以 OpenStack 建立可擴充式物聯網系統雲端平台

蘇俊憲　　陳昌盛
國立交通大學 資訊技術服務中心
chsu@mail.nctu.edu.tw　cschen@mail.nctu.edu.tw

摘要

本論文旨在探討運用 OpenMTC 物聯網平台軟體結合 OpenStack 以建置可自動擴展雲端運算平台之相關課題。本論文以 OpenStack 為 IaaS 基礎架構，整合 fokus 公司的物聯網系統軟體 OpenMTC，利用 OpenStack API 指令，控制虛擬機的開關機以及 Load Balancer 成員的加入與刪除，達成自動擴展(auto scaling)物聯網服務的目的。由於在本研究進行期間，本實驗系統尚未有大量物聯網感測元件運作，因此本論文使用 Jmeter 測試軟體模擬多個連線同時存取所建置的物聯網系統，加以測試其效能。透過效能的模擬測試，可以得知當 NSCL(Network Service Capability Layer)的個數增加時，平均回應時間可以有效地下降，提升系統的效率，尤其是在同時有許多連線數的情況之下，更能夠顯著地提升效能，降低平均反應的時間。

關鍵詞：物聯網、OpenStack、OpenMTC、負載平衡、M2M、雲端運算。

Abstract

This paper aimed to explore the issues on developing an auto-scalable M2M service on the NCTU OpenStack private cloud (with OpenStack' load balancer module, LBasS, installed) by using the OpenMTC software platform. For performance evaluation, we adopted the Apache JMeter benchmark program for conducting the stress test. From the simulation results, we could find that the average response time could be greatly reduced as the number of NSCLs (Network Service Capability Layer) had been increased (i.e., changed from one to eight) gradually.

Keywords: JMeter, LBaSS, M2M, OpenMTC, OpenStack, Private cloud.

* 本論文由國科會計劃(編號 NSC NSC102-2218-E-009-002)提供部分補助，僅此致謝。

1. 物聯網簡介與研究動機

隨著網際網路與行動裝置的普及與發達，雲端運算與物聯網(Internet of Thing；IOT)結合的議題是未來研究的重要課題。2005 年，國際電信聯盟((International Telecommunication Union；ITU)發布了網際網路年度報告《ITU 網際網路報告 2005：物聯網》，報告中具體描述物聯網的概念，資訊與通訊技術的適用範圍已經從任何地點、任何時間連接任何人，演變成連接任何物品的階段，所有的物品透過網路相連，就形成了物聯網[1][2]。

物聯網擁有廣泛發展空間，尤其是在交通、環境、政府機構、全球安防、居家安全、環保節能、運輸和物流領域、健康醫療領域、智能環境（家庭、辦公、工廠）領域，物聯網中所有的感應裝置收集數據資料後，皆透過網路傳回數據中心，數據中心再提供資料掘取分析的平台給各個應用使用。每個人周圍的設備可以達到一千至五千個，所以物聯網可能要包含 500 萬億至一千萬億個物體[1]，因此隨之而來的挑戰為將會有成千上萬以上的裝置從各地不斷地回傳感測資料回資料中心，資料中心內部的物聯網系統將會承受大量的網路流量，大量的網路流量流入將會造成系統 CPU 使用率過高、連線數過高而導致系統失效。

基於以上所遇到的挑戰，本研究擬將物聯網系統導入雲端運算的概念，善用雲端運算具有彈性、可擴展的特性來建立物聯網的雲端服務。本論文以 OpenStack 建立 IaaS 與 PaaS 雲端運算私有雲平台，並使用德國 fokus 公司[3]所開發的 OpenMTC 物聯網系統軟體，將本研究之物聯網系統架設在 OpenStack 上，並額外加入負載平衡(Load Balancer)功能，結合系統的監控，整合成為一個自動擴展的物聯網雲端服務，藉此解決當系統負載過重時，需人工手動調整系統架構的缺點。

由於在本研究進行期間，本實驗系統尚未有大量物聯網感測元件運作，因此本論文使用 Jmeter 測試軟體模擬多個連線同時存取所建置的物聯網系統，加以測試。透過效能的模擬測試，可以得知當 NSCL 的個數增加時，平均回應時間可以有效地下降，提升系統的效率，尤其是在同時有許多連線數的情況之下，更能夠顯著地提升效能，降低平均反應的時間。

