統內的專案介紹。

第三節則是將 NCTU CStack 系統做一個大綱上的介紹，並藉著不同的面相來說明系統的組成元件、使用情境和部署架構，並在 3.3節將虛擬機網路架構獨立出來特別說明，幫助後續章節的理解。

第四章我們依照順序分別介紹了 Ceph 儲存後端和 OpenStack 各個專案的安裝細節，以及在各小節後面補充兩個系統的整合方式和測試方法，並且在 4.3節說明 OpenStack 裡最重要的設定檔 nova.conf 中幾個相當重要的參數以及觀念。

第五章專注在說明使用 Ceph 來提供 OpenStack 物件儲存服務的部分，並且針對兩個系統權限上問題來說明並提出解決辦法。

我們在第六章呈現最終的成果，並將原本的功能以及新的功能以圖片搭配說明來作呈現，介紹 NCTU CStack 的使用方式。

最後，將在第七章作個簡單的結論。

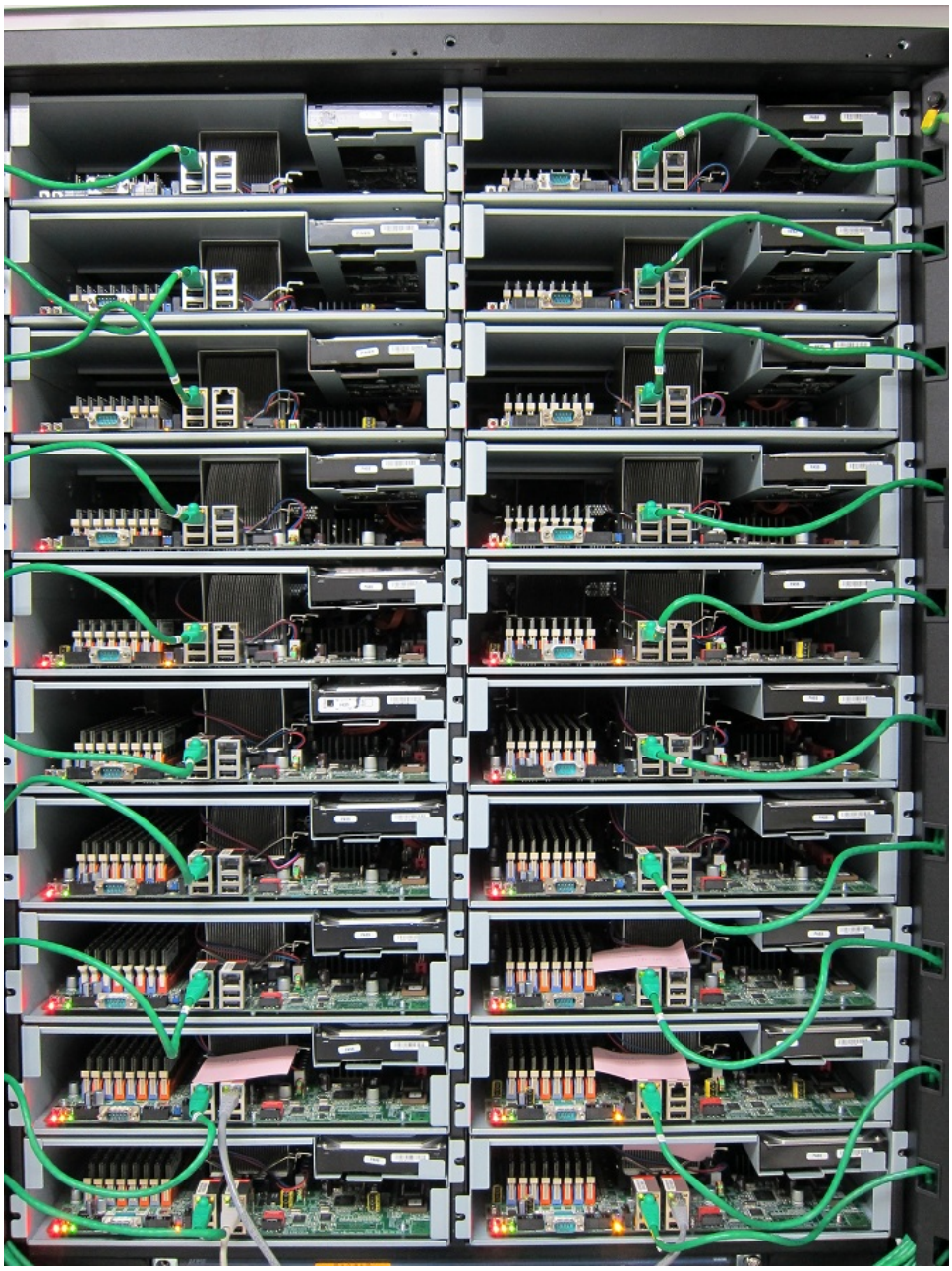

圖 1.3: Ceph cluster 示意圖

第二章　背景知識

這個章節將說明有關雲端運算的基本解釋，並且介紹我們使用的兩個主要系統 OpenStack 和 Ceph 的開發歷史以及用途。

2.1　雲端運算

雲端運算 (Cloud Computing) 為近年來很熱門的一個研究領域，美國國家標準技術研究所 NIST 有對他作了以下三種服務模型的定義 [19]：SaaS、PaaS 以及 IaaS，其中 IaaS(Infrastructure as a Service) 模型即是我們的系統 NCTU CStack 提供的服務種類，我們提供了運算 (開啟虛擬機)、儲存、網路等資源給使用者在上面運行任意的軟體或作業系統，使用者不需要對底層的硬體架構做任何的管理，只需專注於開發或維護作業系統及其上面提供的服務。

關於部署的型態，我們屬於 NIST 定義的 Private Cloud，我們的系統並不對外開放，只提供校內的師生使用。而且為了與傳統的線上虛擬私有主機 (VPS) 做出區分，NIST 也定義了雲端運算的幾種基本特性：

1. On-demand self-service －使用者可以自行依照需求，不用透過管理者而取的額外所需的資源，如網路硬碟、新的虛擬機等。

2. Broad network access －提供一個標準的存取方式，不論運算能力的好壞，讓各式各樣的機器都可以使用雲端服務。

3. Resource pooling －資源將以一個大集合的方式來提供 (pooled resources)，使用者僅能控制很粗淺的選項，但無法得知細節。例如使用者開啟虛擬機器時無法得知實體機器確切的位置，但可以指定想開啟哪一個洲或是哪一個資料中心的虛擬機器。

4. Rapid elasticity －大部分的功能都要可以彈性的甚至自動的取得或釋放，對使用者來說相當於可以無限制的使用。

5. Measured service －使用的資源要可以被控管，並產生使用明細，讓管理者跟使用者可以自由地察看，並掌握使用量。

我們的系統因為是使用 OpenStack 來建置，因此也符合了上述所列的五種特性。

2.2 OpenStack

OpenStack 為 2010 年 NASA 與 Rackspace 共同推出的系統，以 Python 語言撰寫，並使用 Apache 授權開放原始碼，讓所有人都可以利用這套系統自由的部署自己的雲端運算與儲存環境 [16]。目前的支援廠商已經超過 150 家，其中包含了 AT&T、IBM、Intel、HP、Cisco 等著名廠商，並以大約每半年推出一次新版本的速度更新中，最新的穩定版本 Folsom(第六版) 在 2012 年九月底釋出，有超過 330 作者替 Folsom 貢獻原始碼，相較於 Essex(第五版) 大致上為 Diablo(第四版) 的穩定性更新和錯誤修正 [14]，Folsom 在每個專案上皆有相當多的功能更新，並新增了兩個獨立的專案－網路及磁碟 (Block Device)，讓 OpenStack 的運作更臻完善。

OpenStack 一開始由 Nova 與 Swift 兩個專案組成，各自提供雲端運算與資料儲存的功能，接著發展到目前已分成數個核心專案，分別有：Nova 運算、Swift 物件儲存、Glance 映像、Keystone 認證、Nova-network 網路 (Quantum)、Nova-volume 磁碟 (Cinder)、Dashboard 網頁管理等，這些核心專案再搭配資料庫服務 (如 MySQL、SQLite) 以及 Advanced Message Queuing Protocol(AMQP) 服務 (如 RabbitMQ、Apache Qpid) 和 Web VNC 服務 (如 noVNC) 等許多其他 Linux 上的其他服務一起組成一個雲端運算系統。

各個專案的運作細節，以及部署方式將會在本論文後續篇幅使用到時一起介紹，亦可參考 1.3節查詢。

2.3 Ceph

Ceph 是 Sage A. Weil 等人在 2006 年發表的一個擁有高度延展性、良好效能的分散式檔案系統 [23]。整個系統在良好的物件儲存能力基礎上，進而再提供 Block Storage 和 File Storage，提供現有雲端服務多元化的支援。

Ceph 在設計之初即考慮可靠性問題，並把底層設備的硬體錯誤視為正常且一定會發生的現象，因此設計了多項錯誤偵測機制，確保資料的正確性。除了一般的硬體錯誤，作業系統會自動回報之外，Ceph 系統包含了小部分 Monitor Cluster，主動檢查是否發生間斷性的網路問題，監控整個系統的狀態，也負責管理系統設備的增減，讓管理者

可以動態的新增機器來擴充儲存空間，也不用擔心小部分的機器異常離線而導致整個檔案系統損毀。

Ceph 將檔案切成數個物件，合併數個物件成一個存放集合 (Placement Group)，管理者可以自訂 PG 的 Replica 數量，並利用 Placement Rule 來決定 Replica 的存放位置 (如圖 2.1)，管理者可以將實體機器的部署情形 (如那些機器使用共同的電源迴路、Switch) 與檔案備援機制共同考慮，進一步提高資料安全程度。

Placement Rule 由一連串有階層關係的 Controlled Replication Under Scalable Hashing(CRUSH) Map[24] 所組成，表 2.1是個簡單的例子：管理者可以在不同規模的設備分類中用 CRUSH Map 來亂數選擇所需數量的大單位，然後從被選擇出來的大單位中再遞迴的選擇所需的小單位，最後把這個 rule 執行完畢就會擁有一推最終單位，物件就會被儲存在這些最終單位上。

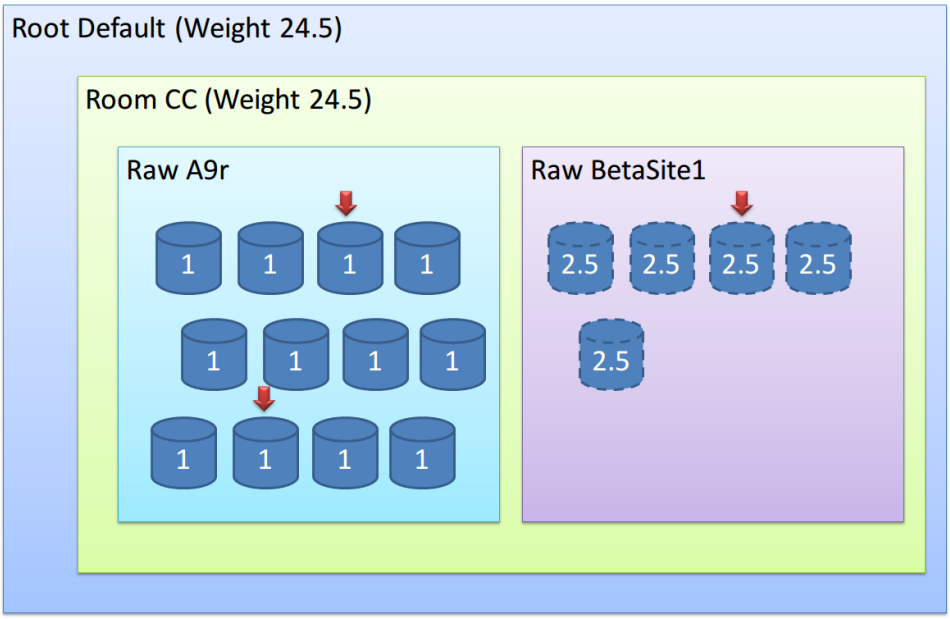

圖 2.1: 以此圖為例，圖中所標示數字為管理者定義每台 OSD 的權重 (Weight)，因為 BetaSite1 的 OSD 容量為 A9r 的 2.5 倍，因此我們可以將此資訊寫進 Placement Rule 中，利用權重控制每台機器所存放的檔案大小，並控制 PG 的 Replica 需要放置一份到 BetaSite1，其餘兩份則存放至 A9r 中，避免因為 BetaSite1 的 Switch 故障，使得有部分檔案暫時無法存取。

2.3.1 Ceph 服務

RADOS(Reliable Autonomic Distributed Object Storage) 為 Ceph 的物件儲存服務的名稱，是 Ceph 其他服務的後端，存放這些物件的機器就叫做 Object Storage Device(OSD)，

Action	Resulting Vector	說明
take(root)	Default	從 root 開始
select(1,room)	Room2	在機房這個單位挑選一個機房
select(3,cabinet)	cab21 cab23 cab24	從上面 Room2 這個機房裡面再挑選三個機櫃
select(1,disk)	disk2107 disk2313 disk2437	從上面挑選出來的機櫃中各自挑選一個磁碟
emit		送出結果

Table 2.1: 一個簡單的 Placement Rule 例子

是 Ceph 裡實際負責儲存資料的基本單位，OSD Cluster 底層可以選擇使用 XFS、Btrfs 等主流的檔案系統。存取 RADOS 上的物件除了可以使用 rados 這個指令，Ceph 也提供了 python-rados、java-rados 等語言的 library 可供使用。

RBD(RADOS Block Device) 可將多個 RADOS 上的物件集合成系統上的一個 Block Device 來使用，存取的方式可透過 Linux 2.6.36 之後內建的 Kernel Module 或是 librbd library。使用 RBD 指令建立的 Block Device 可依需求變動大小、建立快照。

Ceph FS 是 Ceph 提供的另一種服務型態 -File Storage，利用 Ceph 的分散式檔案系統協定，使用者可利用 Linux Kernel 內的 Ceph Client 或透過 User-Space 的 Ceph-FUSE 來掛載 Ceph FS。Ceph 上的 Metadata Server Cluster(MDS) 負責處理檔案系統的 inode、紀錄檔案大小、修改時間、檔案權限等等的資訊，MDS 除了利用 Memory Cache 來加速 metadata 的處理與回應，也會將它寫入 journal，存回 OSD Cluster 來避免資料遺失。

RADOS Gateway 利用了 Apache/Nginx 等 Web Server 對外釋出了 RADOS 的另一種存取方式 -ReST Interface。使用者可以透過 OpenStack Swift API 或是 Amazon S3 API 來向 RADOS Gateway 請求服務，RADOS Gateway 負責將物件轉送給後端的 Ceph RADOS，或是從後端取出物件供使用者下載，並完成與使用者的連線對話。表 2.2比較了 Ceph 的各種服務與常見的儲存服務的對應關係 [18]。

Object Storage	Block Storage (SAN)	File Storage (NAS)
Ceph RADOS	Ceph RADOS Block Device(RBD)	Ceph FS
Amazon S3	Amazon Elastic Block Storage(EBS)	
OpenStack Swift	OpenStack Nova-Volume (Cinder)	
	iSCSI, AoE	NFS, Samba

Table 2.2: Ceph 提供了三種常見的網路儲存方式的解決方案

2.4 指令碼說明

由於本論文之後的內容將會充滿執行指令與顯示結果，我們將所有的指令皆以有底色的方框表示，並且在沒有特別指名的情況之下，命令提示字元 # 開頭的指令皆代表以 root 身分執行，且指令執行時所在的主機名稱將顯示在命令提示字元之前：範例如下：

```
// 在Host1這台主機上執行ls這個指令
Host1# ls
```

沒有底色的方框代表指令執行結果，或是編輯器修改的內容 (設定檔)。

```
// 此例為執行ls後所列出的檔案與資料夾
file1 file2 dir1 dir2 dir3
```

此外，NCTU CStack 系統不論 Ceph 或是 OpenStack 皆安裝在 Ubuntu 12.04 amd64 的作業系統上，因此我們介紹一下本篇論文常用的指令：

1. aptitude - Ubuntu 內建的套件管理程式，install 參數後接著欲安裝的套件名稱

   ```
   Host1# aptitude install <Package1> <Package2> ...
   ```

2. vim - 額外安裝的純文字編輯軟體
3. scp - 遠端檔案複製軟體，將檔案從第一個參數複製到第二個參數，參數內冒號之前代表主機名，冒號之後代表檔名

   ```
   Host1# scp <Local source file> Host2:<Remote destination file>
   Host1# scp Host2:<Remote source file> <Local destination file>
   ```

4. tee - 將標準輸入的資料流寫入指定的檔案內，加上 -a 參數代表不會覆寫檔案，僅將資料流加入檔案末端
5. chown - 改變檔案的擁有者跟擁有群組

   ```
   Host1# chown <New owner>:<New group> <File>
   ```

第三章　系統架構

本章將描述我們建置的 NCTU CStack 系統的整體架構，以及 Ceph 系統的子架構，和介紹虛擬機的網路架構。

3.1　CStack 系統

圖 3.1是從 OpenStack Compute Administration Manual[15] 的 Logic Architecture 修改而來。右半部四個獨立的方框是標準的 OpenStack 元件，由 Dashboard 提供網頁介面接受使用者的建立虛擬機需求，和管理者的維護操作，由 Image 元件提供使用者指定的映像檔給運算節點利用 Compute service 來運行虛擬機，且所有的元件的互動皆透過 Identity 元件來進行認證及授權，如此提供了 IaaS 所需的基本元素。

有鑑於傳統的 OpenStack 有多個元件需要額外獨立的儲存空間，如存放開機映像檔的 Image provider、存放動態磁碟的 Volume provider、支援虛擬機動態遷移所需的共享儲存空間 (shared storage)，以及原本的物件儲存服務 (Swift) 等，若是各別安裝規劃，除了系統複雜度極高之外，後續空間的維護以及擴張也是相當麻煩。因此我們整合了 Ceph 與 OpenStack，讓 Ceph RADOS Gateway 搭配我們撰寫的 rgwauthAPI-這個 Dashboard 模組，取代了原本的 OpenStack 物件儲存服務，並可以和 Identity 上的權限同步，更進一步對外提供相容於 Amazon AWS S3 的物件儲存服務 (參考第 5章)；也利用了 Ceph RBD 來當作開機映像檔和動態磁碟的儲存後端，統一使用 Ceph 來管理所有 OpenStack 產生的資料，節省管理者的負擔 (參考 4.2.2、4.2.3小節)；共享儲存空間則使用 Cephfs 搭配 NFS 來提供，經過單一的 NFS Server 轉送 (Reexport) 的共享儲存空間則是可以穩定的讓每台運算節點使用 (參考 4.2.4小節)。

觀念上的元件區分有助於我們選擇那些元件要安裝在同一台實體機器上。以使用情境來說明，我們的系統除了可以提供 IaaS 的使用者透過 Dashboard 來開啟虛擬機外，也讓 Object Proxy Server 提供了物件儲存的服務，當然，一般的使用者也可以直接使用虛擬機上架設的服務。實際上，我們將實體機器分成兩個 cluster 以及兩台管理節

點。由 Intel Xeon E5310 以上的機器搭配 48GB 以上的記憶體組成運算節點 (Compute Workers) 來提供虛擬機們穩定的運作環境；由擁有 16GB 記憶體的客製化機器提供分散式的 Ceph 來提供可高度水平擴展的儲存空間；另外兩台管理節點各扮演著 Controller 或 Object Storage Proxy 的功能。這些組成了一個完整的雲端運算與儲存的基礎建設 (圖 3.2)。

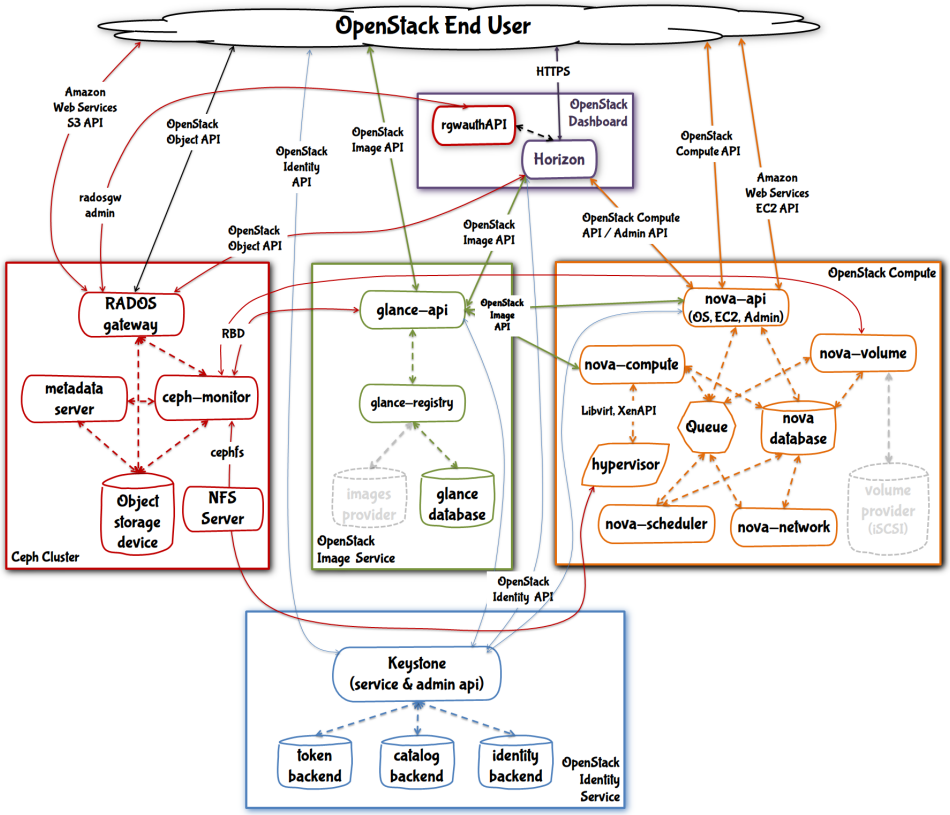

圖 3.1: CStack 系統概念圖

3.2 Ceph 系統架構

圖 3.3顯示了 Ceph 分散式檔案系統由三個部分組成：Object Storage Device(OSD) 來負責所有的物件儲存，Metadata Server(MDS) 負責提供 Ceph FS 來讓遠端設備掛載使用，Monitor 則負責管理整個 Ceph 和監控所有機器的運行狀況。

另外，圖 3.3也顯示了 NCTU CStack 系統中 OpenStack 內的服務和 Ceph 之間的關係。我們使用 Ceph FS 來當作共享儲存 (shared storage) 用來存放虛擬機資料，讓虛擬機擁有動態遷移 (live migration) 的能力；使用 Ceph RADOS Block Device(RBD) 來當作 Image 服務和 Volume 服務的後端，比起存放在服務端的本機硬碟增加了映像檔和快照的資料可靠性；另外使用 Ceph RADOS Gateway 來取代 OpenStack Swift 程式，除了免

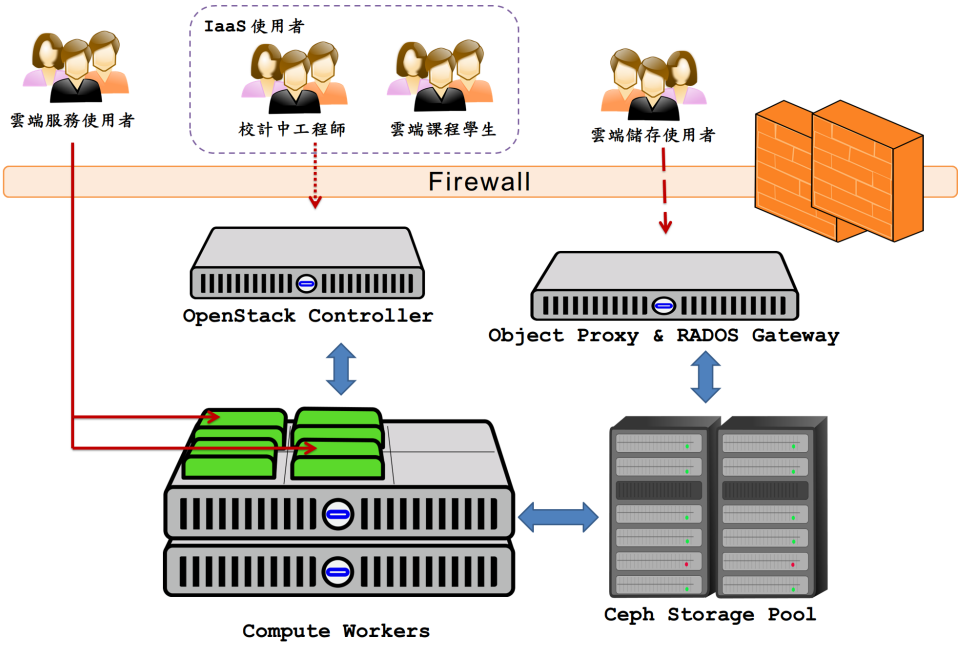

圖 3.2: CStack 系統佈署架構圖

於同時安裝兩個分散式檔案系統，降低系統複雜度之外，還可以比 Swift 提供更多的功能。這樣的架構可以讓整個 CStack 系統都充分利用 Ceph 這個可靠的分散式檔案系統的好處。

3.3 修改後的虛擬機網路架構

圖 3.4是參考 Vishvananda[7] 在"Networking in Nova" 文中的圖修改而來的，它反映了 NCTU CStack 系統中不同專案 (tenant) 的虛擬機之間的網路互動關係。我們使用了 OpenStack 裡的的 VLAN 模式，搭配 High Availability Option，擁有一個安全且高可用性的網路環境，但為了管理上的方便，在沒有安全疑慮下我們僅使用一台 Switch[1] 來供虛擬機和 Nova 的服務程式使用。

如同右上角三種線條所示，我們使用 Tagged VLAN Switch 來將 Nova 服務所在的網路 (VLAN ID 0) 與虛擬機們使用到的網路 (VLAN ID > 100) 切開，虛擬機若遭受外部網路攻陷時，並沒辦法假裝成 Nova 服務的一部份來竊取其他人的資料 [22]；同時虛擬機之間也用不同的 VLAN ID 區分，不同專案的虛擬機也無法互相偷聽，避免掉一些常見的網路攻擊 (如 ARP spoofing 等) 在專案間快速擴散的危險。

傳統 OpenStack 的網路架構僅用單一的節點安裝 nova-network 服務來處理 (轉送)

[1] 若是在系統的規模成長之後必須使用多台 Switch，Switch 之間互相連接的連接埠必須設定為 Trunk 模式，傳送不同 VLAN 的封包。

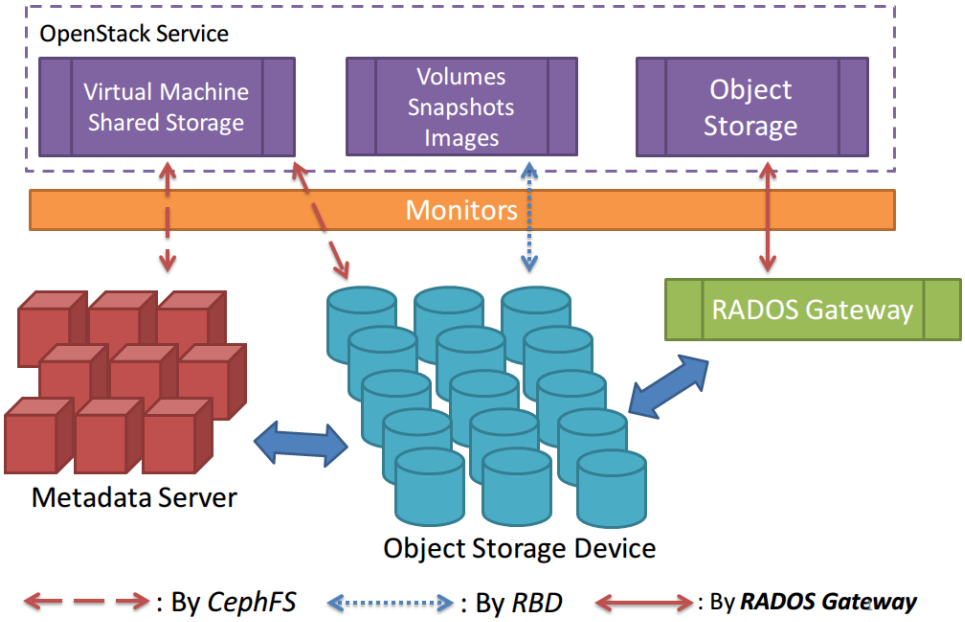

圖 3.3: Ceph 與 OpenStack 關係圖

所有虛擬機與外部網路的流量，但這意味著一旦 nova-network 服務停止或機器離線維修，所有使用者即與虛擬機立刻失去聯繫。High Availability(HA) Option 就是為了避免單點錯誤 (SPOF) 即造成整個 OpenStack 的虛擬機網路完全癱瘓所設計的模式。由圖 3.4上的三個實體節點所示，每個運算節點 (Compute Worker) 除了安裝啟動虛擬機必要的 nova-compute 服務之外，我們也安裝了 nova-network 服務來開啟 HA Option，讓每個虛擬機的網路直接利用宿主機轉送出去，如此就不會因為一台機器的停擺而造成所有虛擬機的網路中斷。另外，metadata-api 也是因應這種模式下而需要每一台運算節點都安裝的服務，詳情請參考 4.3節的 Metadata API 子小節。

除此之外，為了提高系統安全性，以及實際使用情形的限制，我們建立了一種名為部分 HA(PHA) 的網路模式。在此環境之下，我們延用了舊有的 HA 模式，並讓每一台運算節點都只有私有 IP，讓使用者依照專案需求再動態的綁定公開 IP 到運算節點上，若日後運算節點的數量一多，可以替單位裏原本就不夠多的 IPv4 省下可觀的消耗，也避免了外部網路可以直接連線到運算節點的風險。

表 3.1顯示了三種虛擬機網路模式的優缺點。沒有使用 HA 模式的網路模式下，擁有最多單點錯誤的風險，而傳統的 HA 模式又似乎已趨近完美，不但不需要額外的 NAT 主機，而且也避免了所有單點錯誤的情形，但唯一美中不足的是，他需要大量的公開 IP 供運算節點使用。相較之下，我們的 PHA 模式就符合大部分的使用者需求。

圖 3.5顯示了從專案角度來看待虛擬機的網路架構。一般來說，擁有公開 IP 的虛擬機通常為整個專案 cluster 的進入點，也常常是專案裡負責對外提供服務的重要機器，

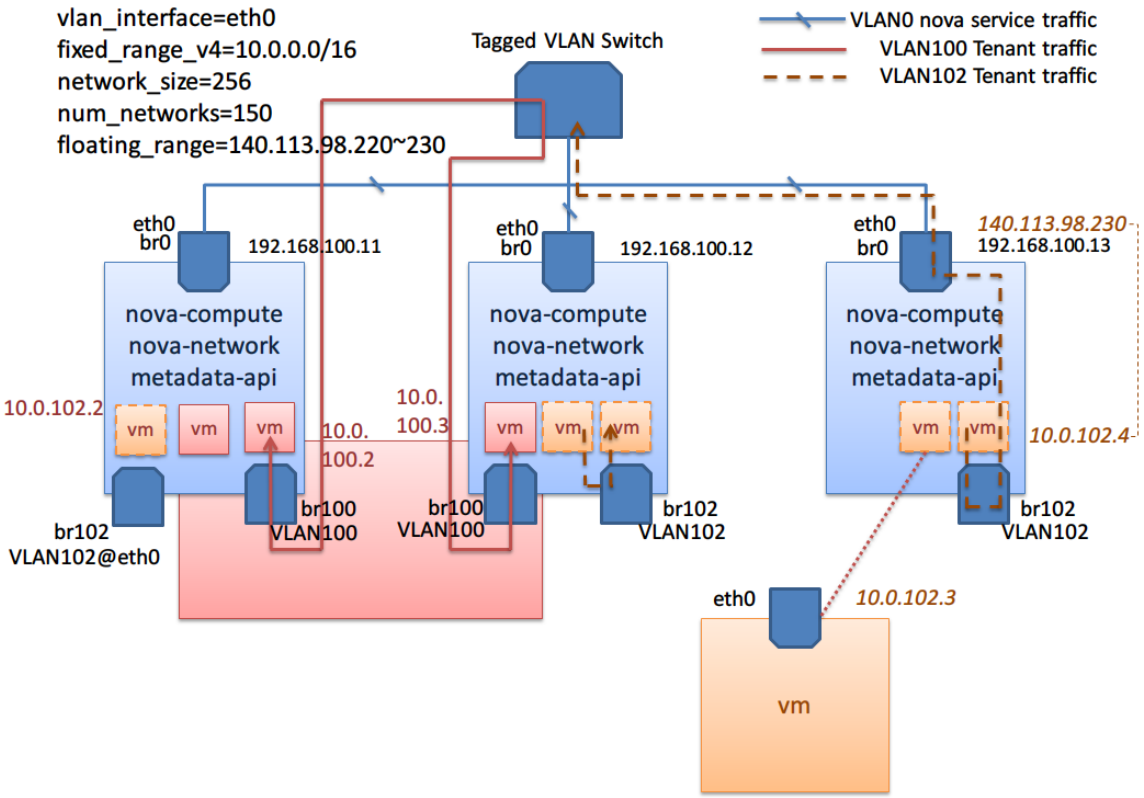

圖 3.4: 虛擬機網路互動關係圖 (Provider View)

在 PHA 模式下這種機器的流量可直接經由運算節點傳送到網路上的實體 Gateway，確保高度可用性；另一方面，由於虛擬機與宿主機的網路卡互相以橋接 (bridge) 模式相連，可視為在同一個實體網路下，所以同專案虛擬機之間的網路流量皆是經由實體交換器來轉送，也不會有單點錯誤的問題；所以在我們的 PHA 模式下，我們節省了大量的公開 IP，讓運算節點以及那些沒有公開 IP 的虛擬機一樣，在有對外連線需求時改用 NAT 主機，而這樣的需求通常是在更新系統套件，或是需要對外連線的時候，而這種情形相較之下是比較不重要，也不常發生的。

此外，由於 OpenStack 在做虛擬機動態遷移時，需要儘量使來源端與目的端的運算節點的相關設定保持一致，其中一點是必須讓運算節點上的 VNC 服務聆聽在 0.0.0.0(vncserver_listen=0.0.0.0)[8]，這樣做是為了避免虛擬機遷移之後，VNC 服務無法使用原本的 IP 而造成遷移失敗。可是這樣的設定在傳統 HA 網路模式上會造成 VNC 服務的暴露；反觀我們的 PHA 模式的網路架構，卻可以很好的支援這個需求。因為運算節點上並沒有公開 IP，外部網路無法直接存取 VNC 服務，至於那些擁有浮動 IP 的運算節點，所有存取浮動 IP 的連線皆會被導入虛擬機上，也無法存取到 VNC 服務，進而提供了比傳統 HA 網路模式更好的安全性。

因此我們使用這種 PHA 模式，必要的時候也可以與傳統 HA 模式混合使用，以符合

		SPOF	Another NAT Server	Public IP per Host	Service Exposure
傳統無 HA	fixed IP inner flow	! (dhcp flow)	V (Network Host)		
	fixed IP outbound flow	V			
	w/ floating IP	V			
傳統 HA	fixed IP inner flow			V	V
	fixed IP outbound flow				
	w/ floating IP				
PHA	fixed IP inner flow		V		
	fixed IP outbound flow	V			
	w/ floating IP				

Table 3.1: 傳統、傳統 HA 模式以及 PHA 模式的網路架構比較表

實際需求，具體做法會在 4.2.4小節－ Compute Worker 安裝步驟之後介紹。

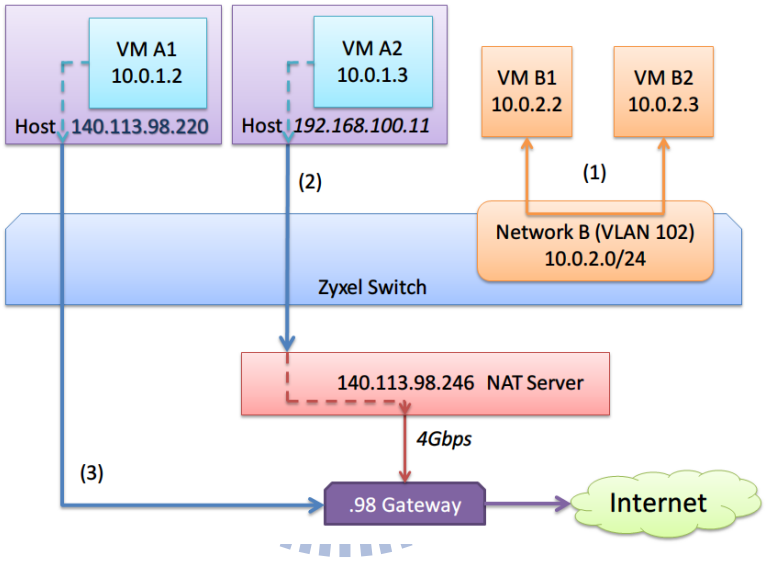

圖 3.5: P-HA 網路模式下的三種使用情況：(1) 同專案的虛擬機之間的網路流量。(2) 沒有公開 IP 的虛擬機往 Internet 的流量需經過 NAT。(3) 有公開 IP 的虛擬機可直接與 Gateway 溝通。

第四章　CStack 系統安裝步驟

由於整合過的 CStack 系統頗具規模，因此我們將其拆成幾個章節來做介紹。

在本章先介紹 Ceph 的基本安裝，接著會在 4.2節介紹 OpenStack 安裝步驟與 Ceph 的整合方法，這樣我們就可以利用 OpenStack 和 Ceph 部署一個基礎的 IaaS 服務了。最後到了 4.3節，我們會針對 nova.conf 裡幾個重要的設定來做說明，並針對我們新增的參數來解釋其用法和意義。

4.1　Ceph 安裝細節

Ceph 分散式檔案系統是整個 NCTU CStack 系統中最重要的部分，因此我們需要先安裝好才能進行後續的步驟。

1. 在所有機器上面匯入套件金鑰，並且加入 Ceph 官方套件庫

```
# wget -q -O- https://raw.github.com/ceph/ceph/master/keys/release.asc \
| sudo apt-key add -
# tee /etc/apt/sources.list.d/ceph.list <<EOF
deb http://ceph.com/debian/ precise main
deb-src http://ceph.com/debian/ precise main
EOF
```

2. 在所有機器上面更新套件庫，安裝 Ceph。

```
# aptitude update; aptitude install ceph
```

3. Ceph 會使用 hostname 來溝通，為了降低複雜度，我們不需要架設自己的 DNS，而是讓所有機器使用同一份 /etc/hosts [附錄 A.1]。

```
mon1# vim /etc/hosts
mon1# scp /etc/hosts slave*:/etc/hosts
```

4. Ceph 的某些管理指令 (如 mkcephfs) 會從管理節點 (monitor node) 直接使用 ssh 連到 Cluster 內的其他的節點來執行命令，所以我們使用 ssh-keygen 來替管理節點上的 root 產生連線所需的金鑰，並使用 ssh-copy-id 來把金鑰複製到其他 slave 機器上。

```
mon1# ssh-keygen -d
mon1# ssh-copy-id root@slave*
mon1# ssh slave* uname    // 測試ssh連線是否不需要輸入密碼
```

5. Ceph 使用 ceph.conf 這個設定檔來定義整個系統的架構、參數等，撰寫符合所需的設定檔 [附錄 A.2]，並同步到所有的 Ceph node 上。

```
mon1# vim /etc/ceph/ceph.conf
mon1# scp /etc/ceph/ceph.conf slave*:/etc/ceph/ceph.conf
```

6. 至此，我們可以來格式化 Ceph 了，ceph.conf 上面指定的 btrf devs 將會被重新格式化，請注意填寫是否正確。

```
mon1# mkcephfs -v -a -c /etc/ceph/ceph.conf --mkbtrfs
```

7. 若過程中沒有任何錯誤，則 ceph.conf 上 keyring 所指定的位置應該會產生出正確的 keyring 檔。

```
mon1# ceph auth list    // 列出所有的keyrings
```

8. 將 Ceph 的所有元件啟動，並確認 Ceph 的狀態。

```
mon1# /etc/init.d/ceph -a start
mon1# ceph health    // 顯示簡單的健康狀態
mon1# ceph -s    // 顯示詳細的狀態
mon1# ceph -w    // 持續觀看
```

9. Ceph 有個特點就是可以根據實體機器的部署配置，決定 PG replica 的儲存方式 (參考 2.3小節)；我們先編寫 crush.nctu.txt [附錄 A.3]，然後編譯成 Ceph 看得懂的 CRUSH map，套用完成後可使用 ceph osd tree 確認結果。

```
// 編譯crush map
mon1# crushtool -c /etc/ceph/crush/crush.nctu.txt -o /etc/ceph/crush/crush.nctu
// 令crush map生效
mon1# ceph osd setcrushmap -i /etc/ceph/crush/crush.nctu
// 查看套用新的crush map後的結果
mon1# ceph osd tree
```

10. 最後我們可以依照需求，設定各個儲存池 (pool) 所需的 replica 數量。

```
mon1# ceph osd pool set <pool> size 3
```

如此 Ceph 的安裝到這邊告一段落。

4.1.1 Ceph 擴展方法

由於 Ceph 擁有高度擴展的能力，因此我們可以在 Ceph 儲存空間不夠時，利用增加 Object Storage Device(OSD) 的方式來彈性增加儲存空間，或是依照其他需求來動態增加 Monitor 節點以及 Metadata server 節點的數量。具體做法可以參考官方文件 [3]，或是使

用附錄 A.6 - mkosd 這個 script 來快速增加一個 OSD 節點。

4.2 OpenStack 安裝細節

OpenStack 本身是一個 cloud manager，除了負責管理虛擬機的建立與刪除，也要管理虛擬機開機所需的映像檔 (image)，以及額外掛載到虛擬機上的虛擬磁碟 (volume)，服務雖然多樣且完整，但是不同的服務各自使用不同的資料儲存方式，會造成管理上的複雜化，以及資源使用沒有效率。

在這一節，我們將參考這個網站 [6] 使用 Ubuntu 12.04 Precise 的官方套件來安裝 OpenStack Essex，每一小節後會有一段介紹該服務如何與 Ceph 結合 [21]，讓所有需要儲存空間的服務後端都改成使用 Ceph-這一個分散式，可靠的物件儲存系統。

4.2.1 Controller 安裝步驟

我們安裝的 Controller 提供的服務包含了 Keystone 認證系統，Dashboard 介面，和 Nova APIs，scheduler 等，因此這部分的介紹主要包含軟體安裝，Keystone 設定，Nova 設定。

1. 這邊安裝 Nova Controller 幾個最重要的元件：各種 API daemon，scheduler，vnc 元件 (因為 Nova-vncproxy 僅有很少的瀏覽器支援，Essex 版本後改用 novnc[17])，和一些會用到的 python 函式庫。

```
# aptitude install nova-api-ec2 nova-api-os-compute nova-api-os-volume \
    nova-cert nova-common nova-console nova-consoleauth nova-ajax-console-proxy \
    nova-scheduler python-django-nova python-nova python-nova-adminclient \
    python-novaclient nova-objectstore python-novnc novnc python-numpy nova-doc
```

2. Essex 版本後，一些重要的系統操作 (如設定 iptables，mount 虛擬機的 image 等) 預設使用一個 wrap 程式來代為呼叫，確認 nova-rootwrap 指令有無加入 sudoers，並指定 NOPASSWD(不需輸入密碼)。

```
# visudo
```

```
// 增加下面這行
nova     ALL=(root)       NOPASSWD: /sbin/vgs,/usr/bin/nova-rootwrap,/sbin/iptables-
restore,/sbin/iptables-save
```

3. 安裝 OpenStack Dashboard (code name: horizon)。

```
# aptitude install libapache2-mod-wsgi openstack-dashboard openstackx \
    openstack-dashboard-ubuntu-theme python-django-openstack python-openstack-common
```

4. 安裝 OpenStack Keystone 認證系統。

```
# aptitude install keystone python-keystone curl python-mysqldb python-dateutil
```

5. 習慣使用 Amazon ec2 服務的人可安裝 Eucalyptus tool(另一套管理系統，有安裝 nova-api-ec2 就可使用) 來控制你的 OpenStack，另外我們還需要 AMQP Server，Database。

```
# aptitude install euca2ools rabbitmq-server mysql-server
```

6. 設定 MySQL 要 listen 的 address，以及建立 nova、Glance 服務會用到的資料表

```
# sed -i 's/127.0.0.1/192.168.100.1/g' /etc/mysql/my.cnf
# mysql -uroot -p
```

```
# 建立nova資料庫
CREATE DATABASE nova;
# 設定nova資料庫的使用者帳號及密碼
GRANT ALL PRIVILEGES ON nova.* TO 'root'@'%' IDENTIFIED BY '<<yournovadbpw>>';
# 建立glance資料庫
CREATE DATABASE glance;
# 設定glance資料庫的使用者帳號及密碼
GRANT ALL PRIVILEGES ON glance.* TO 'root'@'%' IDENTIFIED BY '<<yourglancedbpw>>';
# 讓改變生效
FLUSH PRIVILEGES;
quit;
```

7. OpenStack Essex 使用 Keystone 來負責所有的帳號 (User)、專案 (Tenant/Project)、身分 (Role) 的管理，接著我們來修改 Keystone 設定。

```
# vim /etc/keystone/keystone.conf
```

```
[DEFAULT]
bind_host = 192.168.100.1    // Keystone服務listen在private network上
admin_token = <<yourkeystonetoken>>  // 管理Keystone上的tenants, users, role的最終密碼
...
[catalog]
// 註解下面這行
#driver = keystone.catalog.backends.sql.Catalog
// 預設catalog存放在database上，在大規模佈署OpenStack時(endpoint較多)，這樣做較有效率。
// 但我們使用templated catalog file，簡單的修改檔案內容即可管理catalog。
driver = keystone.catalog.backends.templated.TemplatedCatalog
template_file = /etc/keystone/default_catalog.templates
```

8. 編輯 templated catalog file [附錄 B.1]。

   OpenStack 也使用 Keystone 來定義所有服務的總類 (catalog)，以及存取入口 (endpoint)；舉例來說，我們之後會在 image 服務 (Glance) 的設定檔上註明，使用 keystone 認證 (即 Glance 將不會維護任何的帳號密碼和權限)，並且在 Keystone catalog 註冊一個 image 的服務存取入口，因此任何想要使用 OpenStack 的 image 服務的人 (或其他服務)，都可以先跟 Keystone 做認證取得授權後，再向 catalog 查

詢 Glacne 的存取入口為何，進而正確使用該服務。

OpenStack 的服務使用 ReST 介面來做溝通，任何一個服務使用者 (可能是人或另一個服務) 需要知道每個服務的位址 (URL) 和連接端口 (port)，才能正確地使用該服務。catalog 即定義了所有服務的 endpoint(存取入口)。

為了安全性，除了要開放給 User 使用的 API 之外，所有服務都將設定成 listen 在 192.168 網段下，因此我們在這邊必須 endpoint 的描述從 localhost 改成 controller(或 192.168.100.1)，若有些服務如 image、volume 不是由 controller 提供，也需改成相對應的 hostname 或 IP，這樣之後才能正常連線。

另外加入 (註冊)object-store 的 endpoint，讓 Dashboard 提供" 物件儲存" 的功能。

```
# vim /etc/keystone/default_catalog.templates
```

```
...
// 加入object-store endpoint。
catalog.RegionOne.object-store.publicURL = http://s3.nctu.edu.tw/auth
catalog.RegionOne.object-store.adminURL = http://s3.nctu.edu.tw/auth
catalog.RegionOne.object-store.internalURL = http://s3.nctu.edu.tw/auth
catalog.RegionOne.object-store.name = Swift Service
```

9. 接下來初始化 Keystone 的 database，並將權限設定好。

```
# keystone-manage db_sync
# chown keystone:keystone /var/lib/keystone/keystone.db
# chmod 640 /var/lib/keystone/keystone.db
# /etc/init.d/keystone restart
```

10. 到目前為止，已經安裝好 Keystone 服務，我們可以設定環境變數來簡化執行 Keystone client 所需要的參數，確認 keystone 服務能否正常使用。

```
# vim .keystonerc
```

```
// 設定好Keystone client會使用的兩個參數
export SERVICE_TOKEN=<<yourkeystonetoken>>
export SERVICE_ENDPOINT=http://controller:35357/v2.0/
```

```
# . .keystonerc        // bash shell使用"."讀入環境變數
# keystone user-list
```

```
\\ 因為尚未建立任何user，正常的話會顯示空的user list
+----+---------+-------+------+
| id | enabled | email | name |
+----+---------+-------+------+
+----+---------+-------+------+
```

11. 接著可以來新增專案 (Tenant)，使用者 (User) 了。這邊我們可以使用 devstack 所提供的 script[13] 來快速建立幾個基本的專案跟使用者。執行 script 前可針對想產生的帳號密碼稍作修改，該 script 的註解寫得相當清楚，建議可以全部看過。

```
# . .keystonerc        // 讀入環境變數
# vim keystone_data.sh
```

```
// 預設這個script所建立的user皆使用相同的密碼($ADMIN_PASSWORD)，
// 這邊我們稍作修改各自使用不同的密碼。
...
ADMIN_PASSWORD=${ADMIN_PASSWORD:-<<youradminpw>>}    // 自訂user: admin的密碼
DEMO_PASSWORD=${DEMO_PASSWORD:-<<yourdemopw>>}       // 增加這行定義user: demo的密碼
// service user(eg. nova, glance)也使用不同的密碼
SERVICE_PASSWORD=${SERVICE_PASSWORD:-<<yourservicepw>>}
...
DEMO_USER=$(get_id keystone user-create --name=demo \
                    --pass="$DEMO_PASSWORD" \     // 使用剛剛定義的$DEMO_PASSWORD
```

```
# bash keystone_data.sh     // 執行結果沒有顯示任何東西即為建立成功
# keystone user-list        // 確認我們剛剛建立的user
```

```
+----------------------------------+---------+-----------------------+--------+
|                id                | enabled |         email         |  name  |
+----------------------------------+---------+-----------------------+--------+
| 4146a6ade427467380cb0c8120604373 | True    | chuanyu@cs.nctu.edu.tw | admin  |
| 4f81b14bee8d4637af31fd606145c452 | True    | chuanyu@cs.nctu.edu.tw | glance |
| a37a9074a70d4b78b756dbed11359105 | True    | chuanyu@cs.nctu.edu.tw | nova   |
| a9d4eb6a28414a0f9b3e65e3a87a14b9 | True    | chuanyu@cs.nctu.edu.tw | demo   |
+----------------------------------+---------+-----------------------+--------+
```

12. 可使用一個小 script TUG.sh 顯示 Tenants、Users、Roles 的全部對應關係 [附錄 B.2]

```
# bash TUG.sh
```

```
Tenant: fc280a91aa7a400fad3fb92f4e203a82 ================= service
        User: nova => a37a9074a70d4b78b756dbed11359105
                Role: admin => dda74ea9cbdd4b13b23dbf61fc2647b2
        User: glance => 4f81b14bee8d4637af31fd606145c452
                Role: admin => dda74ea9cbdd4b13b23dbf61fc2647b2
Tenant: f433c5e046ec43ec9cdfa035a4bbd56c ================= admin
        User: admin => 4146a6ade427467380cb0c8120604373
                Role: admin => dda74ea9cbdd4b13b23dbf61fc2647b2
                Role: KeystoneServiceAdmin => c3bdc07586994e04aa38a9dff1057054
                Role: KeystoneAdmin => 68fb24354d374422a0fccbc474a53bd7
```

13. Keystone 的設定到此結束，接著我們需要處理 nova 的設定，但因 nova.conf 為 OpenStack 裡最重要的設定檔之一，請參考 4.3 節來做設定。

```
# vim /etc/nova/nova.conf
```

14. api-paste.ini 定義了 nova 提供的各種 api(eg. EC2, Openstack, Metadata...) 要如何運作，這邊只需針對 nova 如何向 Keystone 溝通的部分做設定。

```
# vim /etc/nova/api-paste.ini
```

```
...
[filter:authtoken]
// 須注意我們的Keystone是listen在192.168網段，這裡需改成controller，
// 不能使用原本的localhost。
service_host = controller
auth_host = controller
auth_uri = http://controller:5000/
// 這邊我們使用剛剛用script建立的service user(nova)來跟keystone做溝通即可。
admin_tenant_name = service
admin_user = nova
admin_password = <<yourservicepw>>
```

15. 接著修改 Apache2 以及 Dashboard 的設定檔。

```
# vim /etc/apache2/conf.d/openstack-dashboard.conf
```

```
#  Allow from all
Allow from 140.113.0.0/16     // 限定校內IP讀取。
```

```
# vim /etc/openstack-dashboard/local_settings.py
```

```
ENABLE_JUJU_PANEL = False     // 關掉沒用到的JUJU頁面。
...
OPENSTACK_HOST = "192.168.100.1"    // 指定Keystone所listen的位址。
```

16. 如此 nova 的設定也告一段落，我們可以將 nova db 初始化，然後重開 Controller 上所有的服務，接著下一些指令測試 nova 的功能，照慣例還是需要先設定環境變數(或是執行時指定一堆參數)。

```
# nova-manage db sync
# for a in keystone nova-api-ec2 nova-api-os-compute nova-api-os-volume \
    nova-objectstore nova-scheduler nova-novncproxy nova-cert nova-console \
    nova-consoleauth apache2; \
    do sudo service "$a" restart; done
# /etc/init.d/novnc restart
# nova --os_username admin --os_password <<youradminpw>> --os_tenant_name admin \
    --os_auth_url http://controller:35357/v2.0/ list
# vim .novarc
```

```
export NOVA_URL="http://controller:35357/v2.0/"
export NOVA_VERSION="1.1"
export NOVA_API_KEY="<<yourservicepw>>"
export NOVA_USERNAME="nova"
export NOVA_PROJECT_ID="service"
```

```
# . .novarc
# nova list; nova image-list; nova volume-list
```

```
+----+------+--------+----------+
| ID | Name | Status | Networks |
+----+------+--------+----------+
+----+------+--------+----------+
```

```
+--------------------------+------------------+--------+--------+
|            ID            |        Name      | Status | Server |
+--------------------------+------------------+--------+--------+
+--------------------------+------------------+--------+--------+
+----+-----------+--------------+------+-------------+-------------+
| ID |   Status  | Display Name | Size | Volume Type | Attached to |
+----+-----------+--------------+------+-------------+-------------+
+----+-----------+--------------+------+-------------+-------------+
```

限制專案可以使用的運算節點

有時候因為安全性的需求，或是實體機器的運用規劃，我們會希望將實體的運算節點做分群，然後限制某些專案的虛擬機只能運行在特定的運算節點上。Essex 版的 OpenStack 可以利用兩種方式來達到這樣的需求，一個是利用 Zone 的切割，另一個是使用 Scheduler。因為 Multiple Zone 的部署通常是使用在超大規模 (如跨地區) 的環境，因此部署成本也相當昂貴 [20]，所以我們利用原有的 Filter Scheduler 搭配自行撰寫的 mapping filter 來達成這個功能。

Scheduler 在 OpenStack 的角色是負責決定新的虛擬機會被建立在哪一台運算節點上，而 Filter Scheduler 則是先利用 filters 來過濾運算節點，接著利用 weighting 來決定這些可供使用的運算節點候選人的偏好順序 [9]。雖然 OpenStack 提供了許多內建的 filters，但是並沒有辦法達到我們的需求，因此我們額外撰寫一個 project-host-filter 來處理專案與實體運算節點的對應關係 (請參考附錄 B.6)。

project-host-filter 維護了一個 Map 資料結構如下：

```
PHMap = {
    # when no project matches, choose from following hosts
    'Default': {
        'compute-1',
        'compute-2',
        'compute-3',
    },
    # specify project(tenant) id and dedicated hosts
    '2bd91f3a4df64427815f721cd663267a': {   # pacas
        'compute-1',
    },
}
```