本篇論文後續的內容如下：第 2 節介紹物聯網、雲端運算與 OpenStack 的相關研究，第 3 節介紹 NCTU OpenStack 建置與 OpenMTC 與 OpenStack 之整合，第 4 節為系統建置成果與效能分析，第 5 節是結論與未來工作。

2. 相關研究

2.1 物聯網

圖 1 為歐洲電信化標準協會(European Telecommunications Standards Institute；ETSI)所定義的物聯網標準化架構，圖中文字 M2M 的解釋為 Machine to Machine(機器對機器)，亦即代表物聯網的意思。ETSI 把物聯網分成三個部份，物聯網的系統架構分別為圖 1 左邊的 Device Domain，中間部分稱之為 Network Domain；而左邊部分定義為 Application Domain。

- Device Domain：此 domain 又稱之為物聯網的感知層，感測器感測資料(例如溫濕度、位置、速度、RFID、條碼等資料)，透過有線網路或短距離低速的無線網路(例如 Zigbee/ RFID/ NFC/ Bluetooth/ WiFi)將所收集到的資料透過 Gateway 進行匯集，再經由 Network Domain 進行傳輸。
- Network Domain：此為物聯網的網路層，是 M2M 在通訊中傳輸資料的關鍵，在串連 Device Domain 與 Application Domain 中扮演重要的角色。在此 domain 所運用到的網路傳輸技術為有線網路(如 Ethernet/ Fiber/ Cable /xDSL 等)、無線網路(如 2G/ GPRS/ 3G/ HPSA/ LTE/ WiMAX 等)、衛星等技術。然而在 Network Domain 中，還有一個重要的部分，那就是 service capability，可視為 M2M 應用與網路的整合平台，儲存 Device Domain 所送進來的感測資料，並提供資料給 Application Domain 使用。
- Application Domain：Application Domain 則是代表著不同的 M2M 應用，例如智能交通、居家照護、物流監控、公共安全等等應用，皆是未來物聯網發展是否成功的重要關鍵。訊息收集、分析與管理技術在此層是一門重要的學問，因此最近火紅的巨量資料(Big Data)也是屬於物聯網的研究範圍之內。

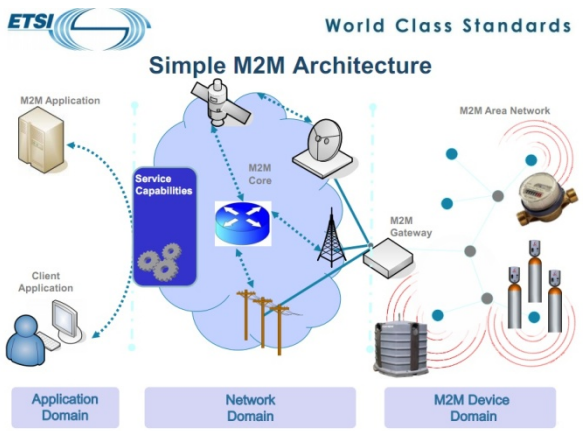

圖 1 ETSI Simple M2M Architecture

2.2 雲端運算

雲端運算是一種依需求，可從網路便利地存取使用設定好的共享運算資源池(例如 CPU、Memory、Storage、Network、Application、Services 等等)，以最少的管理負擔與減少與雲端服務供應商的頻繁互動，來達成快速配置和發佈。如表 1 所示，美國國家技術標準局（National Institute of Standards and Technology；NIST）定義雲端運算五個基礎特徵、三個服務模式及四種佈署模型[4]。

表 1 雲端運算之基礎特徵、服務模式及佈署模型

項目	內容
五大基礎特徵	(1). On-demand self-service (2). Broad network access (3). Resource pooling (4). Rapid elasticity (5). Measured Service
三個服務模式	(1). Software as a Service。 (2). Platform as a Service。 (3). Infrastructure as a Service。
四個佈署模型	(1). Private cloud (2). Community cloud (3). Public cloud (4). Hybrid cloud

本論文的雲端平台以 OpenStack 架設，屬於 Private Cloud 部屬模型，符合以上五大基礎特徵，下一節將會介紹 OpenStack 詳細的內容。

2.3 雲端作業系統 – 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私有雲與儲存環境[5]。OpenStack 內部包括了運算模組、網通模組和儲存模組，如圖 2 所示，再搭配一個可以集中管理上述三大類模組的儀表板模組，最後組合而成一套 OpenStack 共享服務，並且可以提供虛擬機器的方式，對外提供運算資源以便彈性擴充或調度。