利用自訂這個資料結構，我們就可以過濾出符合限制的運算節點有哪些。使用的方法也很簡單，只需在 nova.conf 加入欲使用的 filter 即可：

```
# 將預設的filters設為可用
scheduler_available_filters=nova.scheduler.filters.standard_filters
# 將自行撰寫的ProjectHostFilter設為可用
scheduler_available_filters=nova.scheduler.filters.project_host_filter.ProjectHostFilter
# 列出想要使用的filters
```

```
scheduler_default_filters=ProjectHostFilter,RamFilter
```

另外，scheduler 其實也負責調度新建立的磁碟是由哪一台 nova-volume 來負責提供，可是官方的 Filter Scheduler(nova.scheduler.filter_scheduler.FilterScheduler) 並沒有實作調度磁碟的函式 schedule_create_volume，這樣會導致建立磁碟時狀態一直顯示為 creating，無法正常建立磁碟。因此我們建立新的 Scheduler 繼承自原本的 Filter Scheduler，並實作磁碟調度函式 (請參考 B.5)，如此就可以使用 Filter Scheduler 的功能，並確保其他功能正常。使用方式只需在 nova.conf 加入一行即可；

```
scheduler_driver=nova.scheduler.filter_scheduler_new.FilterSchedulerNew
```

4.2.2 Glance 安裝步驟

OpenStack 建立虛擬機的時候，需要指定虛擬機的開機映像檔，映像檔可以是任意的作業系統安裝光碟，或是一個預先安裝好作業系統的虛擬硬碟，來讓虛擬機直接使用該硬碟開機。一些主流的 Linux 作業系統都有替這種需求額外提供各種格式的映像檔 [10]，我們也可以製作符合自己需求的映像檔上傳到 OpenStack 使用。除此之外，OpenStack 也讓使用者可以在任何時刻對虛擬機執行快照 (snapshot)，將虛擬機當時的硬碟內容完全複製起來，當成另一份映像檔供使用者使用[1]。

Glance 就是負責管理及儲存所有虛擬機的開機映像檔及快照的服務，我們將在這一小節來說明他的安裝步驟。

1. 安裝 Glance 套件並設定 glance-api.conf 使其使用 Keystone 認證 [12]。

```
# aptitude install glance glance-api glance-client glance-common glance-registry
# aptitude install python-keystone python-keystoneclient
# vim /etc/glance/glance-api.conf
```

```
bind_host = 192.168.100.1        // listen在192.168網段
registry_host = 192.168.100.1
workers = 8
...
rabbit_host = controller
qpid_host = controller
...
[paste_deploy]
flavor = keystone          // 使用Keystone認證
```

2. 設定 glance-registry。

[1]OpenStack 裡的 Image 或 Snapshot 類似於 VMware 裡的 Template(樣板)，是可用來開啟另外的虛擬機，而不是用來回朔虛擬機狀態。

```
# vim glance-registry.conf
```

```
bind_host = 192.168.100.1          // listen在192.168網段
#sql_connection = sqlite:////var/lib/glance/glance.sqlite
sql_connection = mysql://root:<<yourglancedbpw>>@controller/glance     // 使用mysql
...
[paste_deploy]
flavor = keystone          // 使用Keystone認證
```

3. 設定 glance-api-paste.ini，glance-registry-paste.ini 這兩個檔案。這邊須注意目前官方的文件仍建議使用 glance_auth_token:filter_factory，這是已過時的寫法，請使用 Ubuntu 內建的設定檔即可。

```
// 兩份檔案皆只需修改向Keystone認證的部分即可。
# vim glance-api-paste.ini
# vim glance-registry-paste.ini
```

```
service_host = controller     // 指定要認證的主機位置
auth_host = controller
auth_uri = http://controller:5000/
// 這邊我們使用剛剛利用Keystone建立的service user(glance)來跟keystone做溝通即可。
admin_tenant_name = service
admin_user = glance
admin_password = <<yourservicepw>>
```

4. 設定檔已修改完畢，我們可以初始化 glance 的資料庫。

```
# glance-manage version_control 0
# glance-manage db_sync
```

5. 重開 Glance 所有服務，然後一樣要設定好環境變數，就可以列出目前 Glance 上的映像檔，測試看看運作是否正確。Glance 的安裝到這邊告一段落，這邊須注意的是我們還沒設定好跟 Ceph 整合的部分，所以暫時還不能上傳映像檔。

```
# service glance-api restart && service glance-registry restart
# vim .glancerc
```

```
export OS_AUTH_URL=http://controller:5000/v2.0/

export SERVICE_ENDPOINT=http://controller:35357/v2.0
export OS_AUTH_STRATEGY=keystone

# glance rc
export OS_USERNAME=glance
export OS_PASSWORD=<<yourservicepw>>
export OS_TENANT_NAME=service
```

```
# . .glancerc
# glance index
```

```
ID            Name          Disk Format   Container Format  Size
------------- ------------- ------------- ----------------- -------
```

Glance with Ceph RBD

Glance 服務提供了很多方式存放虛擬機的映像檔，因為映像檔本身就可以看成是一個大的 object，所以除了可以設定要存放在本地磁碟上，也支援放到常見的 object storage service，如 Swift、Amazon S3[2]，以及，存放到 Ceph 上：

1. 要讓 Glance 使用 Ceph 當儲存後端需要先在 Ceph 端建立一個供 Glance 使用的帳號與儲存池 (pool)，以及將 Ceph 金鑰 (keyring) 複製回 Glance 主機供 Glance 程式使用。

```
// 到Ceph上建立要讓Glance存放映像檔的pool，名為images。
ceph-mon# rados mkpool images
// 建立Glance要使用的帳號(client.glance)與金鑰檔，設定帳號權限為只能存取images pool。
ceph-mon# ceph-authtool --create-keyring image.keyring
ceph-mon# ceph-authtool --gen-key --name client.glance image.keyring
ceph-mon# ceph auth add client.glance mon 'allow r' osd 'allow rwx pool=images' \
    -i image.keyring
// 將金鑰檔拷貝回Glance主機，並設定權限。
ceph-mon# scp image.keyring root@glance:/etc/glance/image.keyring
glance# chown glance:glance image.keyring
glance# chmod 600 image.keyring
// 定義client.glance這個使用者的金鑰位置。
glance# vim /etc/ceph/ceph.conf
```

```
[client.glance]
    keyring = /etc/glance/image.keyring
```

2. 接著設定 Glance 使用我們剛剛建立的帳號來存取 Ceph RBD 上的 images 儲存池 (pool)。

```
# vim /etc/glance/glance-api.conf
```

```
default_store = rbd
...
rbd_store_ceph_conf = /etc/ceph/ceph.conf  // 後端使用Ceph RBD
rbd_store_user = glance
rbd_store_pool = images
rbd_store_chunk_size = 8
```

3. 如此就完成了 Glance 與 Ceph 的整合，重開服務後就可以試著上傳一個映像檔了。

```
glance# service glance-api restart; service glance-registry restart
// 載入執行glance指令所需的環境設定
glance# . .glancerc
glance# glance index
// 下載Ubuntu 12.04的雲端版本映像檔。
glance# wget http://cloud-images.ubuntu.com/precise/current/precise-server-cloudimg-
```

[2]可以參考/etc/glance/glance-api.conf。

```
amd64-disk1.img
// 上傳一個公開的映像檔。
glance# glance add name="Ubuntu 12.04 cloud\" is_public=true container_format=ovf \
    disk_format=qcow2 < ./precise-server-cloudimg-amd64-disk1.img
glance# glance index
```

```
ID                Name                Disk Format   Container Format  Size
----------------- ------------------- ------------- ----------------- ---------
df23d532-9143-234 Ubuntu 12.04 cloud qcow2         ovf               230752256
```

4.2.3 Nova-volume 安裝步驟

Nova-volume 服務提供了虛擬機額外的邏輯磁碟，除了可以在虛擬機開機後動態的增減虛擬機上的磁碟，Essex 版的 OpenStack 也支援使用 volume 來讓虛擬機開機，這個服務因為相當重要，因此到了 OpenStack 下一個版本 Folsom 將獨立成一個核心專案－Cinder。我們來介紹他的安裝步驟，以及如何將它處理的 volume 存放到 Ceph 上。

1. 先安裝好 nova-volume 套件，設定檔則修改向 Keystone 認證的部分即可。

```
# aptitude install nova-volume
# vim /etc/nova/api-paste.ini
```

```
...
[filter:authtoken]service_host = controller
auth_host = controller
auth_uri = http://controller:5000/
admin_tenant_name = service
admin_user = nova
admin_password = <<yourservicepw>>
```

Nova-volume with Ceph RBD

Nova-volume 預設使用 iSCSI 協定和名為 nova-volumes 的 LVM volume group 來儲存虛擬磁碟 (volume) 的資料[3]，但這樣的問題是容易遇到 iSCSI 伺服器的效能瓶頸，而且得另外維護 iSCSI 伺服器，增加管理上的複雜度。

因為虛擬磁碟與虛擬機的開機映像檔性質類似，皆是一份巨大的檔案 (或者看成一個物件)，因此我們希望統一使用 Ceph RBD 來當作 volume 的儲存後端，以下說明設定步驟：

1. 在 Ceph 上建立一個名為 volume 的儲存池 (pool) 給 Nova-volume 來存放所有的資料，並且建立一個使用者 (client.volume) 權限設定成只能存取 volume pool，再把產生的金鑰 (keyring) 複製回運行 Nova-volume 的機器。

[3]在 nova.conf 裡預設 volume_driver=nova.volume.driver.ISCSIDriver，volume_group=nova-volumes。

```
ceph-mon# rados mkpool volume
// 建立帳號及設定權限。
ceph-mon# ceph-authtool --create-keyring volume.keyring
ceph-mon# ceph-authtool --gen-key --name client.volume  volume.keyring
ceph-mon# ceph auth add client.volume mon 'allow r' osd 'allow rwx pool=volume' \
    -i volume.keyring
// 複製回nova-volume主機。
ceph-mon# scp volume.keyring root@volume:/etc/nova/volume.keyring
// 回到nova-volume主機，修改金鑰權限。
volume# chmod 600 /etc/nova/volume.keyring
volume# chown nova:nova /etc/nova/volume.keyring
```

2. 參考 4.1 節加入 Ceph 官方套件庫。
3. 安裝 ceph-common 套件，並建立 ceph.conf 設定檔 [附錄 A.2] 指明 Nova-volume 所使用的 Ceph client 以及金鑰檔位置。

```
// 安裝操作RADOS所需指令
volume# aptitude install ceph-common
// 因為rados指令預設會讀取這個位置的設定檔，所以我們在這邊建立。
volume# vim /etc/ceph/ceph.conf
```

```
// 指定nova-volume所使用的帳號(client.volume)及金鑰位置。
[client.volume]
    keyring = /etc/nova/volume.keyring
```

```
// 測試可否正常執行rados指令。
volume# sudo -u nova rados --id volume -p volume ls
```

4. 接著要稍微修改 nova-volume 的啟動程序，讓 rados 用我們剛剛建立的 client.volume 身分來存取 Ceph[21]。

```
volume# vim /etc/init/nova-volume.conf
```

```
\\ 添加 env CEPH_ARGS=\"--id volume\"
exec su -s /bin/sh -c "exec env CEPH_ARGS=\"--id volume\" nova-volume --flagfile=/
etc/nova/nova.conf" nova
```

5. 修改/etc/nova/nova.conf，讓 nova-volume 使用 Ceph RBD(RADOS) 來存放 volume。

```
# nova-volume use Ceph RBD backend
volume_driver=nova.volume.driver.RBDDriver
rbd_pool=volume
rbd_user=volume
```

6. 接著重啟 nova-volume 就完成了，可以試著建立一個 1G 大小的 volume，並到 Ceph 上看看結果。

```
volume# service nova-volume restart
// 載入執行nova指令時所需的環境設定
volume# . .novarc
volume# nova volume-create --display_name=test-1G-vol 1
```

```
volume# nova volume-list
```

```
+----+-----------+--------------+------+-------------+-------------+
| ID |   Status  | Display Name | Size | Volume Type | Attached to |
+----+-----------+--------------+------+-------------+-------------+
| 1  | available | test-1G-vol  | 1    | None        |             |
+----+-----------+--------------+------+-------------+-------------+
```

```
// 確認volume pool有無剛剛建立的1G volume
ceph-mon# rbd --pool volume ls
```

```
volume-00000001
```

```
ceph-mon# rbd --pool volume info volume-00000001
```

```
rbd image 'volume-00000001':
        size 1024 MB in 256 objects
        order 22 (4096 KB objects)
        block_name_prefix: rb.0.0
        parent:  (pool -1)
```

4.2.4 Compute Workers 安裝步驟

有安裝 nova-compute 服務的機器稱之為運算節點 (compute worker) 負責最重要的任務－開啟虛擬機。在設定 compute worker 時，除了要指定希望使用的 hypervisor，另外我們也使用了 PHA 網路模式，讓每台運算節點也負責處理虛擬機的網路活動。

第一段，我們將介紹在每個 compute worker 上都安裝 nova-network 以及 nova-api-metadata，讓所有虛擬機都可以直接靠它所在的宿主機與外界溝通，不用只靠一個中央的 nova-network 主機來轉送網路封包，避免 single point of failure(如圖 3.4)；另外由於 Essex 版本的 OpenStack 在同一個 Zone 下僅能使用相同的 hypervisor，因此我們統一使用 KVM(Kernel-based Virtual Machine) 來運行虛擬機。

第二段介紹如何讓虛擬機開機後產生的資料存放到共享儲存 (shared storage)，讓虛擬機擁有動態遷移 (live migration) 的能力。

第三段我們提出了一個解決方案 (PHA 模式)，修改部分 Nova-network 的程式碼，讓我們的 compute workers 不用預先綁定公開 IP，在大量擴展運算節點時減少了 IP 的使用，也增加了安全性。

1. 安裝 compute node 所需套件，除了剛剛說明所需的套件之外，需另外安裝一電源管理程式讓 libvirt 使用 [21]。

```
# aptitude install nova-compute nova-compute-kvm
# aptitude install nova-network nova-api-metadata python-keystone
# aptitude install pm-utils
```

2. 一樣要確認 nova-rootwrap 有無加進 sudoers 裡。

```
# visudo
```

```
nova ALL=(root) NOPASSWD: /sbin/vgs,/usr/bin/nova-rootwrap,/sbin/iptables-restore,/
sbin/iptables-save
```

3. 確認 api-paste 設定檔，裡與 Keystone 溝通的部分。

```
# vim /etc/nova/api-paste.ini
```

```
...
[filter:authtoken]service_host = controller
auth_host = controller
auth_uri = http://controller:5000/
admin_tenant_name = service
admin_user = nova
admin_password = <<yourservicepw>>
```

4. 我們的 hypervisor 使用 KVM，所以要告訴 nova。

```
# vim /etc/nova/nova-compute.conf
```

```
--libvirt_type=kvm
```

5. nova.conf 設定檔說明了虛擬機的網路環境 (Flat DHCP, VLAN 等)，VNC（Virtual Network Computing）的運作方式，與 compute worker 息息相關，因此我們將會在之後 4.3節專門介紹。

```
# vim /etc/nova/nova.conf
```

6. 因為 nova 使用自己的 dnsmasq 來替虛擬機指派 IP，我們可以藉由設定 dhcp-option 來告訴虛擬機更多網路設定資訊。

```
// option=6為設定domain name server。
# echo "dhcp-option=6,140.113.235.107,8.8.8.8" | sudo tee /etc/nova/nova-dnsmasq.conf
# sudo chown nova:nova /etc/nova/nova-dnsmasq.conf
```

7. 因為我們的虛擬機網路使用 VLAN 模式，所以需要安裝額外的套件，並開啟 Kernel 的 802.1q module，讓從網卡出去的封包帶有 VLAN Tag，同時也要確定 Kernel 有開啟封包轉送的功能，這樣 computer worker 的安裝即告一段落。

```
# aptitude install vlan
# echo "8021q" | sudo tee -a /etc/modules          // 開機自動載入 802.1q 模組
# modprobe 8021q
# echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward  // 開啟Kernel封包轉送功能
```

Compute worker with Ceph FS

預設情形下 nova-compute 在開啟虛擬機時，都是先將使用者指定的映像檔從 Glance 下載到 $instances_path/_base[4]，並利用該映像檔當作快取，建立一個 copy on write 模式

[4]$instances_path 就是/var/lib/nova/instances。

的 qcow2 差異檔放在 $instances_path/instance-< 虛擬機編號 > 下，然後利用這個差異映像檔來開機。如此一來，一台 compute worker 上的同一個映像檔只會有一份快取，不管開了幾台虛擬機所增加的儲存空間也僅是差異部分，可以大量節省磁碟使用量。

可是這樣將虛擬機的資料存放在本機磁碟上，一旦宿主機需要離線維修時，就不能使用動態遷移 (migration) 到其他 compute worker，虛擬機會被迫關機或刪除，若是虛擬機上執行著一些很重要的服務，長時間停止服務將會造成很大的代價。因此，我們使用本機硬碟存放剛剛所說的映像檔快取，利用本機硬碟的高速特性讓虛擬機運作更加流暢；利用 Ceph 提供的遠端檔案系統 Ceph FS 來當作共享儲存區 (shared storage)，存放映像檔的差異部分，增加資料安全性與符合虛擬機的動態遷移條件。

1. 參考 4.1 節加入 Ceph 官方套件庫。
2. 安裝 ceph-fuse[5]。

```
compute-1# aptitude install ceph-fuse
```

3. 使用 ceph-fuse 下掛載指令時不需要指定 ceph monitor 的位址，因為 ceph-fuse 會使用 Ceph 設定檔裡的 mon addr 的值，所以這邊我們和 controller 使用同一份設定檔，並增加 nova-compute 要使用的帳號金鑰位置。

```
compute-1# mkdir -p /etc/ceph
compute-1# scp controller:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
compute-1# chown nova:nova /etc/ceph/ceph.conf
compute-1# vim /etc/ceph/ceph.conf
```

```
[client.nova]
    keyring = /etc/nova/nova.keyring
```

4. 到 Ceph monitor 上建立 nova-compute 要使用的 ceph 帳號金鑰，指定好帳號權限，並複製回 compute worker 上。

```
ceph-mon# ceph-authtool --create-keyring nova.keyring
ceph-mon# ceph-authtool --gen-key --name client.nova nova.keyring
ceph-mon# ceph auth add client.nova mds 'allow' mon 'allow r' osd 'allow rw pool=data' \
    -i nova.keyring
ceph-mon# scp nova.keyring compute-1:/etc/nova/nova.keyring
compute-1# chown nova:nova /etc/nova/nova.keyring
```

5. 接下來可以試著掛載 Ceph FS，並在上面建立要給 nova-compute 使用的資料夾。

```
// 掛載Ceph FS到/mnt/ceph上。
compute-1# mkdir -p /mnt/ceph
compute-1# ceph-fuse /mnt/ceph --name=client.nova
// 建立之後instances存放的資料夾。
```

[5]因為開發中的 Ceph FS 還不像 RADOS 般穩定 [4]，所以我們使用 ceph-fuse(user mode filesystem) 取代 Kernel 內建的 client 來掛載 Ceph FS，雖然效能稍微不好，但是卻可以避免不可預期的 Kernel panic。

```
compute-1# mkdir -p /mnt/ceph/openstack/instances
compute-1# chown nova:nova /mnt/ceph/openstack/instances
compute-1# sync && umount /mnt/ceph
```

6. 再來格式化要當本地快取的磁碟分割區，先試著掛載起來設定權限。

```
compute-1# mkfs.xfs -f /dev/cciss/c0d0p4
// 先掛載CephFS(存放差異映像檔的位置)。
compute-1# ceph-fuse /var/lib/nova/instances --name=client.nova \
    -r /openstack/instances
// 再掛載要放快取的本地分割區(存放映像檔快取的位置)。
compute-1# mkdir -p /var/lib/nova/instances/_base
compute-1# mount -t xfs /dev/cciss/c0d0p4 /var/lib/nova/instances/_base
// 設定nova-compute可存取的權限。
compute-1# chown nova:nova /var/lib/nova/instances/_base
```

7. 我們可以修改 nova-compute 的 init script，不用每次都手動掛載本機分割區及 Ceph FS。

```
compute-1# vim /etc/init/nova-compute.conf
```

```
pre-start script
...
    # instance on ceph
    if ! cat /proc/mounts | grep \/var\/lib\/nova\/instances; then
        localpart=/dev/cciss/c0d0p4
        mountopt=rw,noatime,nodiratime
        useshared=true
        if $useshared; then
            # use shared storage for migration
            ceph-fuse /var/lib/nova/instances --name=client.nova \
                    -r /openstack/instances
            mkdir -p /var/lib/nova/instances/_base
            mount -t xfs -o $mountopt $localpart /var/lib/nova/instances/_base
        else
            # use local base
            mount -t xfs -o $mountopt $localpart /var/lib/nova/instances
        fi
        chown nova:nova /var/lib/nova/instances || true
        chown nova:nova /var/lib/nova/instances/_base || true
    fi
end script
```

另外這個段落介紹了怎麼設定讓 nova-compute 知道我們已經將 nova-volume 以及 Glance 的資料都存放到 Ceph 去了，所以當需要使用這兩個服務時，nova-compute 才能發出正確的命令。

1. 映像檔 (image) 的部分比較容易，因為與 Ceph 交涉的部分 Glance 已經幫忙處理好了，我們只要告知 nova.conf 我們要使用 Glance 來取得映像檔，以及 Glance API 的位置即可。

```
compute-1# vim /etc/nova/nova.conf
```

```
// Glance API的相關資訊。
glance_host=volume
glance_api_servers=volume:9292
image_service=nova.image.glance.GlanceImageService
```

2. 但是虛擬磁碟 (volume) 的部分比較複雜，而且這部分的設定如果錯誤，通常會導致建立好的虛擬磁碟無法正確掛載到虛擬機上。這是因為實際上是 libvirt 要負責掛載虛擬磁碟，所以 libvirt 必須知道怎麼去跟 Ceph RBD 取得資料，因此 nova-compute 必須透過 nova.conf 告訴 libvirt 一些細節。

```
compute-1# vim /etc/nova/nova.conf
```

```
// 指出存取虛擬磁碟所需要的Ceph帳號和儲存池名稱，
volume_driver=nova.volume.driver.RBDDriver
rbd_pool=volume
rbd_user=volume
```

3. 此外，libvirt 本來就支援了許多類型的虛擬機磁碟，除了各種映像檔，或是宿主機上的某個硬碟/分割區，也可以使用 NFS，或是 Ceph RBD，等這種遠端儲存空間的方式來當虛擬機磁碟，而這種連線方式通常又需要認證，因此 libvirt 就提供了一個管理指令－ virsh 來讓我們定義各種 secret(如 passwords, passphrases and encryption keys)。

   所以我們還需要在 libvirt 上定義一組 UUID 對應到 Ceph 的帳號密碼的關係，填寫進 nova.conf，之後 nova-compute 就可以告知 libvirt 使用這組 UUID 來向 Ceph 取得虛擬磁碟。

   (a) 撰寫一個 xml 格式的定義檔，描述一個 Ceph 的 secret，並且指定連線到 Ceph 時使用的名稱是 client.volume。

```
compute-1# vim secret.xml
```

```
<secret ephemeral='no' private='no'>
   <usage type='ceph'>
     <name>client.volume secret</name>
   </usage>
</secret>
```

   (b) 接著使用 virsh 來加入這個定義檔。

```
compute-1# virsh secret-define --file secret.xml
```

```
Secret 7f74c2c7-062c-d164-e97f-9224ac4571fe created
```

   (c) 如此就取得這個 secret 的 UUID(7f74c2c7-062c-d164-e97f-9224ac4571fe) [6]，接

[6]若希望每次都產生固定的 UUID，則可以在 secret.xml 裡指定 <uuid /> 標籤。

下來就是把這個定義檔 (Ceph 帳號:client.volume) 與他的金鑰互相關聯[7]。

```
compute1# virsh secret-set-value --secret 7f74c2c7-062c-d164-e97f-9224ac4571fe\
    --base64 AQjIdjwLnCGsKhAA99tX+4C4NoLJd89JchW4sg==
```

4. 這樣 libvirt 就有一組 UUID 對應到 Ceph 帳號密碼的關係，我們就可以把這個 UUID 填入 nova.conf。

```
compute-1# vim /etc/nova/nova.conf
```

```
// 指出我們要使用libvirt上的哪一個secret定義來做Ceph RBD的連線。
rbd_secret_uuid=7f74c2c7-062c-d164-e97f-9224ac4571fe    // 剛剛產生的UUID
```

到這邊為止，就可以試著建立虛擬機，並掛載一個虛擬磁碟上去看看功能是否正常了。

取消 Compute Worker 上固定的公開 IP

3.3節有提到在 NCTU CStack 環境下，我們的運算節點並沒有屬於自己的公開 IP，而是使用私有 IP 經過統一的 NAT 主機連線到網際網路，然後再搭配舊有的 High Availability(HA) 模式，在提高安全性與節省 IP 使用量的前提下，依然讓虛擬機擁有很好的連線保障。

改用這個模式除了要設定 NAT 之外，最大的問題是我們運算節點上的預設閘道 (Default Gateway) 必須設定成 NAT 主機，但是當運算節點動態的綁上公開 IP 之後，我們不希望來源是公開 IP 的封包也往 NAT 主機來傳送，造成 NAT 主機的負擔，以及更大的單點錯誤的問題。

要解決這個問題主要依靠 Linux Kernel 內建的簡易來源路由來達成，舉例如下：

```
// 當宿主機動態的綁定一公開IP之後，
# ip addr add 140.113.98.230/24 dev br0
// 修改nova-network，依照虛擬機的IP給定預設閘道(default gateway)。
# ip rule add from 10.0.5.6 table 200
# ip route add default via 140.113.98.254 dev br0 table 200
// 清空kernel快取，讓設定立即生效。
# ip route flush cache
// 同時發出ARP更新，告知大家我擁有了一個新的公開IP。
# arping -U 140.113.98.230 -A -I br0 -c 1
```

因此我們只要修改 nova-network，讓他在 (un)bind_floating_ip 時多做幾件事就好了，詳細的修改可以參考 [附錄 B.4]。這邊我們介紹安裝方法：

<secret ephemeral='no' private='no'>

<uuid>7f74c2c7-062c-d164-e97f-9224ac4571fe</uuid>

<usage type='ceph'>

[7]可以在 ceph monitor 上使用 $ ceph auth get client.volume 來取得某個帳號的金鑰檔。

```
compute-1# cd /usr/lib/python2.7/dist-packages/nova/network
compute-1# patch < ~/patch-compute-nopublicip
```

由於 source routing 是我們新增的部分，因此必須在重新執行 nova-network 時將舊的設定清掉，可以加入一行指令來完成：

```
# vim /etc/init/nova-network.conf
```

```
pre-start script
    ...
    # for PHA patch
    ip ru | grep 200 | awk '{print $3 }' | xargs -I@ sudo ip ru del from @ lookup 200
end script
```

Compute Workers 擴展方法

由於在我們的部署中，運算節點只有與 Ceph 共享儲存空間有相依性，因此在 Ceph 系統建置完成後，我們可以重複 4.2.4小節的方法複製我們的運算節點，以便運行更多的虛擬機。

4.3 nova.conf

fixed_ip VS. floating_ip

在設定 OpenStack 的虛擬機網路時，時常會看到 fixed_ip、fixed_range 或是 floating 的詞，其實 fixed ip 是指這組 IP 將伴隨著這個虛擬機一生，從虛擬機被建立分配到這個 IP，一直到虛擬機被終止銷毀，這個 IP 都只屬於這個虛擬機，甚至可以當作某些服務用來分辨虛擬機的依據 (如 Metadata API)，並且被分配到的虛擬機會經由 DHCP 得知自己被分配到這組虛擬 IP；反之，floating ip 則是一個隨時可以藉著使用者的需求而分配給不同的虛擬機的 IP，它是藉由動態綁定到虛擬機所屬的宿主機上，然後再利用 iptables 來將封包轉送到被分配到的虛擬機上，虛擬機內部是不需要知道的。