應用程式的開發者，只要透過 API 的呼叫就可以和 OpenStack 溝通，例如用 API 來調整虛擬機器的數量、開機或關機等等，OpenStack 內部再負責運算資源的管理還有各元件內部溝通與協調。

目前 OpenStack 一共有 9 個不同的專案套件，9 個不同的專案套件分別提供虛擬化技術、網路、認證、儲存、管理介面、映象檔管理等功能，每個專案皆有專屬的專案名稱，分別說明如下[6]：

I. **Nova**：提供部署、排程與管理虛擬機之功能。
II. **Keystone**：提供不同種類的驗證方式，能查看哪位使用者可存取哪些服務。
III. **Neutron**：2013/6 由 Quantum 改名為 Neutron，管理的 OpenStack 的網路架構，支援多種網路

相關的 Plug-in。

IV. **Cinder**：負責管理區塊儲存容量，具有快照能力。

V. **Glance**：提供映象檔蒐尋、註冊以及服務交付等功能。

VI. **Horizon**：圖形化的網頁管理介面，讓雲端運算使用者管理雲端服務的所分配的資源。

VII. **Swift**：分散式儲存平臺，可存放非結構化的資料，類似於亞馬遜的 S3[7]服務。

VIII. **Heat(開發中)**：編制(Orchestration)的套件整合元件，提供雲端服務流程自動化引擎。

IX. **Ceilometer(開發中)**：計算雲端服務所使用的資料量，作為日後收費參考依據。

OpenStack 版本資訊如圖 3[8]所示，交通大學所使用的版本有 Essex 與 Grizzly，目前正在致力於再 OpenStack 開發新服務，例如本論文的物聯網服務，就是一項在 OpenStack 所使用的應用。未來新版本 Havana 釋出之後，我們將會更新 OpenStack 之版本。

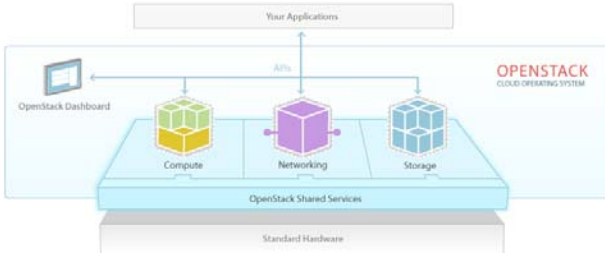

圖 2 OpenStack 模組示意圖

Series	Status	Releases	Date
Havana	Under development	Due	Oct 17, 2013
Grizzly	Current stable release, security-supported	2013.1	Apr 4, 2013
		2013.1.1	May 9, 2013
		2013.1.2	Jun 6, 2013
		2013.1.3	Aug 8, 2013
Folsom	Security-supported	2012.2	Sep 27, 2012
		2012.2.1	Nov 29, 2012
		2012.2.2	Dec 13, 2012
		2012.2.3	Jan 31, 2013
		2012.2.4	Apr 11, 2013
Essex	EOL	2012.1	Apr 5, 2012
		2012.1.1	Jun 22, 2012
		2012.1.2	Aug 10, 2012
		2012.1.3	Oct 12, 2012

圖 3 OpenStack 版本資訊

3. 系統架構說明

3.1 交通大學 OpenStack 系統架構

交通大學資訊技術服務中心於 2013 年 4 月建立新版的 OpenStack Grizzly，並於 Grizzly 上開發新的服務，例如 Hadoop 服務與物聯網雲端平台服務，後續將會持續開發其他之服務。

我們使用了多台伺服器來架設 OpenStack，分別為 1 台 Controller Node、4 台 Compute Node 與 1 台 Network Node，如圖 4 所示。另外儲存方面，使用了分散式檔案系統 Ceph 來負責 Object Storage(類似於亞馬遜 S3)服務，使用 iSCSI 儲存設備存放虛擬機器的檔案與區塊儲存(Cinder)的檔案。OpenStack 實體伺服器規格如表 2 所示，Compute Node 因為需要開啟虛擬機，因此需要規格較好的 CPU 與容量較大的記憶體空間，而 Controller Node 與 Network Node 相較於 Compute Node，所需要的記憶體比較少，CPU 規格也不需跟 Compute Node 一致。