我們的系統將 fixed_range 設定成 10.0.0.0/16，floating 則是使用公開 IP 的網段。

Metadata API

在之前我們有提到要安裝 PHA 模式的運算節點，並提到了在這種模式下，每台運算節點都要安裝 nova-api-metadata 來提供服務。原因是因為在雲環境下有很多虛擬機，每一台虛擬機在開機時，都會需要一些基本資料來設定好作業系統，讓雲端服務使用者可以正常使用虛擬機，如 root 的 ssh 公鑰，主機名等，因此 OpenStack 提供了一個服務來讓虛擬機裡的作業系統有管道查詢這些資訊，一些常見的 Linux 發

行版如 (Ubuntu, Fedora) 都有提供雲端版本的映像檔，而這些映像檔除了較為瘦身之外，最重要的就是開機時會向 169.254.169.254 這個 IP 詢問這些基本資料 (metadata)[25]。所以我們必須在每台運算節點上安裝 nova-api，並在 nova.conf 裡設定 enabled_apis=metadata，只開啟 metadata api；如果是 Ubuntu 12.04 之後可以直接安裝 nova-api-metadata，也就是之前安裝流程建議的套件即可，並在 nova.conf 裡指明 metadata_host=$my_ip，如此一來 nova 就會增加一條轉送指令

```
iptables -A nova-network-PREROUTING -d 169.254.169.254/32 -p tcp -m tcp --dport 80 -j DNAT --to-destination <你的主機IP>:8775
```

讓虛擬機的查詢可以到達 nova-api-metadata 真正處在的 IP 和連接端口，而 nova-api-metadata 也會依照 request 的來源 IP 判斷，是哪一個虛擬機正在向我詢問資料，進而回答正確的 metadata。所以一旦你的虛擬機開機時很慢 (time out)，並伴隨著使用 ssh key 時無法登入，而且 log 出現以下訊息：

```
2012-08-23 12:21:21,694 - util.py[WARNING]: 'http://169.254.169.254/2009-04-04/meta-data/instance-id' failed [119/120s]: url error [timed out]
```

那就請檢查一下 metadata api 的設定，或是參考下一個子小節的 dmz_cidr。

dmz_cidr & routing_source_ip

DMZ 這個詞在網路架構裡常出現，主要就是希望在內部網路 (Private Service Network) 與外部網路 (Internet) 之間加入一個子網段當緩衝，避免外面的主機可以直接對內部重要伺服器進行連線或攻擊，但內部網路還是可以有限的與 DMZ 裡的機器做連線，有解除武裝區的意思。

那在 OpenStack 裡外部網路依然是指 Internet，而內部網路指的就是虛擬機所使用的網段 (fixed_range)。此外，我們知道除非我們設定了 auto_assign_floating_ip = true，不然建立虛擬機時預設只會給 fixed_ip(通常是 private IP)，所以外部網路是沒辦法直接訪問虛擬機的；那內部網路的機器要跟 Internet 連線，主要是透過 routing_source_ip 的設置，然後 nova-network 就會幫你加入以下規則：

```
iptables -A nova-network-snat -s <fixed_range> -j SNAT --to-source <routing_source_ip>
```

讓 iptables 幫你修改虛擬機送出來的封包，把裡面的來源 IP 改成 routing_source_ip，一般的情形下也就是設定成宿主機的 public IP，做類似 NAT 的動作，讓你的虛擬機可以正常的對外連線。

但有的時候我們不希望 VM 發出去的封包裡面的來源 IP 被修改，所以這裡的 dmz_cidr 所指定的就是那些你希望虛擬機可以直接使用內部網路 IP 存取的網段。例如上一個子小節所介紹的 Metadata API 服務，nova-api-metadata(192.168.100.11) 需要

知道來向他訪問的來源端 IP 是甚麼 (10.0.100.2)，才能回答相對應的 hostname 和 ssh key 回去。如果此時來源端 IP 被修改成對外的 public ip(floating ip)，Metadata API 服務就不清楚發出查詢需求的虛擬機是哪一台了。這時我們就可以將 dmz_cidr 設為 192.168.100.0/24，確保這個連線不會被 nova-network 所定義的 iptables 規則修改掉來源 IP(SNAT) 了。[8] [9]

VNC

OpenStack Essex 版本後建議改用 novnc 這個套件 [17]，因此我們除了安裝相關套件之外，還必須把 novnc_enabled 設為 true。值得一提的是，novncproxy_base_url 這個值會被使用在 Dashboard 的 iframe 上，用來指出你的 nova-novncproxy 這隻程式 (剛剛安裝在 controller 上了) 所在的 IP 跟連接端口 (預設是 6080)，因為 Dashboard 的使用者正常情形下皆來自於 Internet，所以這邊必須使用 controller 的公開 IP，使用者才能正常的與 nova-novncproxy 連線。至於 vncserver_listen 這個參數指定了每一台 compute worker 上的 vncserver(KVM 內建) 要 listen 的 IP，為了讓 novncproxy 可以存取，而且又因為安全性問題不想讓整個 Internet 的機器都可以連線，所以我們設定為 listen 在私有 IP 上。

API Listen

我們在 nova.conf 裡 (其實所有 OpenStack 設定檔都是) 都將各種 API(如 ec2_listen, osapi_compute_listen 等) 設定成要 listen 在私有 IP，避免讓服務暴露在 Internet 上，否則 OpenStack 裡預設都是 listen 在 0.0.0.0，可能會有潛在地安全性問題。只是如此一來，各種指定 API 位址的參數，(如 ec2_host, osapi_host 等)，或是 Keystone 的 catalog，甚至於設定環境變數 (如.novarc, .glancerc 等)，都要設定好服務所在的私有 IP，不可以是預設的 127.0.0.1，否則會無法正常的連線。

Quota

OpenStack 可以針對每個專案 (tenant 或 project 常會混用) 制定限額，可以限制的內容包含了可用的記憶體總量 (quota_ram)，可啟動的虛擬機總數 (quota_instances)，可使用的 CPU 總數 (quota_cores) 等，這些限制一般預設的值會太寬鬆，因此可以參考附錄 B.3修改預設的限額設定。

[8] dmz_cidr 的規則在 routing_source_ip 的規則之前，因此任何由虛擬機流向 DMZ 的流量均不會被修改掉來源 IP。

[9] 由於 nova-network 已經有幫 metadata_host 設定排除規則，所以 dmz_cidr 可以設定為其他你想要虛擬機可以直接訪問的私有網段。

虛擬機網路設定

NCTU CStack 系統使用 VLAN 模式 (network_manager=nova.network.manager.VlanManager)。其中較重要的參數是 vlan_interface，用來指定哪一張實體網卡要開啟 VLAN 的功能，從它流出的封包將會帶 VLAN Tag 來區隔虛擬機上的流量；另外 public_interface 則是可以靜態的在 nova.conf 指定，或是動態的產生 floating ip 時給定，其代表的意思就是指定的 floating ip 要綁在哪個網路卡上，我們預設使用 br0。

在我們的環境中，因為 PHA 模式的網路架構，所以有使用 [附錄 B.4] 這個 patch，patch 中提供兩個參數可供自訂：float_gw_ip 代表動態給定的 floating ip 希望使用的預設閘道，以及 float_cidr 代表預設子網路遮罩。

第五章　Dashboard 與 RADOS gateway 的整合

在前一章我們提到了如何將 OpenStack 的 IaaS 服務跟 Ceph 做一個結合，而這個章節要來介紹 RADOS gateway 這個服務的安裝，以及怎麼設定讓 Dashboard 使用 Radosgw 提供的 OpenStack Swift API 和 Amazon S3 API，來提供完整的物件儲存服務。

5.1　安裝 Radosgw

Ceph 做為一個亮眼的分散式檔案系統，除了提供物件儲存 (RADOS)、Block Device(RBD)、Ceph FS 等各種應用方式，還利用了 Apache 或是 Nginx 的 FastCGI 模組，支援 Amazon S3 與 OpenStack Swift 的 API，以下介紹搭配 Apache2 的安裝方式。

1. 首先是安裝 Apache2 和 FastCGI 模組，並且把他們開啟。

```
rgw-1# aptitude install apache2 libapache2-mod-fastcgi
rgw-1# a2enmod rewrite
rgw-1# a2enmod fastcgi
```

2. 參考 4.1 節加入 Ceph 官方套件庫。
3. 安裝 Radosgw。

```
rgw-1# aptitude install radosgw
```

4. 在 Ceph monitor 上產生一把給 Radosgw 專屬的 key，複製回 Radosgw node 上，並在 ceph.conf 上作相對應的設定。

```
mon-1# ceph-authtool --create-keyring radosgw.gateway.keyring
mon-1# ceph-authtool radosgw.gateway.keyring -n client.radosgw.gateway --gen-key
mon-1# ceph auth add client.radosgw.gateway \
    mds 'allow' osd 'allow rwx' mon 'allow r' \
    -i radosgw.gateway.keyring
mon-1# scp radosgw.gateway.keyring rgw-1:/etc/apache2/radosgw.gateway.keyring
rgw-1# chown www-data /etc/apache2/radosgw.gateway.keyring
rgw-1# vim /etc/ceph/ceph.conf
```

```
[client.radosgw.gateway]
    # radosgw服務所在的主機名，需填寫正確否則服務無法啟動。
    host = rgw-1
    keyring = /etc/apache2/radosgw.gateway.keyring
    rgw socket path = /tmp/.radosgw.gateway.keyring
    rgw cache enabled = 1
    # radosgw服務所使用的domain name以及所綁的連接端口
    rgw dns name = s3.nctu.edu.tw
    rgw swift url = http://s3.nctu.edu.tw:8080
    rgw swift url prefix = swift
    rgw print continue = false
    # 使用syslog來記錄
    log file = ""
    syslog = true
```

5. 接著我們把 Radosgw 的 fcgi 程式放在/var/radosgw 下，並設定好 Apache2 讓他知道怎麼呼叫 fcgi。

```
rgw-1# mkdir -p /var/radosgw
rgw-1# vim /var/radosgw/radosgw.fcgi
rgw-1# chmod +x /var/radosgw/radosgw.fcgi
```

```
#!/bin/shexec
/usr/bin/radosgw -c /etc/ceph/ceph.conf -n client.radosgw.gateway \
    --rgw-socket-path=/tmp/.radosgw.gateway.keyring -d
```

```
rgw-1# vim /etc/apache2/sites-enabled/rgw.conf
```

可以參考 [附錄 A.4]

```
// 若要讓radosgw可以支援Amazon S3 API，必須加入這兩行。
RewriteEngine On
RewriteRule    ^/ (.*)    / radosgw.fcgi? % {QUERY_STRING}    [E=HTTP_AUTHORIZATION: %
{HTTP:Authorization},L]
```

6. 我們可以手動執行 radosgw 服務看看與 ceph 的連線是否正常，接著再啟動服務。

```
rgw-1# su -s /bin/sh -c "exec radosgw -n client.radosgw.gateway -d" www-data
rgw-1# /etc/init.d/radosgw start
rgw-1# /etc/init.d/apache2 restart
```

5.1.1 安裝 Vanish

Ceph 官網有提到 RADOS gateway 並沒有任何 caching 的機制，所以在處理大量小檔案的時候會有較高的延遲時間 [2]。因此建議可以使用像是 Varnish, Squid 這類網頁快取服務來提高 RADOS gateway 的效能。我們將 Varnish 與 Radosgw 服務安裝在同一台機器上，並且讓 Varnish(端口:80) 快取 Radosgw(端口:8080) 的所有資料，使用 s3.nctu.edu.tw:80 對外提供統一的物件儲存窗口。

1. 安裝 Varnish

```
rgw-1# aptitude install varnish
```

2. 設定 Varnish 要綁定的 IP 位址 (140.113.98.246:80)，以及要使用多少記憶體來當快取 (6G)。

```
rgw-1# vim /etc/default/varnish
```

```
DAEMON_OPTS="-a 140.113.98.246:80 \
             -T localhost:6082 \
             -f /etc/varnish/default.vcl \
             -S /etc/varnish/secret \
             -s malloc,6G"
```

3. 設定 Varnish 的 ACL，限制來源 IP，以及設定快取來源的部分。可參考 [附錄 A.5]

```
rgw-1# vim /etc/varnish/default.vcl
```

```
acl allowclient {
    "140.113.0.0"/16;
    "140.114.0.0"/16;
    "192.168.0.0"/16;
}
\\ 設定後端要被快取的位置
backend radosgw {
    .host = "140.113.98.246";
    .port = "8080";
    .probe = {
        .url = "/";
        .interval = 5s;
        .timeout = 1s;
        .window = 5;
        .threshold = 3;
    }
}

sub vcl_recv {
    if (client.ip !~ allowclient) {
        error 403 "Not allow source IP.";
    }

    set req.backend = radosgw;
    ...
```

4. 設定完成後，重開 Varnish 即可。

```
rgw-1# /etc/init.d/varnish start
```

5.2 Swift 的認證與授權

我們在 4.2.1小節有提到，controller 在安裝 Keystone 的時候可以指定 object-store 這個服務的 endpoint 在哪 IP 哪個 port 上，讓 Dashboard 可以正確地與 Swift API 溝通，提供 end-user 物件儲存的前端介面；另一方面，我們也在前一節安裝好了 RADOS gateway 來提供 Swift API，使用 Ceph 來支援物件儲存的服務，而不是使用 OpenStack Swift 這個程式。

可是問題來了，Dashboard 會使用 Keystone 的授權來向任何 catalog 提供的 endpoint API 要求服務，舉例來說，若我們使用了 bob 帳號登入了 Dashboard，會得到一組 Keystone 分配的 auth-key(類似 session-key)，那我們就可以使用這組 auth-key 來向任何有跟 Keystone 建立授權連線的服務來進行操作。

但是目前版本的 RADOS gateway 雖然提供了相容的 Swift API，但是並不會向 Keystone 要求授權，取而代之的是 Radosgw 自己維護了一套帳號密碼授權的對應 (藉由 radosgw-admin)，如圖 5.1。

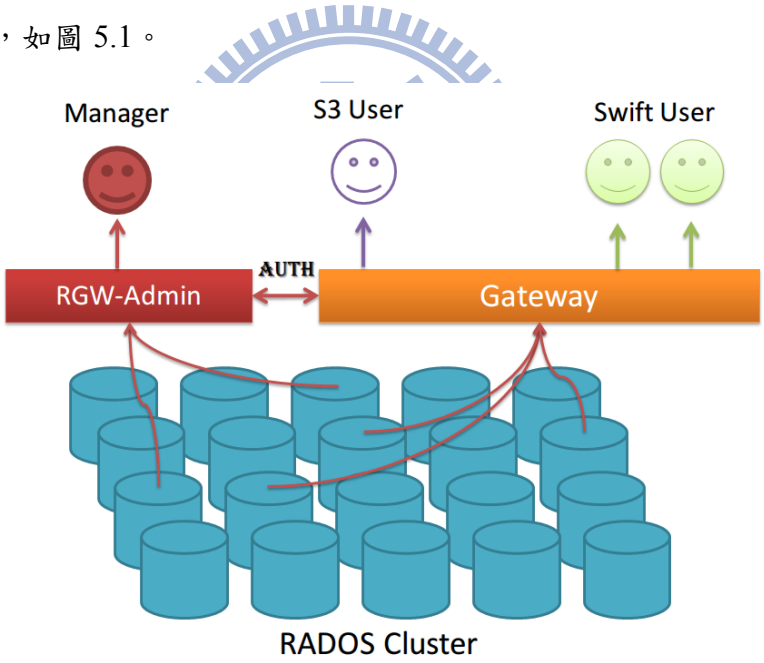

圖 5.1: RADOS gateway 認證示意圖

因此我們幫 radosgw-admin 撰寫了一個 python 介面的 API - rgwauthAPI[附錄 C.1]，讓其他使用者可以使用 python 方便的對 Radosgw 新增或刪除 Swift/S3 協定的使用者，以及其他權限上的操作。並且針對 Dashboard 提供的物件儲存介面的前端與後端，補充了一些令他相容於 Radosgw 的程式碼 [附錄 C.2]，讓 Dashboard 可以完全使用 Ceph RADOS gateway 來提供物件儲存的服務，相關內容都可以在 github 下載使用[1]。

[1]https://github.com/ChuanyuTsai/rgwauthAPI/

5.2.1 rgwauthAPI

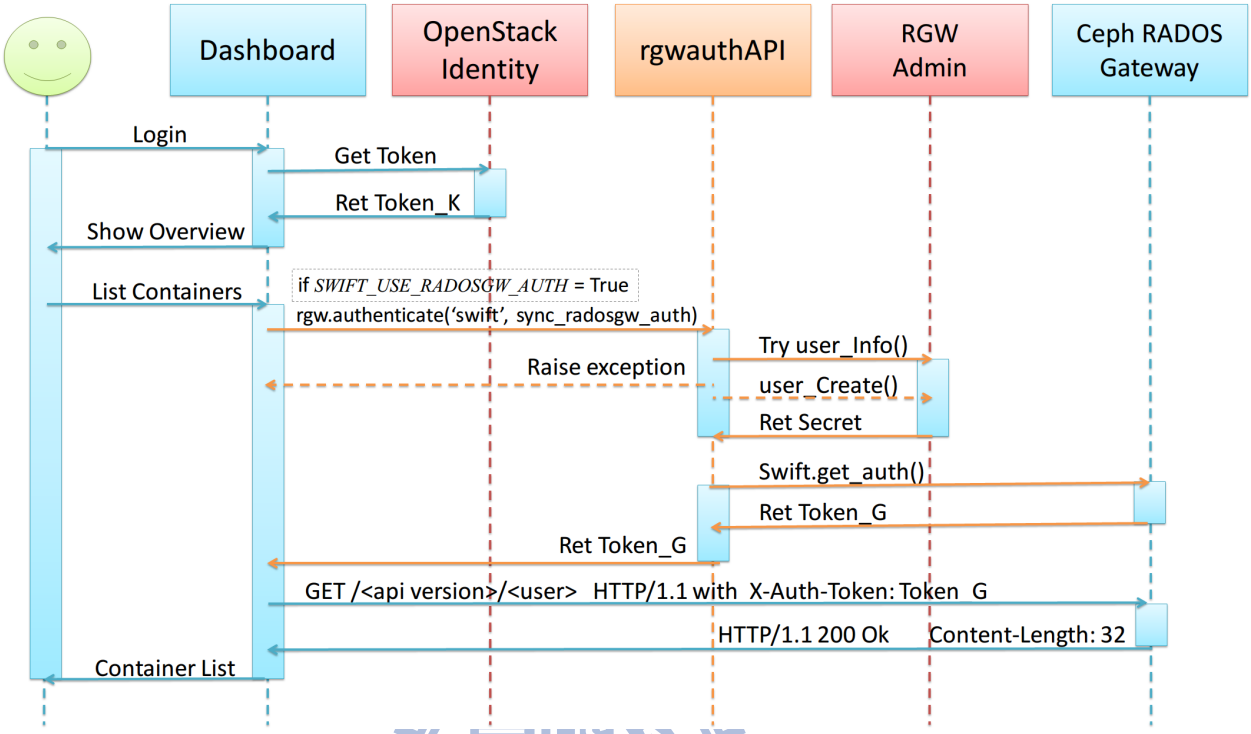

圖 5.2: rgwauthAPI sequence diagram

rgwauthAPI 依賴 python-swift，python-cloudfiles，以及 radosgw 三個套件，必須先安裝且設定好 cephuser 這個變數來指名呼叫 radosgw-admin 時的使用者。

圖 5.2說明了這個 module 的運作流程。使用上提供了 authenticate - user auto create 模式以及 not auto create 模式。若是使用 auto create 模式則不存在的使用者要求認證時，rgwauthAPI 會自動幫該使用者在 radosgw-admin 上建立一個一般使用者權限的帳號，反之則會回傳使用者不存在的例外情形。

NCTU CStack 系統預設每個專案 (tenant) 都擁有物件儲存的服務，因此任何可供使用者登入 (啟用中) 的專案皆會在向 rgwauthAPI 認證時，預設使用 user auto create 模式來做認證，減輕了系統管理者需要手動建立帳號的負擔。

在確認了使用者存在 Ceph RADOS Gateway 上後，rgwauthAPI 會試著跟 Ceph 連線，並取回新的 token，回傳給 dashboad，讓 dashboard 可以繼續接下來的操作。

radosgw-admin 在設計上規定一個使用者可包含多個 subuser，subuser 可各自設定權限並且在建立時可選擇是透過 S3 API 認證或是 Swift API 認證，所有 subuser 皆分享同一個空間 (可存取相同的物件以及容器)。因此 rgwauthAPI 也提供了刪除 subuser，以及

https://github.com/ChuanyuTsai/horizon/

在刪除使用者時會自動刪除屬於該使用者的物件以及容器，避免這些無法再存取的物件和容器永久的占用 ceph 儲存系統的空間，有效的減少 object leak 的問題 [5]。

5.2.2 Dashboard Patches

有了 rgwauthAPI 後就可以修改 Dashboard 來使用它了。在 patch Dashboard 的部分為了維護方便，並且需要相容於之後的官方更新，我們使用 GIT 並從官方的程式碼 fork 出自己的 repository，在上面做修改後，可以輕鬆的產生 patch 並安裝到上線的機器中。

這些 patches 包含了 rgwauthAPI 的支援，提供顯示 S3 服務入口，修正物件名稱有斜線時不能正常顯示的問題，修正專案的使用空間及配額限制，並修正建立虛擬機時規格列表的顯示內容等。

rgwauthAPI 的支援

OpenStack Keystone	Operations	Radosgw-admin
A Tenant	Enable/Suspend	A User
Multiple Users	Add/Remove	Multiple Subusers
Share Same Containers	Create/Delete	Share Same Buckets

Table 5.1: 認證關係的對應

表格 5.1說明了在 NCTU CStack 系統上，Keystone 和 Radosgw-admin 的權限對應關係。我們將 Radosgw-admin 上的 User 對應到 Keystone 的專案，這樣 subuser 對應到 Keystone 的使用者，這樣可以使得相同專案的所有使用者存取共同的容器以及物件，彼此可以分享跟專案相關的檔案資料，也比較符合我們的需求。

這裡主要是修改了/horizon/api/swift.py 這個檔案，並在設定檔增加了兩個參數來讓系統管理者決定要以甚麼方式支援 RADOS gateway：
SWIFT_USE_RADOSGW_AUTH：決定是否要改用 rgwauthAPI 來認證；
SYNC_KEYSTONE_RADOSGW_AUTH：決定是否要使用 user auto create 的認證模式。若設定成 False，則只有管理者手動加入的專案才能使用物件儲存的服務。

1. /horizon/api/swift.py

   嘗試與 rgwauthAPI 認證的程式碼。

```
 def swift_api(request):
     endpoint = url_for(request, 'object-store')
+    auth_token = request.session['token']
+
+    use_radosgw_auth = getattr(settings, 'SWIFT_USE_RADOSGW_AUTH', False)
+    sync_radosgw_auth = getattr(settings, 'SYNC_KEYSTONE_RADOSGW_AUTH', False)
+    if (use_radosgw_auth == True):
+        uid,subuser = request.session['tenant_id'],request.session['user_id']
+        try:
+            rgw = RGW(uid, subuser, endpoint)
+            storage_url, auth_token = rgw.authenticate('swift', sync_radosgw_auth)
+        except Exception as e:
+            # rgwauthAPI now puts error msg in args[0]
+            raise exceptions.RadosgwExceptions(str(e.args[0]))
+    auth = SwiftAuthentication(storage_url, auth_token)
     return cloudfiles.get_connection(auth=auth)
```

2. /horizon/exceptions.py

   定義一個新的例外情形來處理 rgwauthAPI 認證失敗的情形。

```
+class RadosgwExceptions(HorizonException):
+    def __init__(self, msg):
+        self.msg = msg
+
+    def __repr__(self):
+        return self.msg
+
+    def __unicode__(self):
+        return _(self.msg)
```

3. /horizon/api/swift.py

   由於 radosgw 在接收上傳的物件時需要清楚的知道物件的檔案大小，但是目前的 Dashboard 只提供檔案名稱，因此需要額外幫它加上去。

```
 def swift_upload_object(request, container_name, object_name, object_file):
     container = swift_api(request).get_container(container_name)
-    obj = container.create_object(object_name)
+    # radosgw needs to know the size exactly
+    obj = cloudfiles.storage_object.Object(container, object_name,
+            object_record = {
+                'name': object_name,
+                'content_type': object_file.content_type,
+                'bytes': object_file._size,
+                'last_modified': None,
+                'hash': None
+            })
     obj.send(object_file)
     return obj
```

4. /horizon/dashboards/syspanel/projects/tables.py

由於我們希望 Radosgw-admin 的帳號可以與 Keystone 的同步，所以當有專案或使用者被刪除時，也要透過 rgwauthAPI 來將相對應的使用者與 subuser 刪除。

需要注意的是專案的刪除必須遞迴的將所有屬於這個專案的使用者移出這個專案後才能將該專案刪除，不然會導致有的使用者明明還有其他專案的權限，卻在登入 Dashboard 時遭遇" 不屬於任何專案" 的錯誤而無法登入，這邊一併做修正。

```
class DeleteTenantsAction(tables.DeleteAction):
     def delete(self, request, obj_id):
+        use_radosgw_auth = getattr(settings, 'SWIFT_USE_RADOSGW_AUTH', False)
+        sync_radosgw_auth = getattr(settings, 'SYNC_KEYSTONE_RADOSGW_AUTH', False)
+        if use_radosgw_auth and sync_radosgw_auth:
+            endpoint = url_for(request, 'object-store')
+            try:
+                LOG.debug('Radosgw remove user %s from endpoint: %s'
+                        % (obj_id, endpoint))
+                RGW(obj_id, 'admin', authUrl=endpoint).rmUser()
+            except Exception as e:
+                exceptions.handle(request, _("Unable to delete rgw user."))
+        for user in api.keystone.user_list(request, obj_id):
+            api.keystone.remove_tenant_user(request, obj_id, user.id)
...
     def action(self, request, user_id):
         tenant_id = self.table.kwargs['tenant_id']
+        use_radosgw_auth = getattr(settings, 'SWIFT_USE_RADOSGW_AUTH', False)
+        sync_radosgw_auth = getattr(settings, 'SYNC_KEYSTONE_RADOSGW_AUTH', False)
+        if use_radosgw_auth and sync_radosgw_auth:
+            endpoint = url_for(request, 'object-store')
+            uid, subuser = tenant_id, user_id
+            try:
+                LOG.debug('Radosgw remove subuser %s:%s from endpoint: %s'
+                        % (uid, subuser, endpoint))
+                RGW(uid, subuser, endpoint).rmSubuser()
+            except:
+                redirect = reverse("horizon:syspanel:projects:users",
+                                                    args=(tenant_id,))
+                exceptions.handle(request, _('Remove user failed.'),
+                                                    redirect=redirect)
```

額外的 S3 連接方式

由於 Radosgw-admin 本身即支援 OpenStack Swift API 以及 Amazon S3 API 這兩種常見的物件儲存服務的協定，有鑑於大量軟體對於 S3 API 的支援與開發程度，我們另外設計了一種方式讓專案內的使用者可以用其他相容 S3 API 的軟體來使用我們提供的物

件儲存服務，這也是原本的 OpenStack Essex Swift 程式沒有的功能[2]。

具體做法是在建立 Radosgw 的使用者時一起建立一組 S3 協定的 subuser，並且在 Dashboard 的物件儲存頁面提供一個"ShowKey" 頁面顯示那組 subuser 的 Access Key 和 Secret Key 以及 5.1.1小節提到的存取點 http://s3.nctu.edu.tw/，使用者有了這些資訊後就可以使用自己喜歡軟體透過 S3 協定來操作專案上面的容器或是物件，甚至在專案裡啟動的虛擬機上也可以安裝 s3fs[3]來將專案上的容器掛載到虛擬機上的任意資料夾，有些使用情形下這是個非常方便的功能。

由於 Dashboard 是使用 python 的 Django 來開發的一套 MVC framework，因此我們僅需在 view 與 controller 的地方加入相對應的程式碼即可完成這些功能。

1. /horizon/dashboards/nova/containers/tables.py

   如果支援 Radosgw 的參數被設置起來，就在物件儲存的首頁增加 Show Keys 的按鈕來讓使用者取得 S3 服務的連接方式。

```
+class ShowKeys(tables.LinkAction):
+    name = "show_keys"
+    verbose_name = _("Show Keys")
+    url = "horizon:nova:containers:show_keys"
+    classes = ("ajax-modal", "btn-create")
+
```

```
+        if getattr(settings, 'SWIFT_SHOW_RADOSGW_KEYS', False):
+            table_actions = (ShowKeys, CreateContainer, DeleteContainer)
+        else:
+            table_actions = (CreateContainer, DeleteContainer)
```

2. /horizon/dashboards/nova/containers/urls.py

   Django MVC 中的 controller 設定檔，呼叫 ShowKey 的 action。

```
+    url(r'^show/$', ShowKeysView.as_view(), name='show_keys'),
```

3. /horizon/dashboards/nova/templates/nova/containers/_showkeys.html

   Django MVC 中的 view，設定 ShowKey 的左邊是一個表格，右邊為說明文字。

```
+<div class="left">
+    <fieldset>
+        {% include "horizon/common/_form_fields.html" %}
+    </fieldset>
+</div>
+<div class="right">
+    <h3>{% trans "Description" %}:</h3>
```

[2]新版的 OpenStack Folsom Swift 也提供了 swift3 middleware 來讓 Swift 對外提供相容於 Amazon S3 API 的物件儲存服務。

[3]一種使用 FUSE 開發的檔案系統。可以使用 Amazon S3 協定來將遠端的 S3 服務掛載到本機資料夾，任何對該資料夾的讀寫操作即會反映到遠端的伺服器上，類似現在流行的 Dropbox 服務。

```
+          <p>{% trans "We also support the Amazon &reg; AWS S3 compatible service. If you want to try a fresh, the Access Key & Secret Key are shown in the left side. Moreover, you can also mount a bucket as local drive by Gladinet &reg; or other clients use S3 API.      Notice! Dashboard still *NOT* support directory in bucket, so you can't see the files in directory which is uploaded from S3 API." %}</p>
+</div>
+{% endblock %}
+
+{% block modal-footer %}
+   <a href="{% url horizon:nova:containers:index %}" class="btn btn-primary pull-right">{% trans "Back" %}</a>
+{% endblock %}
```

4. /horizon/dashboards/nova/containers/forms.py

   View 的左半部用表單的方式呈現，因此需在這邊定義有哪些欄位。

```
+class ShowKeys(forms.SelfHandlingForm):
+    def __init__(self, *args, **kwargs):
+        helpStr = "Ctrl+C Copy to Clipboard"
+        self.fields['apoint'] = forms.CharField(
+            initial=accessPoint, label='Access Point', help_text=helpStr)
+        self.fields['access'] = forms.CharField(
+            initial=user["keys"][0]["access_key"],
+            label='Access Key ID', help_text=helpStr)
+        self.fields['secret'] = forms.CharField(
+            initial=user["keys"][0]["secret_key"].replace('\\',''),
+            label='Secret key', help_text=helpStr)
+        self.fields['apoint'].widget.attrs['onClick'] = \
+            self.fields['access'].widget.attrs['onClick'] = \
+            self.fields['secret'].widget.attrs['onClick'] = \
+            'Javascript:this.focus();this.select();'
```

5. /horizon/locale/zh_TW/LC_MESSAGES/django.po

   Django 的多語系支援，撰寫關於 ShowKey 的繁體中文說明文件。

```
+msgstr ""
+"我們也提供了相容於 Amazon &reg; AWS S3 的服務。  如果您想試試，左邊提供了必要的 "
+"Access Key & Secret Key。  而且，您還可以試著用像 Gladinet &reg; 這樣支援 S3 "
+"API 的軟體把專案裡的 bucket 掛載到本地端的機器上當作一顆磁碟。    請注意! 目前為"
+"止，Dashboard 還*不支援*顯示 bucket 裡的資料夾，所以如果您使用 S3 的協定上傳資料"
+"夾，將不會顯示在 dashboard 上。"
```

物件名稱有斜線不能正常顯示的問題

這個問題是因為提供了 S3 的服務之後產生的問題。主要原因是某些支援 S3 協定的軟體可以把 S3 空間上的某個容器 (bucket) 掛載到本機當作一個磁碟或資料夾，因此我們就可以在該資料夾進行日常的檔案操作，當然也包括在資料夾內再建立一個資料夾，而這些資料夾 (或資料夾內的檔案) 在物件儲存的機制下就僅是一個名稱內有斜線的物

件 (如"buck1/file1")。

但不幸的是，目前版本 (Essex 穩定分支) 的 Dashboard 還不支援顯示 bucket/container 內的資料夾，因此這邊針對名稱裡有斜線的物件暫時先過濾掉不顯示在 Dashboard 上，讓 Dashboard 只能顯示容器內第一層的所有物件，但並不會影響到其他使用 S3 服務的其餘使用者，而且由於我們使用 GIT 來開發這些功能，在官方修正了相關的問題之後也可以很容易的將那些更新合併 (merge) 進來。

1. /horizon/api/swift.py

   過濾掉所有名稱內包含斜線的物件。

```
+def swift_filter_objects(s):
+    return s[u'name'].find('/') == -1
+
+def swift_filter_names(s):
+    return s.find('/') == -1

def swift_get_objects(request, container_name, prefix=None, marker=None):
     objects = container.get_objects(prefix=prefix,
                                     marker=marker,
                                     limit=limit + 1)
+    # This branch doesn't support to show objects which have
+    # '/' in their names now (object is a directory or files in directorys)
+    # but ceph rados supports that, so the objects may be upload to
+    # container by s3 api. Therefore, just filter out objects like that.
+    objects._objects = filter(swift_filter_objects, objects._objects)
+    objects._names = filter(swift_filter_names, objects._names)
```

規格列表的內容修正

一般 Dashboard 預設的虛擬機規格 (flavor) 是只有 m1.tiny 的主磁碟為 0GB，其他規格為 10GB，但是在啟動新虛擬機時的規格檢查，0GB 其實是不檢查，就是可以開啟任意大小的映像檔的意思。這樣就會造成一旦我們上傳了較大的映像檔 (大於 10GB)，卻只有規格是 m1.tiny 的虛擬機可以啟動，其他較高規格的虛擬機卻因為映像檔大小大於主磁碟 10GB 的限制而不能開機，這樣是很矛盾的。

因此解決辦法是將所有規格的主磁碟皆設為 0GB(因為我們可以決定那些專案可以開啟那些映像檔)，主磁碟大小我們就不計入磁碟使用量裡面，至於在供選擇的規格列表裡就改成顯示暫用磁碟，讓使用者可以知道所選規格的虛擬機還會附帶一顆多大容量的本機磁碟，這樣較符合使用情境。

1. /horizon/dashboards/nova/images_and_snapshots/images/views.py

   在使用者要啟動虛擬機時，改成顯示暫用 (ephemeral) 磁碟大小供使用者選擇。

```
 class LaunchView(forms.ModalFormView):
    flavors = api.flavor_list(self.request)
    flavor_list = [(flavor.id, display % {
        "name": flavor.name,
        "vcpus": flavor.vcpus,
-       "disk": flavor.disk,
+       "disk": getattr(flavor, "OS-FLV-EXT-DATA:ephemeral"),
        "ram": flavor.ram})
```

2. /horizon/dashboards/nova/instances_and_volumes/instances/tables.py

將"執行個體 & 容量"的首頁改成顯示暫用磁碟的大小，方便使用者掌握專案的空間使用情形。

```
 def get_size(instance):
     if hasattr(instance, "full_flavor"):
-        size_string = _("%(RAM)s RAM | %(VCPU)s VCPU | %(disk)s Disk")
+        size_string = _("%(RAM)s RAM | %(VCPU)s VCPU | %(disk)s Disk(vdb)")
         vals = {'RAM': sizeformat.mbformat(instance.full_flavor.ram),
                 'VCPU': instance.full_flavor.vcpus,
-                'disk': sizeformat.diskgbformat(instance.full_flavor.disk)}
+                'disk': sizeformat.diskgbformat(
+                        getattr(instance.full_flavor, 'OS-FLV-EXT-DATA:ephemeral'))}
         return size_string % vals
     return _("Not available")
```

專案的配額與已使用空間的顯示

Dashboard 的使用者可以在啟動新虛擬機時看到目前該專案的配額使用情況，目前的程式指出配額的磁碟已使用空間是代表所有主磁碟以及暫用磁碟的總使用量，但是如同上述所說的，計算使用者的主磁碟使用量是不夠精準的 (例如，一個高規格的虛擬機 (主磁碟配給 80GB) 開啟一 250MB 的 Ubuntu 卻被記入使用了 80G 的磁碟使用量)，因此我們改成只計入暫用磁碟，並且加入該專案在物件儲存的部分使用了多少的空間，來當作磁碟配額的限制對象。

1. /horizon/api/swift.py

先撰寫一函式可供查詢該專案在物件儲存部分的總使用量。

```
+def swift_get_total_size(request):
+    size=0
+    try:
+        containers = swift_api(request).get_all_containers(10000,None)
+    except Exception:
+        containers = None
+    for container in containers:
+        size += container.size_used
+    return size
```

2. /horizon/api/nova.py

再來將該專案的磁碟使用量 (gigabytes) 改成計算暫用磁碟以及物件儲存的部分。

```
def tenant_quota_usages(request):
     usages = {'instances': {'flavor_fields': [], 'used': len(instances)},
               'cores': {'flavor_fields': ['vcpus'], 'used': 0},
-              'gigabytes': {'used': 0,
-                            'flavor_fields': [
-                               'disk',
-                               'OS-FLV-EXT-DATA:ephemeral']},
+              'gigabytes': {'used':
+                           swift_get_total_size(request)/1024/1024/1024,
+                           'flavor_fields': ['OS-FLV-EXT-DATA:ephemeral']},
               'ram': {'flavor_fields': ['ram'], 'used': 0},
               'floating_ips': {'flavor_fields': [], 'used': len(floating_ips)}}
```

3. /horizon/dashboards/nova/images_and_snapshots/images/views.py

在啟動新虛擬機的過程中判斷是否有任何的使用量超過原先的配額，有的話將標示為不允許啟動。

```
class LaunchView(forms.ModalFormView):
         try:
             context['usages'] = api.tenant_quota_usages(self.request)
+            for usage in context['usages']:
+           if usage != "floating_ips" and context['usages'][usage]['available'] <= 0:
+                   context['usages']['disabled'] = 'disabled=disabled'
+                   break
+            context['usages'].setdefault('disabled', '')
         except:
             exceptions.handle(self.request)
         return context
```

4. /horizon/dashboards/nova/templates/nova/images_and_snapshots/images/_launch.html

將配額使用量的顯示改成" 磁碟 + 容器 (使用 GB)"，告知使用者配額將計入物件儲存的使用量。

```
   <div class="quota_title">
-    <strong>{% trans "Disk" %}
-        <span>({{ usages.gigabytes.used }} {% trans "GB" %})</span>
-    </strong>
+    <strong>{% trans "Disk" %}+{% trans "Containers" %}
+        <span>({{ usages.gigabytes.used }} {% trans "GB" %})</span>
+    </strong>
     <p>{{ usages.gigabytes.available|quota:"GB" }}</p>
   </div>
...
 {% block modal-footer %}
-  <input class="btn btn-primary pull-right"
```

```
-         type="submit" value="{% trans "Launch Instance" %}" />
+   <input class="btn btn-primary pull-right" {{ usages.disabled }}
+         type="submit" value="{% trans "Launch Instance" %}" />
  {% endblock %}
```

第六章　Dashboard 操作介面

我們將在這一章完整介紹我們建置完成的 NCTU CStack 系統的 Dashboard 操作介面，雲端運算以及雲端儲存的使用者可以利用這個 Web 介面來使用服務，相當直覺方便。圖 6.1為一開始的登入畫面。之後將在 6.1節介紹專案頁面、6.2節介紹管理者頁面，以及 6.3節介紹個人設定等。

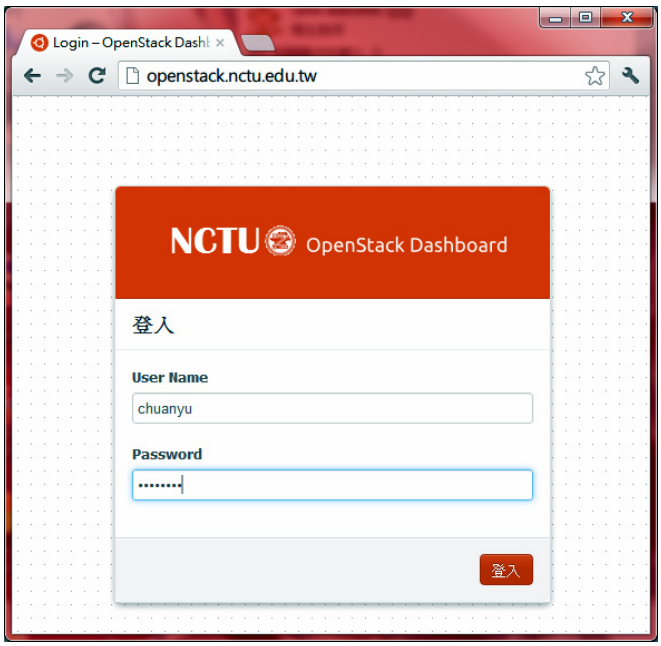

圖 6.1: 登入首頁 http://openstack.nctu.edu.tw/

6.1　專案頁面

圖 6.2顯示一般使用者登入後的第一個畫面：專案總覽。總覽分頁顯示了該專案目前正在執行的虛擬機 (或稱作執行個體) 總數，以及按月統計的處理器時數、GB 時數，右下方列出了該專案目前啟動的虛擬機規格，讓專案使用者可以掌握專案配額的使用情形。

圖的左邊有一個大的下拉式選單，供使用者快速切換不同的專案，而左半部的分頁

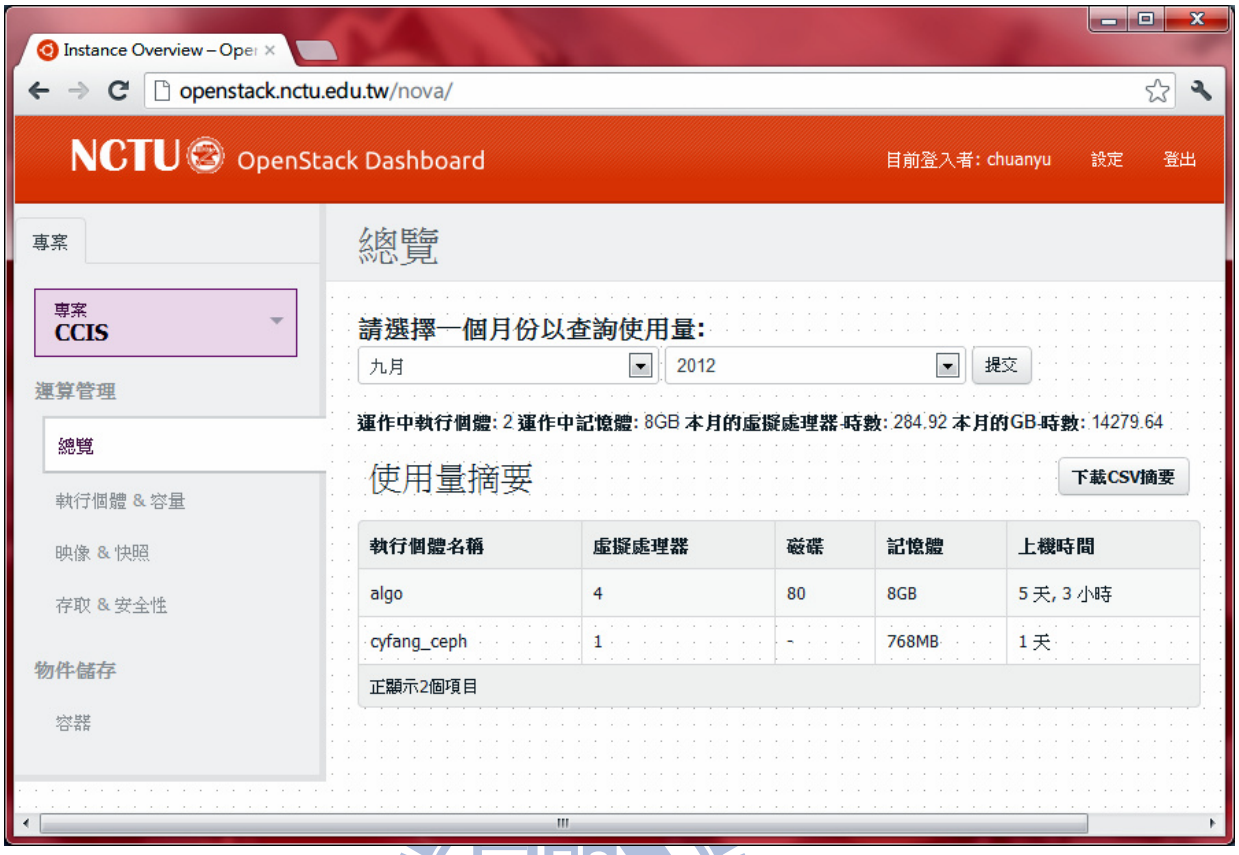

圖 6.2: 專案：總覽

選單可視為兩大部分：6.1.1小節將會介紹提供 IaaS 服務的運算管理，以及 6.1.2小節介紹提供物件儲存服務的" 容器"(Containers)。

6.1.1 運算管理

使用者可以利用這部分的分頁迅速地開啟所需的虛擬機。

映像 & 快照

" 映像 & 快照" 分頁 (圖 6.3) 列出了目前系統上可供使用的所有映像檔。

我們並沒有開放讓使用者可以自行上傳映像檔，目前提供了自行製作的 Windows 7、Windows XP 和 ArchLinux，以及修改過的的 Ubuntu 12.04 UEC 版本 (Ubuntu Enterprise Cloud) 和 Fedora 16 JEOS 版本 (Just Enough OS)，這些官方的 Linux prebuilt image 都是為了可以正常在 OpenStack 的環境上執行所製作的，想取得更多的映像檔可以參考 [11]。

若使用者依然有其他的需要可以告知我們所需的作業系統及版本，我們可以上傳官方的 ISO 檔供使用者自行安裝使用。

圖片下方列出了專案內的所有快照 (snapshot)，Dashboard 提供了方便的介面供使用

圖 6.3: 專案：映像 & 快照

者在任意時刻對虛擬機做快照，產生出來的快照可以當作新的映像檔，再用來開啟新的虛擬機，讓使用者可以利用它對虛擬機做備份或版本控制，是相當方便的功能。

按下了映像檔右邊的啟動之後，使用者就可以開始設定他要新啟動的虛擬機器 (圖 6.4)。

圖的右邊顯示了目前專案的各種配額剩餘情形，5.2.2小節有提到，我們的 patch 將" 物件儲存" 和" 磁碟" 使用量合併起來，用" 磁碟 + 容器" 的名稱來顯示給使用者，讓他知道目前的空間使用情況。若是使用者接下來預計開啟的虛擬機，有某項規格將會超過配額限制，啟動請求將會被拒絕。

另外，除了規格與配額的限制，使用者在開啟 Linux 系列的虛擬機時，必須指定這台虛擬機所綁定的" 金鑰" 跟" 安全性群組"。Linux 系列的虛擬機在製作時皆被設定成只

圖 6.4: 專案：建立新虛擬機 & 配額使用情形

能使用 SSH 金鑰來登入[1]，因此若我們沒有在建立虛擬機時指定金鑰，屆時將沒有任何一組帳號密碼可以用來登入虛擬機。金鑰的產生方法以及" 安全性群組" 會在" 存取 & 安全性" 子小節來做介紹。

使用金鑰來登入虛擬機的方法也很簡單，我們可以把在" 存取 & 安全性" 頁面產生的金鑰下載到任何一台工作站，在工作站上輸入以下指令來登入虛擬機：

```
// 假設金鑰名稱為：user.pem
// 虛擬機IP為：140.113.98.30
# chmod 600 user.pem
# ssh -i user.pem root@140.113.98.30
```

root 帳號預設為不能登入，僅用來顯示可正常登入的一般帳號，如：

```
The authenticity of host '140.113.98.30 (140.113.98.30)' can't be established.
ECDSA key fingerprint is 87:5d:68:46:45:f4:ff:4e:84:ec:0c:03:78:a7:da:73.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '140.113.98.30' (ECDSA) to the list of known hosts.
```

[1]Linux 系列的映像檔，皆把 ssh server 設定成 PubkeyAuthentication yes、PasswordAuthentication no。

```
Please login as the user "ubuntu" rather than the user "root".

Connection to 140.113.98.30 closed.
```

Ubuntu 12.04 UEC 預設的帳號為"ubuntu"，而 Fedora JEOS 預設的帳號為"ec2-user"，只要使用剛剛綁定的金鑰以及正確的帳號名稱，就可以順利的登入虛擬機了。

圖 6.5: 專案：存取 & 安全性

存取 & 安全性

這個分頁提供了讓專案使用者申請浮動 IP(在我們系統是給公開 IP) 的功能 (如圖 6.5)，這些申請來的 IP 是可以配給專案內任意的虛擬機，擁有浮動 IP 的虛擬機並不需要多做任何設定，OpenStack 就會自動將該 IP 的封包轉送給虛擬機，對外的來源 IP 也會改成新的浮動 IP，而且虛擬機也會升級成擁有 HA 的網路模式 (參考 3.3節)。另外就

算 IP 暫時沒有要使用，只要不釋放出去，該 IP 就不會被別的專案取得，讓專案使用者可以對外發布服務時可以使用一致的公開 IP，依照需求來調度公開 IP 的使用。

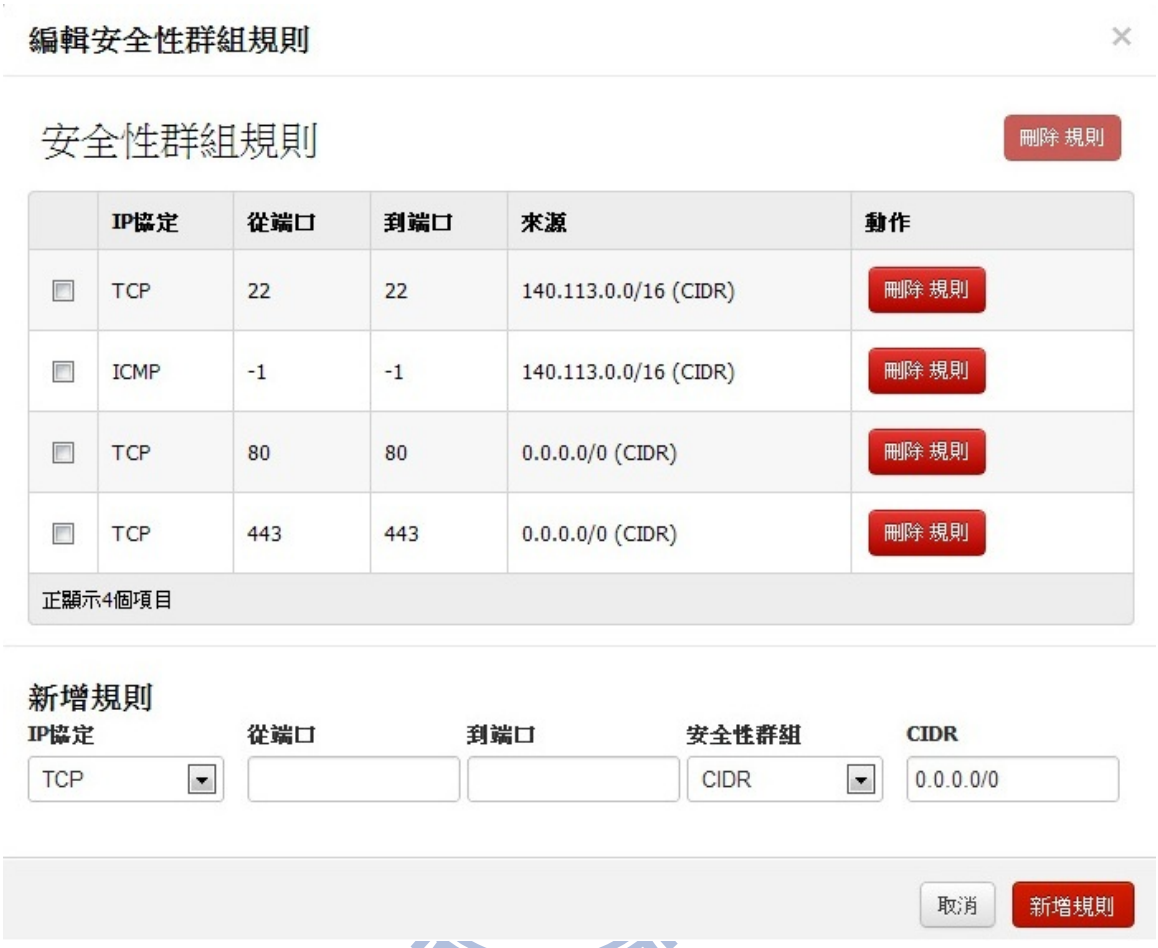

圖 6.6: 編輯安全性群組規則

此外，安全性群組也是 OpenStack 裡重要的設定之一。OpenStack 預設不會開放外部網路到虛擬機上的任何 TCP/IP 連線，使用者若希望使用除了 WebVNC 以外的方式控制虛擬機 (如 Windows 的遠端桌面連線、*nix 的 ssh 連線)，或是把虛擬機當作伺服器對外開放服務，就必須設定正確的防火牆規則，開放相對應的協定及連接端口 (port)。安全性群組可以讓使用者輕鬆的自訂防火牆規則 (如圖 6.6)，並且把多個規則集合成一個群組，然後再開啟新虛擬機時指定這個虛擬機適用的多個群組，這樣就可以控制虛擬機對外的連線能力。

金鑰管理也是" 存取 & 安全性" 內的一個功能，在這裡使用者只要輸入金鑰名稱，系統就會自動建立一組金鑰，公鑰的部分存入資料庫，待建立虛擬機時動態嵌入 root 帳號內，讓擁有私鑰的使用者可以正常連線；私鑰的部分即立刻供使用者下載，使用者需要自行妥善保存，之後系統將無法再次產生相同的金鑰。另外，金鑰雖然顯示在專案畫面內，但其實金鑰是跟著使用者本人的，別人是無法取得某個使用者的金鑰 (私鑰部