交通大學資訊中心與資工系於 2012 年一起建立版本名稱為 Essex 的 OpenStack，接著於 2013 年 4 月建置了 Grizzly 版本之 OpenStack，提供一個 IaaS 架構的私有雲於交通大學中使用，目前主要使用於研究計畫、雲端運算課程、實驗室以及資訊中心同仁測試使用，提供老師與學生在 OpenStack 平台上開發雲端服務，而本論文就是在 OpenStack 上開發物聯網雲端服務，開發人員可以透過網頁介面登入 OpenStack，並管理被分配的 IaaS 資源，開發人員登入 OpenStack 之畫面如圖 5 所示。

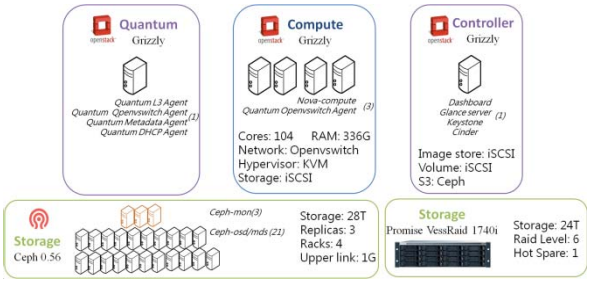

圖 4、NCTU Grizzly OpenStack 架構

表 2 OpenStack 伺服器規格

	CPU	Core	Memory
Controller	Intel X5130 2.00GHz	8	32GB
Compute-1	Intel E5-2620 2.00GHz	24	64GB
Compute-2	Intel E5-2620 2.00GHz	24	64GB
Compute-3	Intel E5-2620 2.00GHz	24	64GB
Compute-4	Intel E5-2620 2.00GHz	32	144GB
Network	Intel X5130 2.00GHz	8	8GB

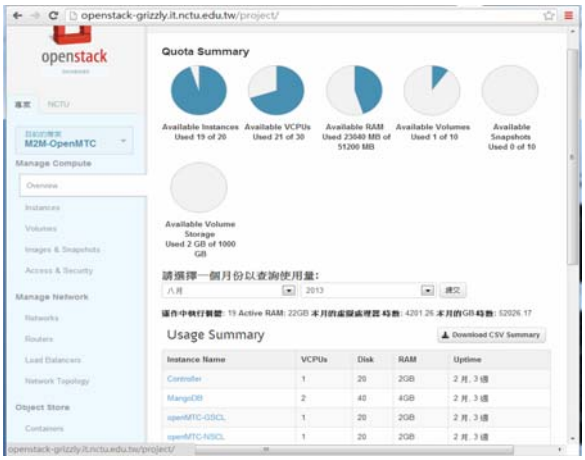

圖 5 OpenStack 登入畫面

3.2 OpenStack Load Balancer - LBaaS

OpenStack 提供了負載平衡器的插件，協助建立一個高可靠度的雲端服務，以避免因單一伺服器當機而停止服務或是流量過高超過單台伺服器的負荷，本論文使用此插件提供負載平衡的服務，此插件名稱稱之為：LBaaS(Load Balancer as a Service)。

LBaaS 安裝與設定之後，使用畫面如圖 6 所示，使用者可以透過 Dashboard 建立 Load Balancer 的 Pool、Member、Health Monitor 與 VIP。除了以 GUI 介面設定之外，我們還可以透過呼叫 API 的方式設定 Load Balancer 的服務，呼叫方式如圖 7 先建立一個名為 webpool 的 Load Balancer 服務。接著再透過 quantum lb-member-create 指令加入 2 台 web server 到 webpool 中，如圖 8。另外可以建立 health monitor 以監控後端的 web server 是否正常，建立的方式(quantum lb-healthmonitor-create)如圖 9，最後需再建立對外服務的 Virtual IP Address(VIP)，最後達成負載平衡的目的，如圖 10 所示。