分)，而且也無法設定虛擬機嵌入別人擁有的金鑰。

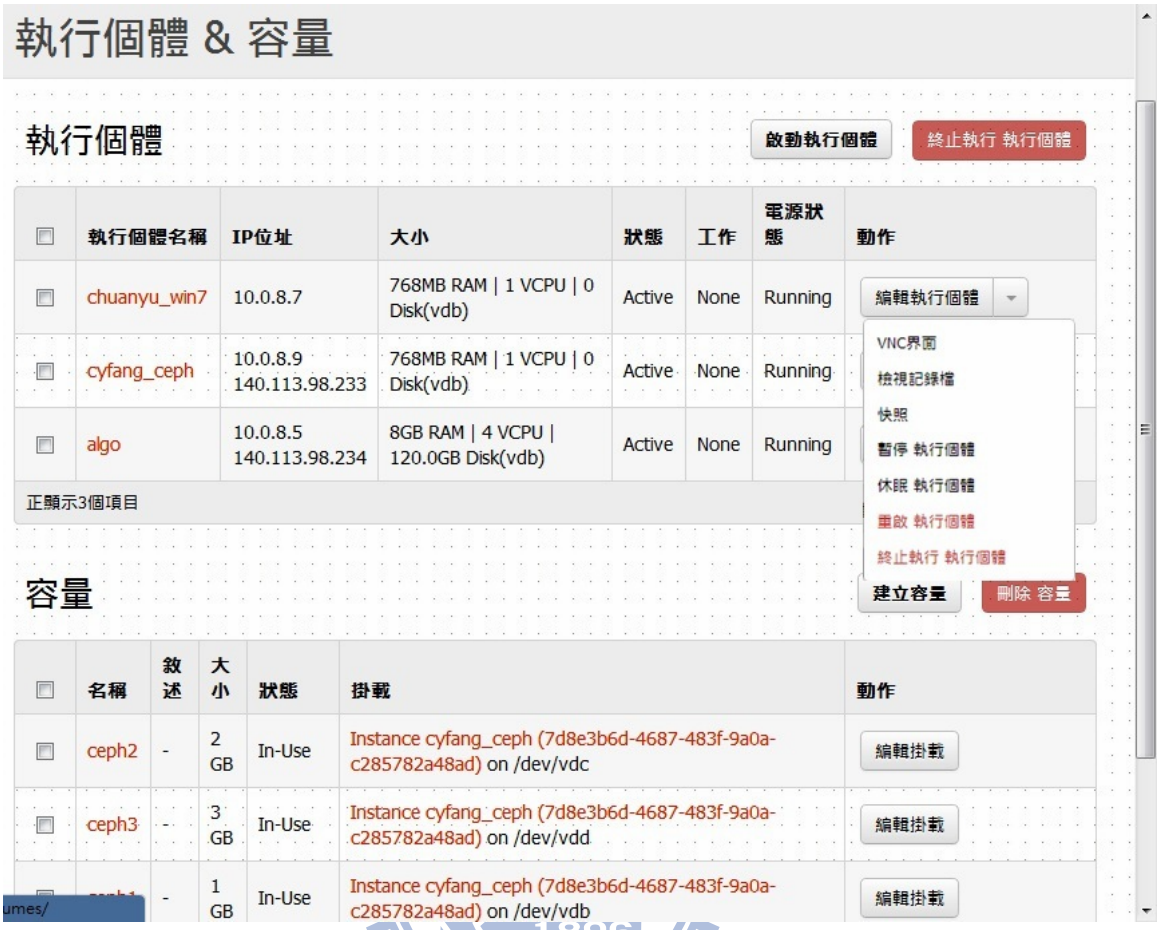

圖 6.7: 執行個體 & 容量

執行個體 & 容量

這個分頁顯示了這個專案內我們開啟的所有虛擬機以及它們各自的狀態 (圖 6.7)，每個虛擬機右邊都可展開向下清單，對虛擬機做更細微的控制，如：透過 VNC 介面查看 console 畫面[2]、對虛擬機做快照、休眠、重新啟動等功能。

" 容量" 是 Dashboard 官方翻譯自"volume" 而來，但依照功能來看比較好的翻譯應該是磁碟。這邊提供了專案新增額外磁碟的能力，並且可以依照需要動態的把磁碟掛載到不同的虛擬機上，大部分的作業系統都可以針對動態掛載的磁碟提供熱插拔支援，不用重新開機就可正常使用。使用者也可對磁碟安裝任意的作業系統，並利用磁碟來開機 (而不是透過我們提供的映像檔)，或是針對重要磁碟做快照備分。

到此，OpenStack 提供了相當完整的 IaaS 服務。

[2]有些版本的瀏覽器會把 Cookies 停用導致 Web VNC 功能失敗 (如 Safari)，此時只要 Cookies 功能開啟即可正常使用。

圖 6.8: 上圖：容器頁面。下圖：Show Keys 頁面。

6.1.2 物件儲存

"物件儲存" 是 OpenStack 的一個子計畫，考量雲端服務的使用情況，提供一個讓使用者可以存取任意大小的檔案物件的服務。Dashboard 也提供了相對應的網頁使用介面 (如圖 6.8上圖)，容器 (container) 的頁面簡單的統計該專案的物件儲存使用情況，所有使用者可以共用同一個專案內的所有容器，把物件儲存服務當作專案內的資料儲存中心。

由於僅使用網頁介面來存取物件，實用性有限，有鑑於美國 Amazon 公司的雲端服務發展已久，最有名的物件儲存服務 (S3) 更是被大家廣為使用，各家廠商在各平台發展了相當多支援 Amazon AWS S3 API 的軟體，S3 API 儼然成為物件儲存的主流。因此我們在頁面上提供了一個"Show Keys" 按鈕，裡面列出了使用 S3 API 連接的相關資訊 (圖 6.8下圖)，讓使用者可以使用網路上眾多的 S3 工具，連接我們的物件儲存服務。

範例：使用免費的工具存取物件儲存服務

我們介紹兩個 Windows 上的免費軟體 CloudBerry Explorer[3]和 Gladinet Cloud Drive[4]，他們皆是為了方便使用網路上的免費空間而開發的，兩者皆有能力存取 Amazon S3 compatible 的服務，Gladinet 還可以將免費空間掛載到本機一顆模擬的硬碟，讓使用者將檔案的上傳下載直覺的視為平常的檔案複製。

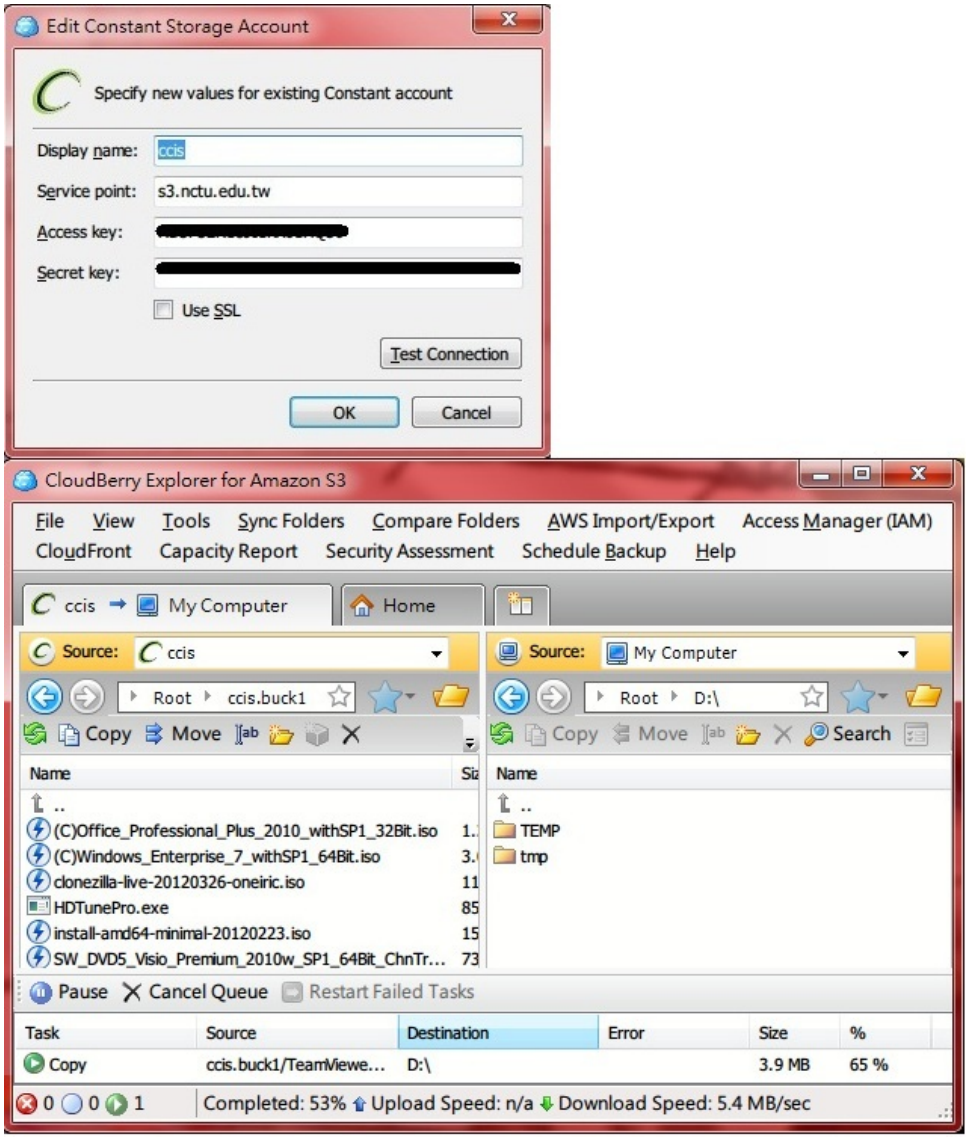

圖 6.9: 上圖：在 CloudBerry Explorer 上建立帳號來存取我們的物件儲存服務。下圖：使用 CloudBerry Explorer 的連線畫面。

圖 6.9是 CloudBerry 的操作畫面，在使用"Show Keys" 頁面內的資訊建立好帳號之後，左邊就會顯示該專案上物件儲存服務的所有物件，右邊是使用者的本機磁碟，使用

[3]http://www.cloudberrylab.com/free-amazon-s3-explorer-cloudfront-IAM.aspx
[4]http://gladinet.com/p/download_starter_V4.htm

者可以利用拖拉的方式操作物件的上傳下載，傳送進度將會顯示在下方的佇列中。

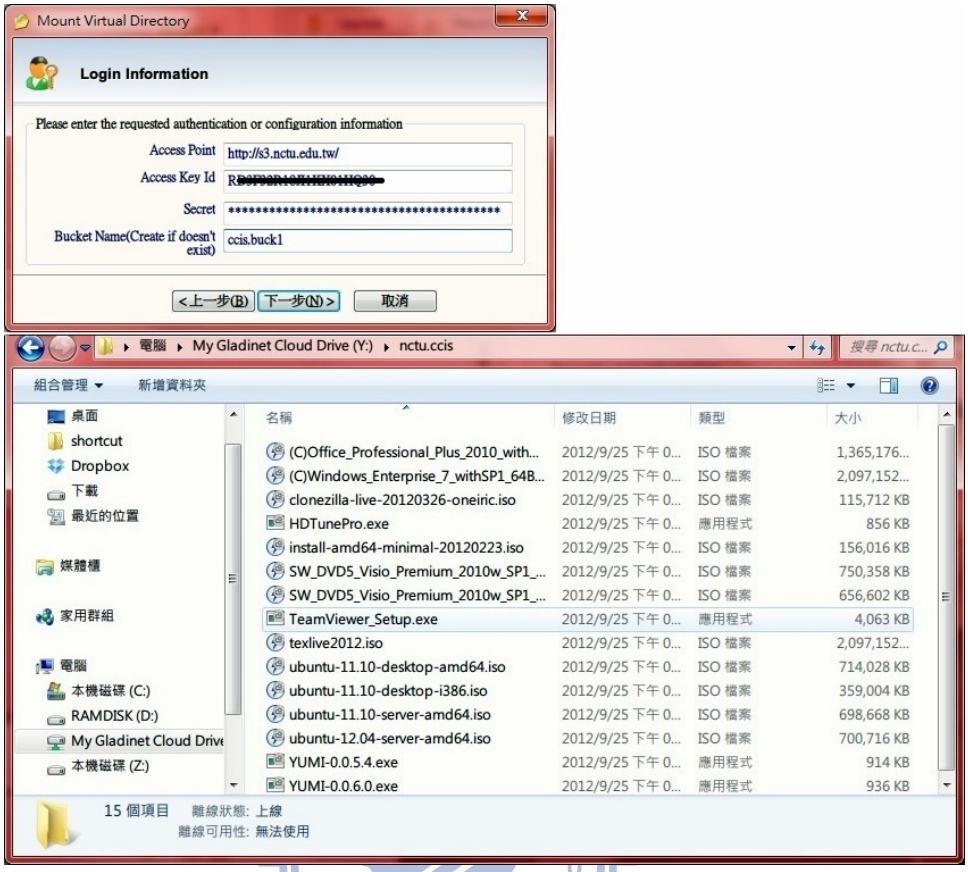

圖 6.10: 上圖：在 Gladinet Cloud Desktop Management Console 上建立帳號來存取我們的物件儲存服務。下圖：使用 Gladinet 掛載好的磁碟 Y: 來列出容器內的物件。

圖 6.10是 Gladinet 的帳號建立畫面。一樣將"Show Keys" 內的 access/secret key 填入之後，即可正確與我們的伺服器溝通，使用者就可以在自己的本機硬碟看到專案內的特定容器裡面的所有物件，本地端的應用程式也可直接對其存取操作，由 Gladinet 幫忙同步檔案，免除需要先下載才能使用的麻煩。

6.2 管理者頁面

擁有 admin 身分 (role) 的使用者即是 OpenStack 裡的管理者。管理者可以在 Dashboard 的專案分頁旁邊看到一個管理者分頁，該分頁提供一個系統面板的分區，裡面有需多重要的功能可以使用。

總覽跟執行個體大抵跟各別專案內部的功能類似，只是可以不分專案一次瀏覽到整個系統的使用狀況，執行個體還可以看到是運行在哪一台運算節點上，對於管理運算節點的系統負荷有相當的幫助 (圖 6.11)。

圖 6.11: 管理者：所有執行個體列表

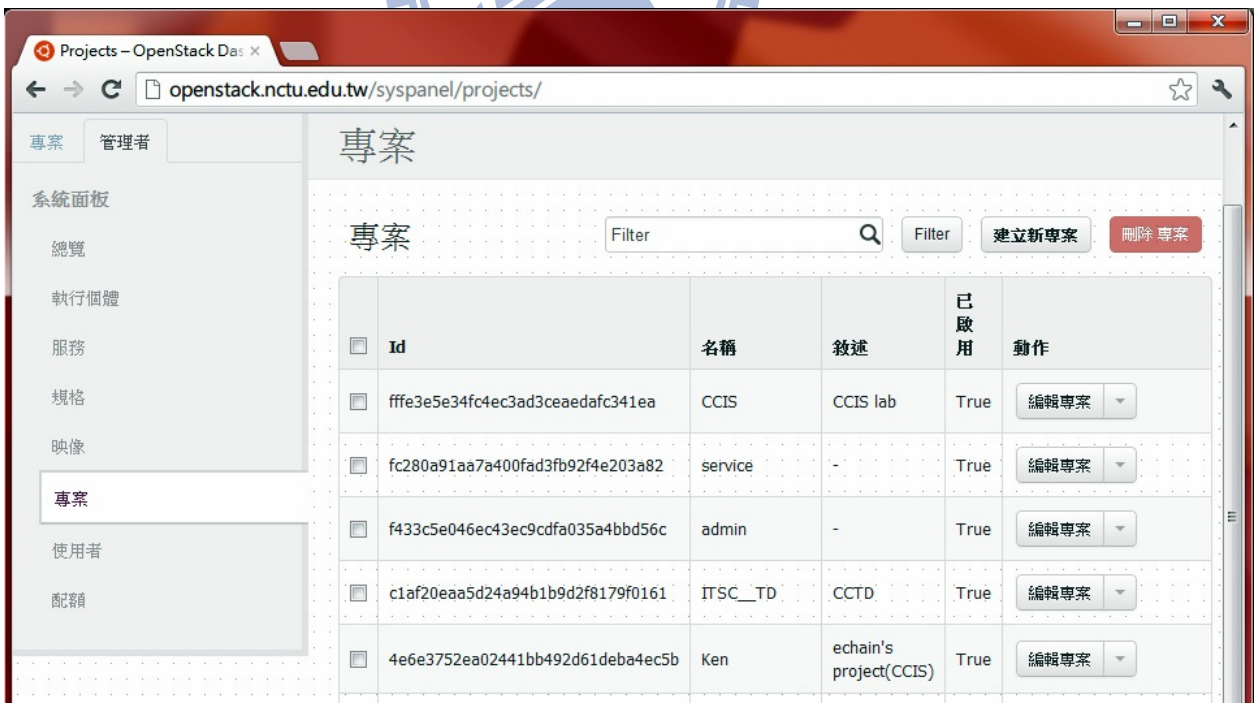

圖 6.12: 管理者：專案列表

圖 6.13: 管理者：屬於 CCIS 專案的使用者列表

管理者可以使用系統面板來對認證系統— Keystone 進行操作，他們可以建立使用者、專案，以及指定使用者要加入哪個專案和用甚麼身分加入 (圖 6.12、圖 6.13)，有了這些功能我們對一些基本的帳號權限操作可以不用使用下指令的方法操作，如此就可以把帳號管理跟系統管理的人員區分開來，減輕系統管理人員的負擔。

6.3 設定

登入 Dashboard 之後，在畫面的右上角有個設定連結供使用者使用。使用者設定以及 OpenStack/EC2 兩種憑證資料下載是原本 Dashboard 內建的，使用者設定僅提供使用者修改顯示語言，因此我們幫 Dashboard 增加了修改使用者資料的功能 (圖 6.14)。目前提供使用者可以自行修改自己的名稱與電子郵件資料，這兩個欄位可於修改後直接點選更新，但若使用者想修改密碼，我們設計為必須先輸入正確的舊密碼才能定義新密碼，避免人為的疏失。

6.3.1 憑證使用方法

OpenStack 在設計之初即考慮相容於舊有的 Amazon AWS EC2 協定，如此雲端運算服務的使用者就可以沿用以往控制指令 (如 euca2ools) 的使用習慣，甚至是重複使用以往針對 EC2 API 所撰寫的程式來控制 OpenStack，減少轉換成本，增加使用意願。

因此為了讓使用者更自由的使用我們的系統，我們將 EC2/NOVA API 和 Keystone

圖 6.14: 設定：更新使用者資料與密碼

圖 6.15: 設定：下載 OpenStack RC 檔

admin API 改成對外開放，以下介紹在 Ubuntu 的環境下使用 OpenStack 憑證 (利用 NOVA API) 來控制 OpenStack 開啟虛擬機的方法。

1. 先從 Dashboard 的"OpenStack 憑證資料"(圖 6.15)，下載 OpenStack RC 檔 (openrc.sh)。
2. 安裝 nova client。

```
# sudo aptitude install python-novaclient
```

3. RC 檔內會包含 Keystone admin API 的連線資料，由於我們在 catalog 裡都用 hostname 來表示 API 的位置，因此使用者也必須在電腦上增加 DNS 對應關係。

```
# echo "140.113.98.245 controller" | sudo tee -a /etc/hosts
```

4. 載入環境變數

```
# source openrc.sh
```

5. 接下來就可以用 nova 這隻 client 程式來控制開關虛擬機了，首先我們先來確認開啟虛擬機所需的各項資訊。

```
// 查看可開啟的虛擬機規格清單
# nova flavor-list
```

```
+----+--------+-----------+------+-----------+------+-------+-------------+
| ID |  Name  | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
+----+--------+-----------+------+-----------+------+-------+-------------+
| 1  | tiny   | 768       | 0    | 0         |      | 1     | 1.0         |
| 2  | small  | 1024      | 0    | 0         |      | 1     | 1.0         |
| 3  | normal | 2048      | 0    | 40        |      | 2     | 1.0         |
| 4  | large  | 4096      | 0    | 80        |      | 3     | 1.0         |
| 5  | xlarge | 8192      | 0    | 120       |      | 4     | 1.0         |
+----+--------+-----------+------+-----------+------+-------+-------------+
```

```
// 查看可開啟映像檔清單
# nova image-list
```

```
+--------------------------------------+--------------------------+--------+--------
+
|                  ID                  |           Name           | Status | Server |
+--------------------------------------+--------------------------+--------+--------
+
| 23461b5c-124d-4318-b1de-740665be4bab | Ubuntu 12.04 64bit Cloud | ACTIVE |        |
| 4f46c149-66d1-4e72-afd8-14d8ef5e0d6a | Fedora 16 64bit Cloud    | ACTIVE |        |
| bca89768-97fd-425c-abf8-7a5b6ff261bd | Windows XP 32bit sp3     | ACTIVE |        |
| d1a52048-e126-42de-9b65-8cbccf6314f0 | Windows 7 64bit Ent      | ACTIVE |        |
+--------------------------------------+--------------------------+--------+--------
+
```

```
// 查看該專案目前可使用的安全性群組清單
# nova secgroup-list
```

```
+---------+-------------+
|   Name  | Description |
+---------+-------------+
| default | default     |
+---------+-------------+
```

```
# nova secgroup-list-rules default
```

```
+-------------+-----------+---------+----------------+--------------+
| IP Protocol | From Port | To Port |    IP Range    | Source Group |
+-------------+-----------+---------+----------------+--------------+
| tcp         | 22        | 22      | 140.113.0.0/16 |              |
| tcp         | 3389      | 3389    | 0.0.0.0/0      |              |
+-------------+-----------+---------+----------------+--------------+
```

```
// 查看已建立的ssh金鑰清單
# nova keypair-list
```

```
+---------+-------------------------------------------------+
|   Name  |                   Fingerprint                   |
+---------+-------------------------------------------------+
| chuanyu | 32:e7:61:74:93:fb:40:52:e2:07:7c:0a:4c:4a:0a:6e |
+---------+-------------------------------------------------+
```

6. 有了上面的資訊就可以開機了! 我們建立一個名稱是"cmd_run_1" 的虛擬機。

```
# nova boot --flavor 1 --image 23461b5c-124d-4318-b1de-740665be4bab \
    --security_groups default --key_name chuanyu "cmd_run_1"
```

```
+------------------------+--------------------------------------+
|        Property        |                Value                 |
+------------------------+--------------------------------------+
| OS-DCF:diskConfig      | MANUAL                               |
| OS-EXT-STS:power_state | 0                                    |
| OS-EXT-STS:task_state  | scheduling                           |
| OS-EXT-STS:vm_state    | building                             |
| accessIPv4             |                                      |
| accessIPv6             |                                      |
| adminPass              | oxMWrFL2uVq2                         |
| config_drive           |                                      |
| created                | 2012-09-21T09:23:21Z                 |
| flavor                 | tiny                                 |
| hostId                 |                                      |
| id                     | e89f3964-c98e-4fc7-b28b-e871b0b30af5 |
| image                  | Ubuntu 12.04 64bit Cloud             |
| key_name               | chuanyu                              |
| metadata               | {}                                   |
| name                   | cmd_run_1                            |
| progress               | 0                                    |
| status                 | BUILD                                |
| tenant_id              | 2bd91f3a4df64427815f721cd663267a     |
```

```
| updated                | 2012-09-21T09:23:21Z                 |
| user_id                | cc83ed485b014548871128f2316f0f60     |
+------------------------+--------------------------------------+
```

7. 查看建立後虛擬機的狀態。

```
# nova list
```

```
+--------------------------------------+-----------+--------+-------------------------+
|                  ID                  |    Name   | Status |         Networks        |
+--------------------------------------+-----------+--------+-------------------------+
| be30c-9ea0-4b90-9f34-2563caf1080f | cmd_run_1 | ACTIVE |default-net_5=10.0.5.18  |
+--------------------------------------+-----------+--------+-------------------------+
```

等待 Status 轉變為 ACTIVE，即代表虛擬機建立完畢。

第七章　結論

OpenStack 是近年來最受矚目的開放原始碼軟體之一，我們可以利用它來部署自己的雲端運算系統，並且依照需求自行修改使用，提高機器的使用效率，降低管理人員的維護成本；Ceph 是一套相當優秀的分散式檔案系統，提供了相當完整的網路儲存解決方案，已經被收入主流 Linux Kernel，發展潛力與實用價值皆受肯定。我們在交通大學資訊技術服務中心，整合這兩套具有相當規模的系統，並針對虛擬機網路架構提出了新的 PHA 模式，節省公開 IP 使用量，避免了服務暴露的風險，另外也使用 Ceph 取代原本的物件儲存系統 Swift，進一步提供 Amazon S3 相容的服務，發展了 NCTU CStack-一個提供交大師生使用的雲端運算與儲存系統。

雖然 OpenStack 與 Ceph 皆在設計之初即把系統延展能力納入考量，但本系統為了充分掌握兩個系統的運作情形，我們皆使用官方套件來部署安裝以利日後的維護與功能更新，但這樣也導致了日後系統規模逐漸擴大時，設定檔的調整及同步等相關管理成本勢必再度提高，這部分可能得利用 Chef、Puppet 等自動部署工具來管理我們的雲端系統設定檔，或是直接使用這些工具來幫助我們快速擴展運算節點和儲存單元。

此外，在使用雲端運算服務的過程中，OpenStack 還是有許多環節不近完美，如缺乏完整的 Monitor 功能：現階段 OpenStack 雖然可以統計專案的虛擬處理器時數和記憶體使用時數，但尚且無法掌握專案內使用浮動 IP 的時段與連線紀錄，倘若有使用者利用雲端的虛擬機對外進行惡意攻擊，我們無法在事後準確地鎖定可疑的來源，對使用者進行停權等限制。若要解決這個問題，我們可能得利用防火牆本身的日誌功能另外發展一套日誌系統，或是尋求官方的協助，討論可行的解決方案。

當然，高度發展中的系統，可以帶給我們較短的臭蟲修復時間、功能穩定性的增強以及許多新功能的體驗，如最新的 OpenStack Folsom 版本則有超過 100 項的新功能被開發出來，因此如何將我們使用的 Essex 版本無縫升級到新的版本，也是我們必須努力的目標。

參考文獻

[1] Campbell, M. "Hyper-V Versus VMware ESX in 2012.", 2012. Available: http://www.unitrends.com/blog/hyper-v-vmware-es-2012/

[2] Inktank Storage Inc, "Accelerating the gateway with Varnish", 2012. Available: http://ceph.com/wiki/RADOS_Gateway

[3] Inktank Storage Inc, "ADDING/REMOVING OSDS", 2012. Available: http://ceph.com/docs/master/rados/operations/add-or-rm-osds/

[4] Inktank Storage Inc, "Ceph FAQ", 2012. Available: http://ceph.com/wiki/FAQ

[5] Inktank Storage Inc, "rgw: ability to delete users without first emptying and deleting all buckets", 2012 Available: http://tracker.newdream.net/issues/2499

[6] mwjpiero "Install openstack on Ubuntu 12.04", 2012. Available: http:// live-moon.dyndns.org/life/2012/04/install-openstack-on-ubuntu-1204.html

[7] "Networking in Nova", 2011. Available: http:// unchainyourbrain.com/ openstack/ 13-networking-in-nova

[8] OpenStack LLC, "Configuring Migrations", 2012. Available: http://docs.openstack.org/trunk/openstack-compute/admin/content/configuring-migrations.html

[9] OpenStack LLC, "Filter Scheduler", 2012. Available: http://docs.openstack.org/developer/nova/devref/filter_scheduler.html

[10] OpenStack LLC, "Getting Images that Work with OpenStack", 2012. Available: http://wiki.openstack.org/GettingImages

[11] OpenStack LLC, "Getting virtual machine images", 2012. Available: http://docs.openstack.org/trunk/openstack-compute/admin/content/starting-images.html

[12] OpenStack LLC, "Glance Authentication With Keystone", 2012. Available: http://docs.openstack.org/developer/glance/authentication.html

[13] OpenStack LLC, "Initial data for Keystone using python-keystoneclient", 2012. Available: https://github.com/openstack-dev/devstack/blob/master/files/keystone_data.sh

[14] OpenStack LLC, "OpenStack 2012.1 (Essex) Release Notes", 2012. Available: http://wiki.openstack.org/ReleaseNotes/Essex

[15] OpenStack LLC, "OPENSTACK COMPUTE ADMINISTRATION MANUAL", 2012. Available: http://docs.openstack.org/essex/

[16] OpenStack LLC, "OpenStack Open Source Cloud Computing Software", 2012. Available: http://www.openstack.org/

[17] OpenStack LLC, "Q: Why is nova-vncproxy no longer part of nova?", 2012. Available: http://docs.openstack.org/developer/nova/runnova/vncconsole.html

[18] OpenStack LLC, "Storage: objects, blocks, and files", 2012. Available: http://docs.openstack.org/trunk/openstack-compute/install/content/terminology-storage.html

[19] P. Mell, T. Grance, "The NIST Definition of Cloud Computing", 2012. Available: http://www.au.af.mil/au/awc/awcgate/nist/cloud-def-v15.doc

[20] SandyWalsh, "MultiClusterZones", 2011. Available: http://wiki.openstack.org/MultiClusterZones

[21] Sébastien Han, "INTRODUCING CEPH TO OPENSTACK", 2012. Available: http://www.sebastien-han.fr/blog/2012/06/10/introducing-ceph-to-openstack/

[22] TaheriMonfared, A. and M. G. Jaatun "As strong as the weakest link: Handling compromised components in OpenStack", Cloud Computing Technology and Science (CloudCom) IEEE, pp. 189-196, 2011.