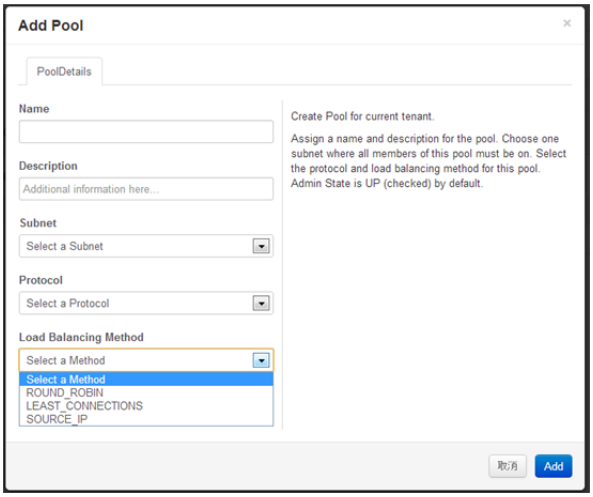

圖 6 LBaaS in OpenStack

```
chsu@OpenStack-Grizzly:~$ quantum lb-pool-create --lb-method ROUND_ROBIN --name webpool --protocol HTTP --subnet-id ea075
0c7-8293-4755-9e29-e86cbeff41c5
Created a new pool:
+------------------+--------------------------------------+
| Field            | Value                                |
+------------------+--------------------------------------+
| admin_state_up   | True                                 |
| description      |                                      |
| health_monitors  |                                      |
| id               | fd659cf5-499b-4a25-8329-dcf1d061224f |
| lb_method        | ROUND_ROBIN                          |
| members          |                                      |
| name             | webpool                              |
| protocol         | HTTP                                 |
| status           | PENDING_CREATE                       |
| subnet_id        | ea0750c7-8293-4755-9e29-e86cbeff41c5 |
| tenant_id        | fa5dd1ac50664703af87853599db1544     |
| vip_id           |                                      |
+------------------+--------------------------------------+
```

圖 7 LBaaS API 建立 pool API

```
chsu@OpenStack-Grizzly:~$ quantum lb-member-create --address 172.16.1.14 --protocol-port 80 webpool
Created a new member:
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| address        | 172.16.1.14                          |
| admin_state_up | True                                 |
| id             | 94f52676-98ac-480a-be23-a4f4a849fd3d |
| pool_id        | fd659cf5-499b-4a25-8329-dcf1d061224f |
| protocol_port  | 80                                   |
| status         | PENDING_CREATE                       |
| tenant_id      | fa5dd1ac50664703af87853599db1544     |
| weight         | 1                                    |
+----------------+--------------------------------------+
chsu@OpenStack-Grizzly:~$ quantum lb-member-create --address 172.16.1.15 --protocol-port 80 webpool
Created a new member:
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| address        | 172.16.1.15                          |
| admin_state_up | True                                 |
| id             | 42b6c20f-46bc-4c64-92c9-36b671ce8ac9 |
| pool_id        | fd659cf5-499b-4a25-8329-dcf1d061224f |
| protocol_port  | 80                                   |
| status         | PENDING_CREATE                       |
| tenant_id      | fa5dd1ac50664703af87853599db1544     |
+----------------+--------------------------------------+
```

圖 8 LBaaS API 加入 member

```
chsu@OpenStack-Grizzly:~$ quantum lb-healthmonitor-create --delay 3 --type HTTP --max-retries 3 --timeout 3
Created a new health_monitor:
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| admin_state_up | True                                 |
| delay          | 3                                    |
| expected_codes | 200                                  |
| http_method    | GET                                  |
| id             | c74f5f64-a574-49ba-b0e6-60b3d8491fad |
| max_retries    | 3                                    |
| status         | PENDING_CREATE                       |
| tenant_id      | fa5dd1ac50664703af87853599db1544     |
| timeout        | 3                                    |
| type           | HTTP                                 |
| url_path       | /                                    |
+----------------+--------------------------------------+
```

圖 9 LBaaS API 建立 health monitor

```
chsu@OpenStack-Grizzly:~$ quantum lb-vip-create --name webvip --protocol-port 80 --protocol HTTP --subnet-id  ea0750c7-82
93-4755-9e29-e86cbeff41c5 webpool
Created a new vip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| address             | 172.16.1.18                          |
| admin_state_up      | True                                 |
| connection_limit    | -1                                   |
| description         |                                      |
| id                  | 51851952-33fd-4321-8591-3968174cef98 |
| name                | webvip                               |
| pool_id             | fd659cf5-499b-4a25-8329-dcf1d061224f |
| port_id             | 7e61b503-2ed3-48ae-86c6-284a5fa53564 |
| protocol            | HTTP                                 |
| protocol_port       | 80                                   |
| status              | PENDING_CREATE                       |
| subnet_id           | ea0750c7-8293-4755-9e29-e86cbeff41c5 |
| tenant_id           | fa5dd1ac50664703af87853599db1544     |
+---------------------+--------------------------------------+
```

圖 10 LBaas API 建立 VIP

3.3 物聯網整合 OpenStack 雲端作業系統

本論文使用 fokus 公司所開發的 OpenMTC 物聯網平台軟體建置，此軟體所需的作業系統與安裝的軟體套件如表 3，OpenMTC 資料庫採用 MongoDB[9]，程式語言使用 node.js 撰寫，OpenMTC 的資料存取支援 RESTful APIs 提供以 GET 或 POST 交換資料。

表 3 OpenMTC software requirement

OS.	Ubuntu 12.04 64bit
Software	NodeJS 0.8.16
	MongoDB 2.0.4
	GNU make 3.81

4. 系統建置成果與分析

4.1 物聯網雲端平台平行擴展(Scale Out)

在物聯網中，遍佈於世界各地的感測裝置為了傳回資料，將會產生成千上萬甚至上億個連線，若只靠傳統的方式使用單一台伺服器接收回傳的資料，系統可能會因架構無法平行擴展，很快地遇到效能的瓶頸。然而雲端運算強調的是快速有彈性的平行擴展，因此本論文將 OpenMTC 建置在 OpenStack 上，並使用 OpenStack 的負載平衡 LBaaS，當連線數到達一定的門檻值後，搭配 OpenStack API 的呼叫，彈性地擴充或縮減後端服務的伺服器，建置一個可平行擴充的物聯網雲端平台。

圖 11 為本論文的物聯網雲端平台架構圖，在圖裡的 Device Domain 中，我們可以想像有成千上萬的 DA(Device Application)節點透過 GSCL 傳送回 Network Domain，本論文將 OpenMTC 架設在 OpenStack 的平台上，負載平衡器使用 OpenStack 提供的 LBaaS，如圖 12 所示，本論文建立了 2 個負載平衡器於 NSCL 上使用，分別為提供 Device Domain 回傳資料的 Load Balancer A 與提供給 Application Domain 使用的 Load Balancer B，而每

台 Load Balancer 後端有 2 台 NSCL 負責服務，如圖 13 所示。

在雲端平台平行擴展方面，當連線數超過一定的門檻值後，本論文使用 OpenStack Nova API 指令”nova boot --flavor=1 --image=NSCL_Orig NSCL-3”(如圖 14)呼叫開啟新的虛擬機，虛擬機內自動執行 OpenMTC 軟體的 NSCL 程式，再呼叫 OpenStack Quantum LBaaS API 指令” quantum lb-member-create --address 172.16.1.26 --protocol-port 15000 NSCL-mia-15000-LB”(如圖 15)將新的虛擬機加入到 Load Balancer 的 Pool 中(172.16.1.26 為虛擬機 NSCL-3 所取得的 IP 位址)，此時後端就有 3 台 NSCL 伺服器服務了。另一個情況是當連線數降低時，系統會以 API 指令自動將虛擬機關閉(使用 nova delete 指令)，並移除 Load Balancer 的 pool 中停止使用的 IP(使用 quantum lb-member-delete 指令)，平行擴展之演算法如下方文字方塊所描述。

```
max_VM_size =8
min_VM_size =2
current_VM = min_VM_size

while (check LBaaS incoming connection session every 1 second)
{
 if number of session > 30*current_VM and current_VM   <
max_VM_size
    {
          OpenStack API boot new VM
          OpenStack LbaaS API add VM as a member to openMTC Load
Balancer Pool
          current_VM++
    }
 else if number of session < 30*(current_VM - 1) and current_VM >
min_VM_size
    {
           OpenStack API terminate new VM
          OpenStack LbaaS API remove VM as a member to openMTC
Load Balancer Pool
          current_VM++
    }
 else
    exit
}
```