[23] Weil, S. A., S. A. Brandt, et al. "Ceph: A scalable, high-performance distributed file system.", Operating Systems Design and Implementation (OSDI), pp. 307-320, 2006.

[24] Weil, S. A., S. A. Brandt, et al. "CRUSH: Controlled, scalable, decentralized placement of replicated data", In Proc. Supercomputing (SC), pp. 122, 2006.

[25] 彭勇, " 什么是 openstack 的 metadata", 2012. Available: http://www.pubyun.com/blog/openstack/什么是 openstack 的 -metadata/

附　錄

附錄 A　Ceph 相關檔案

附錄 A.1: /etc/hosts

```
127.0.0.1 localhost.localdomain localhost

# ccc10
192.168.100.1 controller volume
192.168.100.11 compute-1
192.168.100.12 compute-2
192.168.100.13 compute-3

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# cca9r ceph mon1~3
192.168.10.1 ceph1 mon1
192.168.10.2 ceph2 mon2
192.168.10.5 ceph5 mon5

# cca9r ceph osd,mds
192.168.10.13 ceph13
192.168.10.14 ceph14
192.168.10.15 ceph15
192.168.10.16 ceph16
192.168.10.17 ceph17
192.168.10.18 ceph18
192.168.10.19 ceph19
192.168.10.20 ceph20
192.168.10.21 ceph21
<following omitted>
```

附錄 A.2: /etc/ceph/ceph.conf

```
[global]
   # Enable secure authentication
```

```
  auth supported = cephx
    # For manage ceph cluster
    keyring = /etc/ceph/keyring/keyring.admin
    pid file = /var/run/ceph/$name.pid
  # choose log to syslog
    debug filestore = 20
  log_file = ""
  log_to_syslog = true
  daemonize = true

########################################
[mon]
    mon data = /mnt/mon.$id
  keyring = /mnt/mon.$id/keyring
    mon clock drift allowed = 1
    mon clock drift warn backoff = 30
    osd pool default size = 3

[mds]
    keyring = /etc/ceph/keyring/keyring.$name

[osd]
  osd data = /mnt/osd.$id
  keyring = /mnt/osd.$id/keyring
  # 2 option should set immediatly
  osd journal = /mnt/osd.$id/journal
    osd journal size = 1000

    osd class error timeout = 30
    osd class timeout = 60.0
    osd recovery max active = 3
    filestore commit timeout = 120
    filestore journal parallel = true

[mount /]
  allow = %everyone
########### openstack glance ##############
# needed by fuse --name=client.nova
[client.nova]
    keyring = /etc/nova/nova.keyring

[client.glance]
  keyring = /etc/glance/image.keyring

[client.volume]
  keyring = /etc/nova/volume.keyring

[client.radosgw.gateway]
  host = controller
```

```
 keyring = /etc/apache2/radosgw.gateway.keyring
 rgw socket path = /tmp/.radosgw.gateway.keyring
 rgw cache enabled = 1
 rgw dns name = s3
 rgw swift url = http://s3.nctu.edu.tw:8080
 rgw swift url prefix = swift
 rgw print continue = false
 log file = ""
 syslog = true

########################################
[mon.0]
   host = ceph1
   mon addr = 140.113.98.240:6789
[mon.1]
   host = ceph2
   mon addr = 192.168.10.2:6789
[mon.2]
 host = ceph5
 mon addr = 192.168.10.5:6789

########################################
# ceph origin
[mds.0]
 host = ceph5

[osd.0]
 host = ceph5
 btrfs devs = /dev/md1
########################################

# cca9r
[mds.1]
   host = ceph13
[osd.1]
   host = ceph13
   btrfs devs = /dev/md1

[mds.2]
   host = ceph14
[osd.2]
   host = ceph14
   btrfs devs = /dev/md1

[mds.3]
   host = ceph15
[osd.3]
   host = ceph15
   btrfs devs = /dev/md1
```

```

< following ommited >
```

附錄 A.3: /etc/ceph/crush/crush.nctu.txt

```
# begin crush map

######## devices #######
# rack 1
device 0 osd.0
device 1 osd.1
device 2 osd.2
device 3 osd.3
device 4 osd.4
device 5 osd.5
device 6 osd.6
device 7 osd.7
device 8 osd.8
device 9 osd.9
device 10 osd.10
device 11 osd.11
device 12 osd.12
device 13 osd.13
device 14 osd.14
device 15 osd.15
device 16 osd.16
device 17 osd.17

# types
type 0 device
type 1 rack
type 2 raw
type 3 room
type 4 root

# buckets
rack cca9r { # weight 18.000
  id -2
  alg straw
  hash 0 # rjenkins1
  item osd.0 weight 1.000
  item osd.1 weight 1.000
  item osd.2 weight 1.000
  item osd.3 weight 1.000
  item osd.4 weight 1.000
  item osd.5 weight 1.000
  item osd.6 weight 1.000
  item osd.7 weight 1.000
  item osd.8 weight 1.000
```

```
  item osd.9 weight 1.000
  item osd.10 weight 1.000
  item osd.11 weight 1.000
  item osd.12 weight 1.000
  item osd.13 weight 1.000
  item osd.14 weight 1.000
  item osd.15 weight 1.000
  item osd.16 weight 1.000
  item osd.17 weight 1.000
}

raw cca9 { # weight 18.000
  id -3
  alg straw
  hash 0 # rjenkins1
  item cca9r weight 18.000
}

room cc { # weight 18.000
  id -4
  alg straw
  hash 0
  item cca9 weight 18.000
}


root default { # weight 18.000
  id -1
  alg list
  hash 0
  item cc weight 18.000
}

# rules
rule data {
  ruleset 0
  type replicated
  min_size 2
  max_size 10
  step take default
  step chooseleaf firstn 0 type device
  step emit
}
rule metadata {
  ruleset 1
  type replicated
  min_size 2
  max_size 10
  step take default
```

```
 step chooseleaf firstn 0 type device
 step emit
}
rule rbd {
 ruleset 2
 type replicated
 min_size 2
 max_size 10
 step take default
 step chooseleaf firstn 0 type device
 step emit
}
# end crush map
```

附錄 A.4: /etc/apache2/sites-enabled/rgw.conf

```
Listen 140.113.98.246:8080
FastCgiExternalServer /var/radosgw/radosgw.fcgi -socket /tmp/.radosgw.gateway.keyring
<VirtualHost *:8080>
      ServerName 140.113.98.246
      ServerAlias s3.nctu.edu.tw
      ServerAdmin chuanyu@cs.nctu.edu.tw
      DocumentRoot /var/radosgw

      # s3-compatible
      RewriteEngine On
      # following is important for S3
      RewriteRule ^/(.*) /radosgw.fcgi?%{QUERY_STRING} [E=HTTP_AUTHORIZATION:%{HTTP:
          Authorization},L]

      <IfModule mod_fastcgi.c>
            <Directory /var/radosgw/>
                  Options +ExecCGI
                  AllowOverride All
                  SetHandler fastcgi-script
                  Order allow,deny
                  Allow from 140.113.0.0/16
                  Allow from 140.114.0.0/16
                  Allow from 192.168.0.0/16

                  #Allow from all
                  AuthBasicAuthoritative Off
            </Directory>
      </IfModule>

      AllowEncodedSlashes On
      ErrorLog /var/log/apache2/radosgw.error.log
      CustomLog /var/log/apache2/radosgw.access.log combined
      ServerSignature Off
```

```
</VirtualHost>
```

附錄 A.5: /etc/varnish/default.vcl

```
# This is a basic VCL configuration file for varnish. See the vcl(7)
# man page for details on VCL syntax and semantics.

backend radosgw {
    .host = "140.113.98.246";
    .port = "8080";
    .probe = {
        .url = "/";
        .interval = 5s;
        .timeout = 1s;
        .window = 5;
        .threshold = 3;
    }
}

acl allowclient {
    "140.113.0.0"/16;
    "140.114.0.0"/16;
    "192.168.0.0"/16;
}


sub vcl_recv {
    if (client.ip !~ allowclient) {
        error 403 "Not allow source IP.";
    }

    set req.backend = radosgw;

    unset req.http.X-Forwarded-For;
    set req.http.X-Forwarded-For = client.ip;

    unset req.http.Accept-Encoding;
    unset req.http.Cookie;

    if (!req.backend.healthy) {
        set req.grace = 180s;
    } else {
        set req.grace = 15s;
    }

    if (req.request != "GET" && req.request != "HEAD") {
        return (pipe);
    }

```

```
    if (req.http.Authorization) {
       return (pipe);
    }

    return (lookup);
}

sub vcl_fetch {
    unset beresp.http.Server;
       set beresp.http.Server = "RADOS";

    set beresp.ttl = 1s;
    set beresp.grace = 180s;
    return(deliver);
}

sub vcl_pipe {
    set req.http.connection = "close";
    return (pipe);
}
```

附錄 A.6: mkosd

```
#!/bin/sh
#
# Make osd server
# ref from ceph doc.
#
# master$ mkosd --ip 192.168.100.50
# by chuanyu 10/31/2012

set -e

usage_exit() {
  echo "Usage: $0 [--ip 192.168.100.x] [--mkbtrfs]"
  echo " Setup the server which's hostname can be found in /etc/ceph/ceph.conf"
  echo " to become ceph RADOS"
  exit
}

BINDIR=/usr/bin
LIBDIR=/usr/lib/ceph
ETCDIR=/etc/ceph

. $LIBDIR/ceph_common.sh

CCONF="$BINDIR/ceph-conf"
hostname=`hostname | cut -d . -f 1`
conf=$ETCDIR"/ceph.conf"
```

```
27
28 forceyes=0
29 verbose=0
30 mkbtrfs=0
31 ip=""
32 while [ $# -ge 1 ]; do
33 case $1 in
34    -v )
35       verbose=1;
36       ;;
37  --yes | -y)
38    forceyes=1
39    ;;
40    --mkbtrfs | -b)
41       mkbtrfs=1
42       ;;
43  --ip | -i)
44    [ -z "$2" ] && usage_exit
45    shift
46    ip=$1
47    ;;
48    *)
49       echo unrecognized option \'$1\'
50       usage_exit
51       ;;
52 esac
53 shift
54 done
55
56 if [ -z "$ip" ]; then
57  echo "You must set a osd ip."
58  usage_exit
59 fi
60
61 mkosd_finish=0
62 get_name_list "osd"
63 maxosd=0
64 for name in $what; do
65  type=`echo $name | cut -c 1-3` # e.g. 'osd', if $name is 'osd1'
66  id=`echo $name | cut -c 4- | sed 's/^\\.//'`
67  num=$id
68  name="$type.$id"
69  [ $maxosd -lt $id ] && maxosd=$id
70
71  targethost=`$CCONF -c $conf -n $type.$id host`
72  tip=`egrep "${targethost}( |$)" /etc/hosts | awk '{print $1}'`
73  if [ "$tip" = "$ip" ]; then
74    ssh="ssh root@$ip"
75      get_conf osd_data "" "osd data"
```

```
  get_conf osd_journal "" "osd journal"
  get_conf btrfs_path "$osd_data" "btrfs path" # mount point defaults so osd data
  get_conf btrfs_devs "" "btrfs devs"
get_conf btrfs_opt "rw,noatime" "btrfs options"

  if [ -n "$osd_journal" ] && echo "$btrfs_devs" | grep -q -w "$osd_journal" ; then
    echo "ERROR: osd journal device ($osd_journal) also used by btrfs devs ($btrfs_devs)"
    exit 1
  fi

echo "== Start to mkosd at host : $ip"
echo "osd :   $type.$id"
echo "osd data :  $osd_data"
echo "osd journal : $osd_journal"

echo "== Sync ceph.conf..."
scp -q $conf $ip:$ETCDIR
$ssh "service ceph stop"

echo
echo "== Start to prepare osd fs..."
$ssh "test -d $osd_data || mkdir -p $osd_data"
[ -n "$osd_journal" ] && $ssh "test -d $osd_journal || mkdir -p `dirname $osd_journal`"

   if [ $mkbtrfs -eq 1 ]; then
      echo "btrfs path : $btrfs_path"
      echo "btrfs devs : $btrfs_devs"
      echo "btrfs_opt : $btrfs_opt"
      echo
      echo "== $btrfs_devs will be FORMAT as btrfs,"
      [ "$forceyes" != 1 ] && read -p "** Is below all right? [Y/n] : " ans
      if [ "`echo $ans | grep -i n`" != "" ]; then
         echo "== User stopped."
         exit
      fi
   btrfs_devs_name=`echo $btrfs_devs | cut -d '/' -f 3`
   $ssh "umount $btrfs_path" || true
   $ssh "umount $btrfs_devs" || true
   $ssh "modprobe btrfs" || true
   $ssh "mkfs.btrfs $btrfs_devs"
   $ssh "btrfsctl -a"
   $ssh "mount -t btrfs -o $btrfs_opt $btrfs_devs $btrfs_path"
   $ssh "chown root $btrfs_path"
   $ssh "chmod +w $btrfs_path"
   fi


dir=`mktemp -d -t mkcephfs.XXXXXXXXXX` || exit 1
echo "== temp dir is $dir"
```

```
    echo == Get the monitor map
    trap "rm -rf $dir ; exit" INT TERM EXIT
    ceph mon getmap -o $dir/monmap
    cp $conf $dir/conf

    rdir="/tmp/mkfs.ceph.$$"
    echo "== remote temp dir is $rdir"
    $ssh "mkdir -p $rdir"
    scp -q $dir/* $ip:$rdir

    echo "== run cmd: $BINDIR/ceph-osd -c $conf --monmap $rdir/monmap -i $id --mkfs"
    $ssh "$BINDIR/ceph-osd -c $conf --monmap $rdir/monmap -i $id --mkfs"

    get_conf keyring "$rdir/keyring.$name" "keyring"
    echo "== creating private key for $name keyring $keyring"
       $ssh "$BINDIR/ceph-authtool --create-keyring --gen-key -n $name $keyring"

    echo == collecting $name key
    scp -q $ip:$keyring $dir/key.$name
    $ssh "rm -r $rdir"

    echo == Add OSD auth key
    ceph auth add $name osd 'allow *' mon 'allow rwx' -i $dir/key.$name

    mkosd_finish=1
  fi
done # for

if [ $mkosd_finish -eq 1 ]; then
  echo == make OSD finished.
else
  echo Error. Plz set $ip to $conf first!
  exit
fi
```

附錄 B　OpenStack 相關檔案

附錄 B.1: /etc/keystone/default_catalog.templates

```
# config for TemplatedCatalog, using camelCase because I don't want to do
# translations for keystone compat
catalog.RegionOne.identity.publicURL = http://controller:$(public_port)s/v2.0
catalog.RegionOne.identity.adminURL = http://controller:$(admin_port)s/v2.0
catalog.RegionOne.identity.internalURL = http://controller:$(public_port)s/v2.0
catalog.RegionOne.identity.name = Identity Service

# fake compute service for now to help novaclient tests work
catalog.RegionOne.compute.publicURL = http://controller:$(compute_port)s/v1.1/$(tenant_id)s
catalog.RegionOne.compute.adminURL = http://controller:$(compute_port)s/v1.1/$(tenant_id)s
catalog.RegionOne.compute.internalURL = http://controller:$(compute_port)s/v1.1/$(tenant_id)s
catalog.RegionOne.compute.name = Compute Service

catalog.RegionOne.volume.publicURL = http://volume:8776/v1/$(tenant_id)s
catalog.RegionOne.volume.adminURL = http://volume:8776/v1/$(tenant_id)s
catalog.RegionOne.volume.internalURL = http://volume:8776/v1/$(tenant_id)s
catalog.RegionOne.volume.name = Volume Service

catalog.RegionOne.ec2.publicURL = http://controller:8773/services/Cloud
catalog.RegionOne.ec2.adminURL = http://controller:8773/services/Admin
catalog.RegionOne.ec2.internalURL = http://controller:8773/services/Cloud
catalog.RegionOne.ec2.name = EC2 Service

catalog.RegionOne.image.publicURL = http://volume:9292/v1
catalog.RegionOne.image.adminURL = http://volume:9292/v1
catalog.RegionOne.image.internalURL = http://volume:9292/v1
catalog.RegionOne.image.name = Image Service

catalog.RegionOne.object-store.publicURL = http://s3.nctu.edu.tw/auth
catalog.RegionOne.object-store.adminURL = http://s3.nctu.edu.tw/auth
catalog.RegionOne.object-store.internalURL = http://s3.nctu.edu.tw/auth
catalog.RegionOne.object-store.name = Swift Service
```

附錄 B.2: 可顯示 Keystone 內 Tenant<->User<->Role 的相關對應

```
#!/bin/bash
```

```
#
# List all privilege mappings in Keystone
#
# by Chuanyu 2012.05.14
TENENTS=(`keystone tenant-list | awk '{if ($6~/True/) tok=$2 " " tok } END {print tok}'`)
TENENT_NAMES=(`keystone tenant-list | awk '{if ($6~/True/) tok=$4 " " tok } END {print tok}'`)
USERS=(`keystone user-list | awk '{if ($4~/True/) tok=$2 " " tok} END {print tok}'`)
USER_NAMES=(`keystone user-list | awk '{if ($4~/True/) tok=$8 " " tok} END {print tok}'`)

tlen=${#TENENTS[@]}
ulen=${#USERS[@]}
for ((t=0; t<$tlen; t++)); do
   for ((u=0; u<$ulen; u++)); do
      if [ $u -eq 0 ]; then
         printf "Tenant: %s ================ %s\n" "${TENENTS[$t]}" "${TENENT_NAMES[$t]}"
      fi
      grp=`keystone role-list --tenant ${TENENTS[$t]} --user ${USERS[$u]} | \
         awk '{if ($1~/\|/ && $2 !~ /id/) tok = "\t\tRole: " $4 " => " $2 "\n" tok} END {print
              tok}'`
      if [ -n "$grp" ]; then
         printf "\tUser: %s => %s\n" "${USER_NAMES[$u]}" "${USERS[$u]}"
         printf "%s\n" "$grp"
      fi

   done
done
```


附錄 B.3: /etc/nova/nova.conf

```
[DEFAULT]
# general settings
# static set to private ip
my_ip=192.168.100.11
# need to set to public ip (in HA mode)
# no need to set it on PHA mode
#routing_source_ip=140.113.98.244
verbose=True
debug=True
logdir=/var/log/nova
use_syslog=False

sql_connection=mysql://root:[yournovadbpw]@controller:3306/nova
dhcpbridge_flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
state_path=/var/lib/nova
lock_path=/var/lock/nova
root_helper=sudo nova-rootwrap
auth_strategy=keystone
use_deprecated_auth=false
```

```
21
22 # vlan network settings
23 network_manager=nova.network.manager.VlanManager
24 # depend on server
25 vlan_interface=eth2
26 fixed_range_v4=10.0.0.0/16
27 network_size=256
28
29 firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
30 force_dhcp_release=True
31 # will be overwritten when create netwrok
32 public_interface=br0
33 # patched arg
34 float_cidr=24
35 float_gw_ip=140.113.98.254
36 # set more info to let dnsmasq pass to vm
37 dnsmasq_config_file=/etc/nova/nova-dnsmasq.conf
38 send_arp_for_ha=True
39 # nova-appi-metadata run in each compute worker
40 metadata_host=${my_ip}
41 dmz_cidr=192.168.100.0/24
42 multi_host=True
43
44 # APIs ( controller = 192.168.100.1 )
45 rabbit_host=192.168.100.1
46 cc_host=192.168.100.1
47 s3_host=192.168.100.1
48 s3_dmz=192.168.100.1
49 ec2_host=192.168.100.1
50 ec2_dmz_host=192.168.100.1
51 osapi_host=192.168.100.1
52
53 # controller's settings
54 # all service default is set to listen on 0.0.0.0
55 # set to listen on 192.168.100.1 for security
56 #
57 # these listen on public for "Cloud OS course" (skhuang)
58 ec2_listen=0.0.0.0
59 osapi_volume_listen=0.0.0.0
60 osapi_compute_listen=0.0.0.0
61 osapi_compute_extension=nova.api.openstack.compute.contrib.standard_extensions
62 nova_url=http://192.168.100.1:8774/v1.1/
63 ec2_url=http://192.168.100.1:8773/services/Cloud
64 keystone_ec2_url=http://192.168.100.1:5000/v2.0/ec2tokens
65
66 # web vnc setup
67 novnc_enabled=true
68 # follow use public ip of proxy node
69 novncproxy_base_url=http://140.113.98.245:6080/vnc_auto.html
```

```
# compute node's vnc server listen on private ip
vncserver_proxyclient_address=${my_ip}
# if u want use migration, this should be set 0.0.0.0
#vncserver_listen=${my_ip}
vncserver_listen=0.0.0.0

api_paste_config=/etc/nova/api-paste.ini

# Filter Scheduler with SimpleScheduler's "schedule_create_volume" implementation
scheduler_driver=nova.scheduler.filter_scheduler_new.FilterSchedulerNew
scheduler_available_filters=nova.scheduler.filters.standard_filters
scheduler_available_filters=nova.scheduler.filters.project_host_filter.ProjectHostFilter
scheduler_default_filters=ProjectHostFilter,RamFilter
# 8 core cpu allow instances up to 128
cpu_allocation_ratio=16.0
allow_admin_api=true

# nova-volume use Ceph RBD backend
volume_driver=nova.volume.driver.RBDDriver
rbd_pool=volume
rbd_user=volume
# the libvirt secrets(uuid) for the rbd_user
rbd_secret_uuid=[yourlibvirtuuid]
glance_host=volume
glance_api_servers=volume:9292
image_service=nova.image.glance.GlanceImageService

# compute
libvirt_use_virtio_for_bridges=true
start_guests_on_host_boot=false
resume_guests_state_on_host_boot=true
connection_type=libvirt

# project(tenant) default quota
quota_instances=10
quota_cores=10
quota_ram=10240
# limit number of volumes (not total size)
quota_volumes=10
# limit total size of Ephemeral disk(vdb) & total size of volumes
# summarize then 'independently'
quota_gigabytes=300
quota_floating_ips=1
```

附錄 B.4: patch-compute-nopublicip

```
--- l3.py.orig 2012-09-23 01:03:15.848972389 +0800
+++ l3.py 2012-09-23 01:03:24.000750625 +0800
@@ -102,11 +102,11 @@
```

```
       linux_net.unplug(network_ref)

    def add_floating_ip(self, floating_ip, fixed_ip, l3_interface_id):
- linux_net.bind_floating_ip(floating_ip, l3_interface_id)
+ linux_net.bind_floating_ip(floating_ip, fixed_ip, l3_interface_id)
       linux_net.ensure_floating_forward(floating_ip, fixed_ip)

    def remove_floating_ip(self, floating_ip, fixed_ip, l3_interface_id):
- linux_net.unbind_floating_ip(floating_ip, l3_interface_id)
+ linux_net.unbind_floating_ip(floating_ip, fixed_ip, l3_interface_id)
       linux_net.remove_floating_forward(floating_ip, fixed_ip)

    def add_vpn(self, public_ip, port, private_ip):
--- linux_net.py.orig 2012-09-23 01:06:28.699726434 +0800
+++ linux_net.py 2012-09-23 00:59:39.382861828 +0800
@@ -79,6 +79,12 @@
             default=False,
             help='Use single default gateway. Only first nic of vm will '
                 'get default gateway from dhcp server'),
+ cfg.IntOpt('float_cidr',
+    default=32,
+    help='The cidr is used when bind floating ip to Public IF'),
+ cfg.StrOpt('float_gw_ip',
+    default='$my_ip',
+    help='Gateway ip of floating ip'),
    ]

 FLAGS = flags.FLAGS
@@ -457,23 +463,36 @@
    iptables_manager.apply()


-def bind_floating_ip(floating_ip, device):
- """Bind ip to public interface."""
- _execute('ip', 'addr', 'add', str(floating_ip) + '/32',
+def bind_floating_ip(floating_ip, fixed_ip, device):
+ """Bind ip to public interface,
+ and add source based routing to routing table '200'.
+ """
+ _execute('ip', 'addr', 'add', str(floating_ip) + '/' + str(FLAGS.float_cidr),
          'dev', device,
          run_as_root=True, check_exit_code=[0, 2, 254])
+ _execute('ip', 'rule', 'add', 'from', str(fixed_ip), 'table', '200',
+   run_as_root=True, check_exit_code=[0, 2, 254])
+ _execute('ip', 'route', 'add', 'default', 'via', str(FLAGS.float_gw_ip),
+   'dev', device, 'table', '200',
+   run_as_root=True, check_exit_code=[0, 2, 254])
+ _execute('ip', 'route', 'flush', 'cache',
+   run_as_root=True, check_exit_code=[0, 2, 254])
```

```
    if FLAGS.send_arp_for_ha:
        _execute('arping', '-U', floating_ip,
                 '-A', '-I', device,
                 '-c', 1, run_as_root=True, check_exit_code=False)


-def unbind_floating_ip(floating_ip, device):
- """Unbind a public ip from public interface."""
- _execute('ip', 'addr', 'del', str(floating_ip) + '/32',
+def unbind_floating_ip(floating_ip, fixed_ip, device):
+ """Unbind a public ip from public interface,
+ and del related routing rule and table.
+ """
+ _execute('ip', 'addr', 'del', str(floating_ip) + '/' + str(FLAGS.float_cidr),
           'dev', device,
           run_as_root=True, check_exit_code=[0, 2, 254])
-
+ _execute('ip', 'rule', 'del', 'from', str(fixed_ip), 'table', '200',
+    run_as_root=True, check_exit_code=[0, 2, 254])
+

 def ensure_metadata_ip():
     """Sets up local metadata ip."""
```

附錄 B.5: /usr/lib/python2.7/dist-packages/nova/scheduler/filter_scheduler_new.py

```
# Simple module inherit from filter_scheduler.FilterScheduler,
# and use SimpleScheduler's implement "schedule_create_volume"
#
# by chuanyu 11/12/2012
"""
The FilterScheduler is for creating instances locally.
You can customize this scheduler by specifying your own Host Filters and
Weighing Functions.
"""


from nova import flags
from nova import log as logging

from nova.scheduler import filter_scheduler
from nova import db
from nova import utils
from nova.scheduler import driver
from nova import exception



FLAGS = flags.FLAGS
```

```
LOG = logging.getLogger(__name__)

class FilterSchedulerNew(filter_scheduler.FilterScheduler):
    """ """
    def __init__(self, *args, **kwargs):
        super(FilterSchedulerNew, self).__init__(*args, **kwargs)

    def schedule_create_volume(self, context, volume_id, *_args, **_kwargs):
        """Picks a host that is up and has the fewest volumes."""
        elevated = context.elevated()

        volume_ref = db.volume_get(context, volume_id)
        availability_zone = volume_ref.get('availability_zone')

        zone, host = None, None
        if availability_zone:
            zone, _x, host = availability_zone.partition(':')
        if host and context.is_admin:
            service = db.service_get_by_args(elevated, host, 'nova-volume')
            if not utils.service_is_up(service):
                raise exception.WillNotSchedule(host=host)
            driver.cast_to_volume_host(context, host, 'create_volume',
                    volume_id=volume_id, **_kwargs)
            return None

        results = db.service_get_all_volume_sorted(elevated)
        if zone:
            results = [(service, gigs) for (service, gigs) in results
                       if service['availability_zone'] == zone]
        for result in results:
            (service, volume_gigabytes) = result
            if utils.service_is_up(service) and not service['disabled']:
                driver.cast_to_volume_host(context, service['host'],
                     'create_volume', volume_id=volume_id, **_kwargs)
                return None
        msg = _("Is the appropriate service running?")
        raise exception.NoValidHost(reason=msg)
```

附錄 B.6: /usr/lib/python2.7/dist-packages/nova/scheduler/filters/project_host_filter.py

```
# Use this map indicates that the relation between hosts and the projects.
#
# by chuanyu 11/09/2012


from nova.scheduler import filters
from nova.api.openstack.compute.contrib.aggregates import AggregateController as Aggr
from nova import log as logging
LOG = logging.getLogger(__name__)
```

```
10
11
12 #FIXME this map should be placed to configure file (eg. nova.conf)
13 PHMap = {
14    # when no project matches, choose from following hosts
15    'Default': {
16       'compute-1',
17       'compute-2',
18       'compute-3',
19    },
20    # specify project(tenant) id and dedicated hosts
21    '2bd91f3a4df64427815f721cd663267a': { # pacas
22       'compute-1',
23    },
24 }
25
26 class ProjectHostFilter(filters.BaseHostFilter):
27    """Filters Hosts by Project-Host Mapping."""
28
29    def host_passes(self, host_state, filter_properties):
30       spec = filter_properties.get('request_spec', {})
31       props = spec.get('instance_properties', {})
32       project_id = props.get('project_id')
33
34       if project_id in PHMap:
35          return host_state.service['host'] in PHMap[project_id]
36       return host_state.service['host'] in PHMap['Default']
```