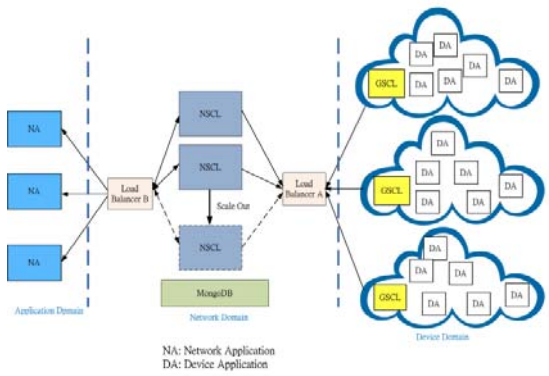

圖 11 物聯網雲端平台架構圖

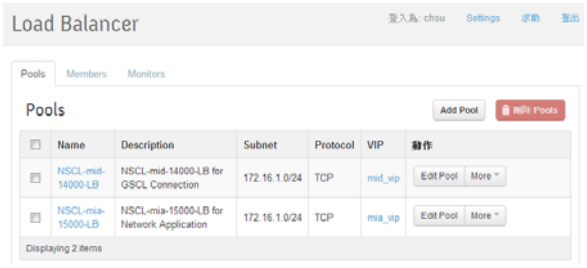

圖 12 物聯網 OpenMTC LBaaS

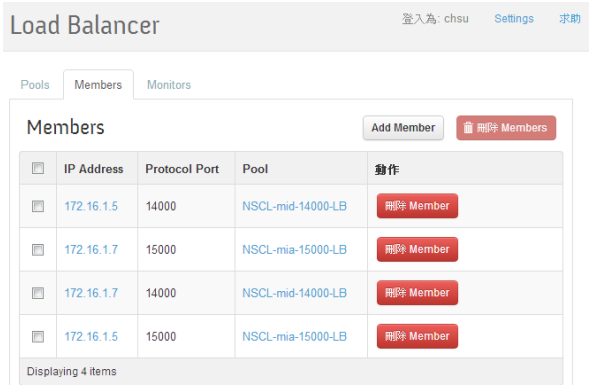

圖 13 LBaaS Member – NSCL

```
chsu@OpenStack-Grizzly:~$ nova boot --flavor=1 --image=NSCL_Orig NSCL-3
+-----------------------------+--------------------------------------+
| Property                    | Value                                |
+-----------------------------+--------------------------------------+
| status                      | BUILD                                |
| updated                     | 2013-08-22T13:41:06Z                 |
| OS-EXT-STS:task_state       | scheduling                           |
| key_name                    | None                                 |
| image                       | NSCL_Orig                            |
| hostId                      |                                      |
| OS-EXT-STS:vm_state         | building                             |
| flavor                      | m1.tiny                              |
| id                          | b2615ea7-9158-4f3b-8055-e62e657e8464 |
| security_groups             | [{u'name': u'default'}]              |
| user_id                     | b49b710a33fa429ba5fdbb7a5456279a     |
| name                        | NSCL-3                               |
| adminPass                   | ZyCBCYX4MdiH                         |
| tenant_id                   | fa5dd1ac50664703af87853599db1544     |
| created                     | 2013-08-22T13:41:06Z                 |
| OS-DCF:diskConfig           | MANUAL                               |
| metadata                    | {}                                   |
| accessIPv4                  |                                      |
| accessIPv6                  |                                      |
| progress                    | 0                                    |
| OS-EXT-STS:power_state      | 0                                    |
| OS-EXT-AZ:availability_zone | nova                                 |
| config_drive                |                                      |
```

圖 14 OpenStack Nova boot API

```
chsu@OpenStack-Grizzly:~$ quantum lb-member-create --address 172.16.1.26 --protocol-port 15000 NSCL-mia-15000-LB
Created a new member:
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| address        | 172.16.1.26                          |
| admin_state_up | True                                 |
| id             | c4293871-a470-4c08-addc-9c9e4aab26f7 |
| pool_id        | 11fcc873-1986-44f8-a37f-71b0ed5e896f |
| protocol_port  | 15000                                |
| status         | PENDING_CREATE                       |
| tenant_id      | fa5dd1ac50664703af87853599db1544     |
| weight         | 1                                    |
+----------------+--------------------------------------+
```

圖 15 OpenStack Quantum LBaaS API

4.2 效能測試模擬結果

因本實驗系統尚未有大量的物聯網感測元件回傳感測資料以及大量的應用(i.e., Network Application)，因此本論文採用 Jmeter 軟體執行壓力測試，Jmeter 參數設定如表 4 所示，本論文建立兩個測試項目，第一個項目是針對 OpenMTC 的埠 15000 進行 HTTP Request(Get)，同時送出 100 個 request，重複 100 次，總共取樣 10000 次。第二個改為同時送出 1000 個 request，重複 10 次，取樣總數一樣為 10000 次。我們將會比較此系統於 Load Balancer 後端不同的 NSCL 個數之平均回應時間與

每秒鐘可以處理要求的數量。

圖 16 為 NSCL 個數與系統平均回應時間之比較圖，當後端只有 1 台 NSCL 服務時，同時有 1000 個連線存取的平均回應時間遠大於同時指有 100 個連線的時間，當 Load Balancer 後端後端陸續增加了 NSCL 的個數之後，平均反映時間很快地將低了，代表著當連線數較多的時候，使用 Load Balancer 的效果越明顯。