附錄 C　Patches & rgwauthAPI

附錄 C.1: rgwauthAPI.py

```
# vim: tabstop=4 shiftwidth=4 softtabstop=4

# Radosgw authication and administration API.
#
# Runtime depend:
# python-swift - Swift client tools
# radosgw - REST gateway for RADOS distributed object store
# python-cloudfiles - Python language bindings for Cloud Files API
#
# Chuanyu Tsai 2012/07/05


from swift.common import client as swiftClient
import horizon
import cloudfiles
import subprocess

# Default ceph user of radosgw
# Remember add `r' perm to keyring with apache2 user (eg. www-data)
cephuser = 'client.radosgw.gateway'

class RadosGW(object):
   """ """
   def __init__(self, uid, subuser=None, authUrl='http://localhost/auth'):
      self.authUrl, self.uid, self.subuser = authUrl, uid, subuser
      self.accessKey, self.secretKey = None, None
      self._checkRGWInstall()

   def _checkRGWInstall(self):
      return
# raise Exception('not installed')

   def _rgwadmin(self, cmd):
      stdRet = subprocess.Popen(args='radosgw-admin -n %s %s' % (cephuser, cmd),
         shell=True,
         stdout=subprocess.PIPE)
```

```
37      return stdRet.stdout.read()
38
39  def _userInfo(self):
40      user = self._rgwadmin('user info --uid="%s"' % self.uid)
41      return eval(user)
42
43  def _authSwift(self):
44      # radosgw only support auth version 1.0
45      return swiftClient.get_auth(self.authUrl, "%s:%s" % (self.uid, self.subuser)
46            , self.secretKey.replace('\\',''), auth_version="1.0")
47
48  def _authS3(self):
49      """ Not implement yet """
50      return
51
52  def _userCreate(self, keyType='swift'):
53      user = self._rgwadmin('user create --uid="%s" --display-name="%s" --email="web@site"'
54            % (self.uid, self.uid))
55      return eval(user)
56
57  def _subuserCreate(self, access='full'):
58      # subuser is only for swift api
59      self._rgwadmin('subuser create --subuser="%s:%s"'
60            % (self.uid, self.subuser))
61      subuser = self._rgwadmin('key create --subuser="%s:%s" --key-type="swift" --access="%s"'
62            % (self.uid, self.subuser, access))
63      return eval(subuser)
64
65  def authenticate(self, keyType='swift', autoCreate=False):
66      """ Return token of radosgw authentication """
67      try:
68          info = self._userInfo()
69      except:
70          if autoCreate == False:
71              raise Exception('User: %s not found.' % (self.uid))
72          else:
73              try:
74                  info = self._userCreate(keyType)
75              except:
76                  raise Exception('User create failed.')
77
78      # Got the user created in rgw,
79      # then check the subuser exists or not,
80      # if exists, return the token,endpoint
81      if keyType == 'swift':
82          for subuser in info['swift_keys']:
83              tuser = self.uid + ':' + self.subuser
84              if (subuser['user'] == tuser):
85                  self.secretKey = subuser['secret_key']
```

```
                return self._authSwift()
        else: # find S3 access/secret key
            return self._authS3()

        if autoCreate == False:
            raise Exception('Subuser: %s:%s not found.' % (self.uid, self.subuser))
        else:
            try:
                self._subuserCreate()
                return self.authenticate(keyType)
            except:
                raise Exception('Subuser create failed.')

    def rmSubuser(self):
        self._rgwadmin('key rm --uid="%s" --subuser="%s:%s" --key-type="swift"'
              % (self.uid, self.uid, self.subuser))
        self._rgwadmin('subuser rm --uid="%s" --subuser="%s:%s"'
              % (self.uid, self.uid, self.subuser))

    def rmUser(self):
        # By http://tracker.newdream.net/issues/2499
        # & http://tracker.newdream.net/issues/2786
        # rgw now still have to delete objects/buckets manually
        storage_url, auth_token = self.authenticate(autoCreate=True)
        swift_api = cloudfiles.get_connection(
            auth=horizon.api.swift.SwiftAuthentication(storage_url, auth_token) )
        containers = swift_api.get_all_containers()

        for name in containers._names:
            container = swift_api.get_container(name)
            objects = container.get_objects()
            for obj_name in objects._names:
                container.delete_object(obj_name)
            swift_api.delete_container(name)

        self._rgwadmin('user rm --uid="%s"'
              % (self.uid))
```

附錄 C.2: Patches about OpenStack Dashboard

```
diff --git a/horizon/api/nova.py b/horizon/api/nova.py
index 1099f56..b8bb1cc 100644
--- a/horizon/api/nova.py
+++ b/horizon/api/nova.py
@@ -30,6 +30,7 @@ from novaclient.v1_1.security_groups import SecurityGroup as
    NovaSecurityGroup
 from novaclient.v1_1.servers import REBOOT_HARD

 from horizon.api.base import APIResourceWrapper, APIDictWrapper, url_for
```

```
+from horizon.api.swift import swift_get_total_size

 from django.utils.translation import ugettext as _

@@ -410,9 +411,8 @@ def tenant_quota_usages(request):
    flavors = dict([(f.id, f) for f in flavor_list(request)])
    usages = {'instances': {'flavor_fields': [], 'used': len(instances)},
            'cores': {'flavor_fields': ['vcpus'], 'used': 0},
- 'gigabytes': {'used': 0,
- 'flavor_fields': ['disk',
- 'OS-FLV-EXT-DATA:ephemeral']},
+ 'gigabytes': {'used': swift_get_total_size(request)/1024/1024/1024,
+ 'flavor_fields': ['OS-FLV-EXT-DATA:ephemeral']},
            'ram': {'flavor_fields': ['ram'], 'used': 0},
            'floating_ips': {'flavor_fields': [], 'used': len(floating_ips)}}

diff --git a/horizon/api/swift.py b/horizon/api/swift.py
index 9af3aa9..037ce4a 100644
--- a/horizon/api/swift.py
+++ b/horizon/api/swift.py
@@ -27,6 +27,7 @@ from django.utils.translation import ugettext as _
 from horizon import exceptions
 from horizon.api.base import url_for

+from rgwauthAPI import RadosGW as RGW

 LOG = logging.getLogger(__name__)

@@ -43,9 +44,26 @@ class SwiftAuthentication(object):

 def swift_api(request):
    endpoint = url_for(request, 'object-store')
+ auth_token = request.session['token']
+
+ use_radosgw_auth = getattr(settings, 'SWIFT_USE_RADOSGW_AUTH', False)
+ sync_radosgw_auth = getattr(settings, 'SYNC_KEYSTONE_RADOSGW_AUTH', False)
+ if (use_radosgw_auth == True):
+ """ Same tenant's users shares same bucket """
+ uid,subuser = request.session['tenant_id'],request.session['user_id']
+
+ LOG.debug('Get radosgw auth by user: %s:%s and endpoint %s'
+    % (uid, subuser, endpoint))
+ try:
+ rgw = RGW(uid, subuser, endpoint)
+ storage_url, auth_token = rgw.authenticate('swift', sync_radosgw_auth)
+ except Exception as e:
+ # rgwauthAPI now puts error msg in args[0]
+ raise exceptions.RadosgwExceptions(str(e.args[0]))
+
```

```
    LOG.debug('Swift connection created using token "%s" and url "%s"'
- % (request.session['token'], endpoint))
- auth = SwiftAuthentication(endpoint, request.session['token'])
+ % (auth_token, storage_url))
+ auth = SwiftAuthentication(storage_url, auth_token)
    return cloudfiles.get_connection(auth=auth)


@@ -76,6 +94,15 @@ def swift_get_containers(request, marker=None):
    else:
        return (containers, False)

+def swift_get_total_size(request):
+ size=0
+ try:
+ containers = swift_api(request).get_all_containers(10000,None)
+ except Exception:
+ containers = None
+ for container in containers:
+ size += container.size_used
+ return size

 def swift_create_container(request, name):
    if swift_container_exists(request, name):
@@ -86,6 +113,11 @@ def swift_create_container(request, name):
 def swift_delete_container(request, name):
    swift_api(request).delete_container(name)

+def swift_filter_objects(s):
+ return s[u'name'].find('/') == -1
+
+def swift_filter_names(s):
+ return s.find('/') == -1

 def swift_get_objects(request, container_name, prefix=None, marker=None):
    limit = getattr(settings, 'API_RESULT_LIMIT', 1000)
@@ -93,6 +125,12 @@ def swift_get_objects(request, container_name, prefix=None, marker=None):
    objects = container.get_objects(prefix=prefix,
                                    marker=marker,
                                    limit=limit + 1)
+ # This branch doesn't support to show objects which have
+ # '/' in their names now (object is a directory or files in directorys)
+ # but ceph rados supports that, so the objects may be upload to
+ # container by s3 api. Therefore, just filter out objects like that.
+ objects._objects = filter(swift_filter_objects, objects._objects)
+ objects._names = filter(swift_filter_names, objects._names)
    if(len(objects) > limit):
        return (objects[0:-1], True)
    else:
```

```
@@ -124,7 +162,15 @@ def swift_copy_object(request, orig_container_name, orig_object_name,

 def swift_upload_object(request, container_name, object_name, object_file):
     container = swift_api(request).get_container(container_name)
- obj = container.create_object(object_name)
+ # radosgw needs to know the size exactly
+ obj = cloudfiles.storage_object.Object(container, object_name,
+ object_record = {
+ 'name': object_name,
+ 'content_type': object_file.content_type,
+ 'bytes': object_file._size,
+ 'last_modified': None,
+ 'hash': None
+ })
     obj.send(object_file)
     return obj

diff --git a/horizon/dashboards/nova/containers/forms.py b/horizon/dashboards/nova/containers/
    forms.py
index 1aeba8c..90e074e 100644
--- a/horizon/dashboards/nova/containers/forms.py
+++ b/horizon/dashboards/nova/containers/forms.py
@@ -30,6 +30,8 @@ from horizon import api
 from horizon import exceptions
 from horizon import forms

+from django.conf import settings
+from rgwauthAPI import RadosGW as RGW

 LOG = logging.getLogger(__name__)

@@ -54,6 +56,31 @@ class CreateContainer(forms.SelfHandlingForm):
       return shortcuts.redirect("horizon:nova:containers:index")


+class ShowKeys(forms.SelfHandlingForm):
+ def __init__(self, *args, **kwargs):
+ super(ShowKeys, self).__init__(*args, **kwargs)
+ endpoint = kwargs.get('initial', {}).get('endpoint', [])
+ tenant_id = kwargs.get('initial', {}).get('tenant_id', [])
+
+ accessPoint = getattr(settings, 'RADOSGW_S3_ACCESS_POINT', endpoint)
+ user = RGW(tenant_id, None, endpoint)._userInfo()
+
+ helpStr = "Ctrl+C Copy to Clipboard"
+ self.fields['apoint'] = forms.CharField(
+ initial=accessPoint, label='Access Point', help_text=helpStr)
+ self.fields['access'] = forms.CharField(
+ initial=user["keys"][0]["access_key"],
```

```
+ label='Access Key ID', help_text=helpStr)
+ self.fields['secret'] = forms.CharField(
+ initial=user["keys"][0]["secret_key"].replace('\\',''),
+ label='Secret key', help_text=helpStr)
+
+ self.fields['apoint'].widget.attrs['onClick'] = \
+ self.fields['access'].widget.attrs['onClick'] = \
+ self.fields['secret'].widget.attrs['onClick'] = \
+ 'Javascript:this.focus();this.select();'
+
+
 class UploadObject(forms.SelfHandlingForm):
    name = forms.CharField(max_length="255",
                       label=_("Object Name"),
diff --git a/horizon/dashboards/nova/containers/tables.py b/horizon/dashboards/nova/containers
    /tables.py
index 27b915e..fd04920 100644
--- a/horizon/dashboards/nova/containers/tables.py
+++ b/horizon/dashboards/nova/containers/tables.py
@@ -27,6 +27,7 @@ from django.utils.translation import ugettext_lazy as _
 from horizon import api
 from horizon import tables

+from django.conf import settings

 LOG = logging.getLogger(__name__)

@@ -93,6 +94,12 @@ class UploadObject(tables.LinkAction):
       # styles meant for the table action version.
       self.attrs = {'class': 'ajax-modal'}

+class ShowKeys(tables.LinkAction):
+ name = "show_keys"
+ verbose_name = _("Show Keys")
+ url = "horizon:nova:containers:show_keys"
+ classes = ("ajax-modal", "btn-create")
+

 def get_size_used(container):
    return filesizeformat(container.size_used)
@@ -112,7 +119,10 @@ class ContainersTable(tables.DataTable):
    class Meta:
       name = "containers"
       verbose_name = _("Containers")
- table_actions = (CreateContainer, DeleteContainer)
+ if getattr(settings, 'SWIFT_SHOW_RADOSGW_KEYS', False):
+ table_actions = (ShowKeys, CreateContainer, DeleteContainer)
+ else:
+ table_actions = (CreateContainer, DeleteContainer)
```

```
203         row_actions = (ListObjects, UploadObject, DeleteContainer)
204
205
206 diff --git a/horizon/dashboards/nova/containers/urls.py b/horizon/dashboards/nova/containers/
        urls.py
207 index 9ba2159..ec1bdc9 100644
208 --- a/horizon/dashboards/nova/containers/urls.py
209 +++ b/horizon/dashboards/nova/containers/urls.py
210 @@ -20,7 +20,7 @@
211
212  from django.conf.urls.defaults import patterns, url
213
214 -from .views import IndexView, CreateView, UploadView, ObjectIndexView, CopyView
215 +from .views import IndexView, ShowKeysView, CreateView, UploadView, ObjectIndexView, CopyView
216
217
218  OBJECTS = r'^(?P<container_name>[^/]+)/%s$'
219 @@ -30,6 +30,7 @@ OBJECTS = r'^(?P<container_name>[^/]+)/%s$'
220  urlpatterns = patterns('horizon.dashboards.nova.containers.views',
221     url(r'^$', IndexView.as_view(), name='index'),
222     url(r'^create/$', CreateView.as_view(), name='create'),
223 + url(r'^show/$', ShowKeysView.as_view(), name='show_keys'),
224     url(OBJECTS % r'$', ObjectIndexView.as_view(), name='object_index'),
225     url(OBJECTS % r'upload$', UploadView.as_view(), name='object_upload'),
226     url(OBJECTS % r'(?P<object_name>[^/]+)/copy$',
227 diff --git a/horizon/dashboards/nova/containers/views.py b/horizon/dashboards/nova/containers/
        views.py
228 index c872031..4721030 100644
229 --- a/horizon/dashboards/nova/containers/views.py
230 +++ b/horizon/dashboards/nova/containers/views.py
231 @@ -32,9 +32,10 @@ from horizon import api
232  from horizon import exceptions
233  from horizon import forms
234  from horizon import tables
235 -from .forms import CreateContainer, UploadObject, CopyObject
236 +from .forms import CreateContainer, ShowKeys, UploadObject, CopyObject
237  from .tables import ContainersTable, ObjectsTable
238
239 +from horizon.api.base import url_for
240
241  LOG = logging.getLogger(__name__)
242
243 @@ -59,6 +60,15 @@ class IndexView(tables.DataTableView):
244         return containers
245
246
247 +class ShowKeysView(forms.ModalFormView):
248 + form_class = ShowKeys
249 + template_name = 'nova/containers/showkeys.html'
```

```
+
+ def get_initial(self):
+ return {'endpoint': url_for(self.request, 'object-store'),
+ 'tenant_id': self.request.session['tenant_id']}
+
+
 class CreateView(forms.ModalFormView):
    form_class = CreateContainer
    template_name = 'nova/containers/create.html'
diff --git a/horizon/dashboards/nova/images_and_snapshots/images/views.py b/horizon/dashboards
    /nova/images_and_snapshots/images/views.py
index 1803f97..f9fa586 100644
--- a/horizon/dashboards/nova/images_and_snapshots/images/views.py
+++ b/horizon/dashboards/nova/images_and_snapshots/images/views.py
@@ -64,6 +64,11 @@ class LaunchView(forms.ModalFormView):
       context = super(LaunchView, self).get_context_data(**kwargs)
       try:
          context['usages'] = api.tenant_quota_usages(self.request)
+ for usage in context['usages']:
+ if usage != "floating_ips" and context['usages'][usage]['available'] <= 0:
+ context['usages']['disabled'] = 'disabled=disabled'
+ break
+ context['usages'].setdefault('disabled', '')
       except:
          exceptions.handle(self.request)
       return context
@@ -78,7 +83,7 @@ class LaunchView(forms.ModalFormView):
          flavors = api.flavor_list(self.request)
          flavor_list = [(flavor.id, display % {"name": flavor.name,
                                     "vcpus": flavor.vcpus,
- "disk": flavor.disk,
+ "disk": getattr(flavor, "OS-FLV-EXT-DATA:ephemeral"),
                                     "ram": flavor.ram})
                                    for flavor in flavors]
       except:
diff --git a/horizon/dashboards/nova/instances_and_volumes/instances/tables.py b/horizon/
    dashboards/nova/instances_and_volumes/instances/tables.py
index 82c0a18..c1618f7 100644
--- a/horizon/dashboards/nova/instances_and_volumes/instances/tables.py
+++ b/horizon/dashboards/nova/instances_and_volumes/instances/tables.py
@@ -205,10 +205,11 @@ def get_ips(instance):

 def get_size(instance):
    if hasattr(instance, "full_flavor"):
- size_string = _("%(RAM)s RAM | %(VCPU)s VCPU | %(disk)s Disk")
+ size_string = _("%(RAM)s RAM | %(VCPU)s VCPU | %(disk)s Disk(vdb)")
       vals = {'RAM': sizeformat.mbformat(instance.full_flavor.ram),
             'VCPU': instance.full_flavor.vcpus,
- 'disk': sizeformat.diskgbformat(instance.full_flavor.disk)}
```

```
+ 'disk': sizeformat.diskgbformat(
+ getattr(instance.full_flavor, 'OS-FLV-EXT-DATA:ephemeral'))}
       return size_string % vals
    return _("Not available")

diff --git a/horizon/dashboards/nova/templates/nova/containers/_showkeys.html b/horizon/
    dashboards/nova/templates/nova/containers/_showkeys.html
new file mode 100644
index 0000000..c3f2b23
--- /dev/null
+++ b/horizon/dashboards/nova/templates/nova/containers/_showkeys.html
@@ -0,0 +1,22 @@
+{% extends "horizon/common/_modal_form.html" %}
+{% load i18n %}
+
+{% block form_id %}show_keys_form{% endblock %}
+
+{% block modal-header %}{% trans "Show Keys" %}{% endblock %}
+
+{% block modal-body %}
+<div class="left">
+ <fieldset>
+ {% include "horizon/common/_form_fields.html" %}
+ </fieldset>
+</div>
+<div class="right">
+ <h3>{% trans "Description" %}:</h3>
+ <p>{% trans "We also support the Amazon &reg; AWS S3 compatible service. If you want to try
    a fresh, the Access Key & Secret Key are shown in the left side. Moreover, you can also
    mount a bucket as local drive by Gladinet &reg; or other clients use S3 API. Notice!
    Dashboard still *NOT* support directory in bucket, so you can't see the files in directory
     which is uploaded from S3 API." %}</p>
+</div>
+{% endblock %}
+
+{% block modal-footer %}
+ <a href="{% url horizon:nova:containers:index %}" class="btn btn-primary pull-right">{%
    trans "Back" %}</a>
+{% endblock %}
diff --git a/horizon/dashboards/nova/templates/nova/containers/showkeys.html b/horizon/
    dashboards/nova/templates/nova/containers/showkeys.html
new file mode 100644
index 0000000..fc63e11
--- /dev/null
+++ b/horizon/dashboards/nova/templates/nova/containers/showkeys.html
@@ -0,0 +1,11 @@
+{% extends 'nova/base.html' %}
+{% load i18n %}
+{% block title %}Create Container{% endblock %}
```

```
+
+{% block page_header %}
+ {% include "horizon/common/_page_header.html" with title=_("Create Container") %}
+{% endblock page_header %}
+
+{% block dash_main %}
+ {% include "nova/containers/_showkeys.html" %}
+{% endblock %}
diff --git a/horizon/dashboards/nova/templates/nova/images_and_snapshots/images/_launch.html b
    /horizon/dashboards/nova/templates/nova/images_and_snapshots/images/_launch.html
index 0b1c0f5..04d339d 100644
--- a/horizon/dashboards/nova/templates/nova/images_and_snapshots/images/_launch.html
+++ b/horizon/dashboards/nova/templates/nova/images_and_snapshots/images/_launch.html
@@ -34,7 +34,7 @@
   <div class="quota_bar">{% horizon_progress_bar usages.cores.used usages.cores.quota %}</div>

   <div class="quota_title">
- <strong>{% trans "Disk" %} <span>({{ usages.gigabytes.used }} {% trans "GB" %})</span></
    strong>
+ <strong>{% trans "Disk" %}+{% trans "Containers" %} <span>({{ usages.gigabytes.used }} {%
    trans "GB" %})</span></strong>
    <p>{{ usages.gigabytes.available|quota:"GB" }}</p>
   </div>
   <div class="clearfix"></div>
@@ -50,6 +50,6 @@
 {% endblock %}

 {% block modal-footer %}
- <input class="btn btn-primary pull-right" type="submit" value="{% trans "Launch Instance" %}
    " />
+ <input class="btn btn-primary pull-right" {{ usages.disabled }} type="submit" value="{%
    trans "Launch Instance" %}" />
   <a href="{% url horizon:nova:images_and_snapshots:index %}" class="btn secondary cancel
       close">{% trans "Cancel" %}</a>
 {% endblock %}
diff --git a/horizon/dashboards/syspanel/projects/tables.py b/horizon/dashboards/syspanel/
    projects/tables.py
index 3b6b81f..604315b 100644
--- a/horizon/dashboards/syspanel/projects/tables.py
+++ b/horizon/dashboards/syspanel/projects/tables.py
@@ -8,6 +8,10 @@ from horizon import tables

 from ..users.tables import UsersTable

+from django.conf import settings
+from horizon.api.base import url_for
+from horizon import exceptions
+from rgwauthAPI import RadosGW as RGW

```

```
 LOG = logging.getLogger(__name__)

@@ -52,6 +56,18 @@ class DeleteTenantsAction(tables.DeleteAction):
     data_type_plural = _("Projects")

     def delete(self, request, obj_id):
+ use_radosgw_auth = getattr(settings, 'SWIFT_USE_RADOSGW_AUTH', False)
+ sync_radosgw_auth = getattr(settings, 'SYNC_KEYSTONE_RADOSGW_AUTH', False)
+ if use_radosgw_auth and sync_radosgw_auth:
+ endpoint = url_for(request, 'object-store')
+ try:
+ LOG.debug('Radosgw remove user %s from endpoint: %s'
+ % (obj_id, endpoint))
+ RGW(obj_id, 'admin', authUrl=endpoint).rmUser()
+ except Exception as e:
+ exceptions.handle(request, _("Unable to delete rgw user."))
+ for user in api.keystone.user_list(request, obj_id):
+ api.keystone.remove_tenant_user(request, obj_id, user.id)
         api.keystone.tenant_delete(request, obj_id)


@@ -94,9 +110,20 @@ class RemoveUserAction(tables.BatchAction):

     def action(self, request, user_id):
         tenant_id = self.table.kwargs['tenant_id']
+ use_radosgw_auth = getattr(settings, 'SWIFT_USE_RADOSGW_AUTH', False)
+ sync_radosgw_auth = getattr(settings, 'SYNC_KEYSTONE_RADOSGW_AUTH', False)
+ if use_radosgw_auth and sync_radosgw_auth:
+ endpoint = url_for(request, 'object-store')
+ uid, subuser = tenant_id, user_id
+ try:
+ LOG.debug('Radosgw remove subuser %s:%s from endpoint: %s'
+ % (uid, subuser, endpoint))
+ RGW(uid, subuser, endpoint).rmSubuser()
+ except:
+ redirect = reverse("horizon:syspanel:projects:users", args=(tenant_id,))
+ exceptions.handle(request, _('Remove user failed.'), redirect=redirect)
         api.keystone.remove_tenant_user(request, tenant_id, user_id)

-
 class TenantUsersTable(UsersTable):
     class Meta:
         name = "tenant_users"
diff --git a/horizon/exceptions.py b/horizon/exceptions.py
index cf460e3..d6a0416 100644
--- a/horizon/exceptions.py
+++ b/horizon/exceptions.py
@@ -109,6 +109,15 @@ class AlreadyExists(HorizonException):
     def __unicode__(self):
```

```
430         return _(self.msg) % self.attrs
431
432 +class RadosgwExceptions(HorizonException):
433 + def __init__(self, msg):
434 + self.msg = msg
435 +
436 + def __repr__(self):
437 + return self.msg
438 +
439 + def __unicode__(self):
440 + return _(self.msg)
441
442  class HandledException(HorizonException):
443     """
444 @@ -148,7 +157,8 @@ RECOVERABLE = (keystoneclient.ClientException,
445             novaclient.ClientException,
446             glanceclient.GlanceException,
447             swiftclient.Error,
448 - AlreadyExists)
449 + AlreadyExists,
450 + RadosgwExceptions)
451  RECOVERABLE += tuple(EXCEPTION_CONFIG.get('recoverable', []))
452
453
454 diff --git a/horizon/locale/zh_TW/LC_MESSAGES/django.mo b/horizon/locale/zh_TW/LC_MESSAGES/
        django.mo
455 index 3227f1c..7ccc5ce 100644
456 Binary files a/horizon/locale/zh_TW/LC_MESSAGES/django.mo and b/horizon/locale/zh_TW/
        LC_MESSAGES/django.mo differ
457 diff --git a/horizon/locale/zh_TW/LC_MESSAGES/django.po b/horizon/locale/zh_TW/LC_MESSAGES/
        django.po
458 index 3f51bcf..272d387 100644
459 --- a/horizon/locale/zh_TW/LC_MESSAGES/django.po
460 +++ b/horizon/locale/zh_TW/LC_MESSAGES/django.po
461 @@ -1248,6 +1248,20 @@ msgstr ""
462  "不能有其他容器。但是，您可以在您的帳戶建立無限個容器。資料必須儲存在容器裡，"
463  "因此您必須在帳戶內建立至少一個容器，才能上傳資料。"
464
465 +#: dashboards/nova/templates/nova/containers/_showkeys.html:16
466 +msgid ""
467 +"We also support the Amazon &reg; AWS S3 compatible service. If you want to "
468 +"try a fresh, the Access Key & Secret Key are shown in the left side. Moreover, "
469 +"you can also mount a bucket as local drive by Gladinet &reg; or other clients "
470 +"use S3 API. Notice! Dashboard still *NOT* support directory in bucket, so "
471 +"you can't see the files in directory which is uploaded from S3 API."
472 +msgstr ""
473 +"我們也提供了相容於 Amazon &reg; AWS S3 的服務。如果您想試試，左邊提供了必要的"
474 +"Access Key & Secret 。Key 而且，您還可以試著用像Gladinet &reg; 這樣支援S3 "
475 +"API 的軟體把專案裡的bucket 掛載到本地端的機器上當作一顆磁碟。請注意! 目前為"
```

```
+"止，Dashboard 還不支援顯示** bucket 裡的資料夾，所以如果您使用S3 的協定上傳資料"
+"夾，將不會顯示在 dashboard 上。"
+
 #: dashboards/nova/templates/nova/images_and_snapshots/index.html:3
 #: dashboards/nova/templates/nova/images_and_snapshots/index.html:6
 msgid "Images &amp;amp; Snapshots"
```