圖 17 表示每秒鐘系統可以處理 request 的數量，每秒可處理的數量，會隨著後端的 NSCL 數量增加而變多。在同時有 8 個 NSCL 時，因為 8 個 NSCL 可以較快處理完成同時 100 個連線，因此執行緒 100 比執行緒 1000 有較高的吞吐量(每秒可處理數量)。

表 4 Jmeter 參數

Jmeter 參數				
測試 ID	**取樣數**	**執行緒數量**	**迴圈數**	**HTTP**
1	10000	100	100	GET
2	10000	1000	10	GET

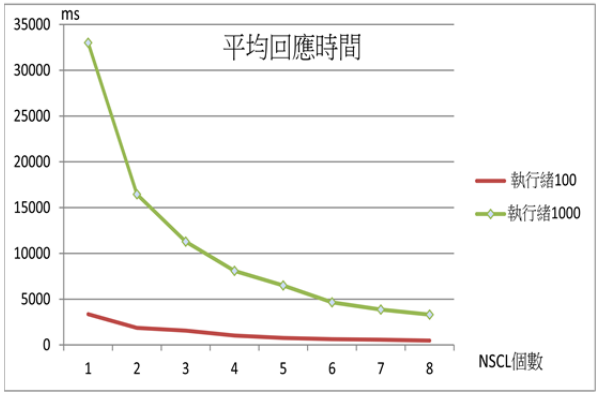

圖 16 平均回應時間

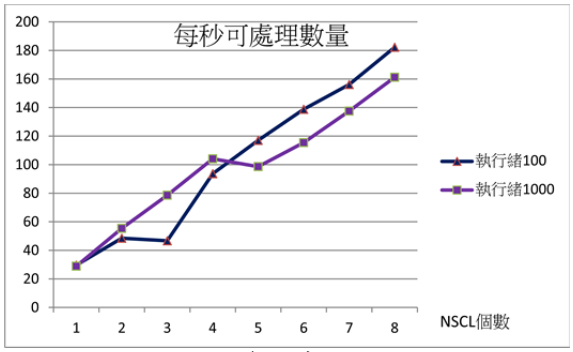

圖 17 每秒處理個數

5. 結論與未來工作

本論文以 OpenStack 為 IaaS 基礎架構，整合 fokus 公司的物聯網系統軟體 OpenMTC，利用 OpenStack API 指令，控制虛擬機的開關機以及 Load Balancer 成員的加入與刪除，達成自動擴展(auto scaling)物聯網服務的目的。此系統透過效能的模擬，可以得知當 NSCL 的個數增加時，平均回應時間可以有效地下降，提升系統的效率，尤其是在同時有許多連線數的情況之下，更能夠顯著地提升效能，降低平均反應的時間。

由於目前尚未有大量的感測節點於網路上運作，因此本論文使用 Jmeter 軟體以模擬的方式，模擬物聯網的運作，後續待校內系所開發許多應用之後，將可以預期有大量的資料量使用本系統，有可能會遇到不可預知的問題發生。未來我們模擬時，Jmeter 將使用不同之參數來模擬更多大量的連線需求，以利了解更多大量連線需求時，系統所可能遭遇到的問題。

OpenMTC 使用 Mongo 儲存感測端所送回的資料，當儲存的資料量變多時，MongoDB 效能可能會將低，因此 MongoDB 支援自動擴展之功能相形重要，未來將會開發 MongoDB as a Service 於 OpenStack 上。

參考文獻

[1] 維基百科-物聯網, http://zh.wikipedia.org/zh-tw/物聯網

[2] 電子時報 - 雲端運算加速物聯網市場發展, http://www.digitimes.com.tw/tw/iot/shwnws.asp?CnlID=15&id=0000280390_MQB4PCGM8HZOUK1TCINNU&ct=1

[3] Fraunhofer Fokus, http://www.fokus.fraunhofer.de/en/ngni/index.html

[4] The NIST Definition of Cloud Computing, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[5] OpenStack, "OpenStack Open Source Cloud Computing Software", http://www.openstack.org/

[6] OpenStack 是一套打造 IaaS 服務的開源軟體, http://www.ithome.com.tw/itadm/article.php?c=81091&s=2

[7] Amazon Simple Storage Service, http://aws.amazon.com/s3/

[8] OpenStack 版本 history

[9] mongoDB, http://www.mongodb.org/

[10] node.js, http://nodejs.org/