開放原始碼雲端運算平台服務需求配置之研究 ---以 OpenStack 為例

王淑卿
朝陽科技大學
scwang@cyut.edu.tw

嚴國慶*
朝陽科技大學
kqyan@cyut.edu.tw
(聯絡人)

陳慶維
朝陽科技大學
s10114901@cyut.edu.tw

留旻懋
朝陽科技大學
s10214612@cyut.edu.tw

摘要

隨著雲端運算的技術及應用不斷的進步，開放原始碼的雲端運算平台紛紛出現，不論是屬於平台即服務(Platform as a Service；PaaS)或屬於基礎設施即服務(Infrastructure as a Service；IaaS)，都各有其代表性的平台。但這些開放原始碼的平台卻各有其優缺點，因此必須透過不同平台的整合，才能符合使用者需求，使得雲端運算環境建置的成本相對升高。因此本研究以 OpenStack 為例，提出一套架構在雲端運算平台的系統架構與虛擬機器配置策略。本研究所建置的系統架構位於 PaaS 與 IaaS 之間，可做為資源配置的中介層。而虛擬機器配置策略則利用先到先服務(First Come First Served；FCFS)排班演算法及最先合適(First Fit)資源配置演算法，將使用者的服務需求配置到適當的虛擬機器。透過本研究所提出的系統架構與虛擬機器配置策略，除了可以達到服務需求與虛擬資源的有效配置，更可以降低 IaaS 中實體與虛擬資源的浪費。

關鍵詞：雲端運算平台、OpenStack、虛擬機器配置策略、平台即服務、基礎設施即服務。

Abstract

With cloud computing technologies and applications continue to progress, the open source cloud computing platform have been emerged. However, there are the representative platforms of Platform as a Service (PaaS) or Infrastructure as a Service (IaaS). In order to meet the requirements of user, open source cloud computing platforms need to be integrated. Therefore, a system architecture and virtual machine allocation strategy based on OpenStack are proposed in this study. The system architecture is used as an intermediate layer between PaaS and IaaS. The virtual machine allocation strategy uses the First Come First Served (FCFS) scheduling algorithm and the First Fit allocation algorithm to configure the appropriate virtual machine. By using the proposed system architecture and virtual machine allocation strategy, the efficient allocation of virtual resources can be achieved, and the waste of physical and virtual resources in IaaS can be reduced.

Keywords: Cloud Computing Platform, OpenStack, Virtual Machine Allocation Strategy, PaaS, IaaS.

1. 前言

近幾年網路頻寬及硬體設備相關技術不斷的提昇，促使網際網路的相關應用逐漸蓬勃發展。架構在網際網路的雲端運算(Cloud Computing)為一分散式系統(Distributed System)之概念，其使用較低效能的機器來達到高可靠性(Reliability)及高效率(Efficiency)的運算能力[1]。

雲端運算是一種具高延展性的網路環境，經由網際網路提供運算、儲存與頻寬的能力，並且透過「服務(Service)」的形式來滿足使用者的需求。另外，雲端運算具備隨需即用(On-demand Self-service)的特性，使用者可以根據自己的需求向供應商購買適合的資源，供應商能根據使用者的需求制定客製化服務。雲端運算改變了傳統網路服務供應商的運作模式，創造新型的服務方式，能夠把資訊科技的能力，如運算能力、儲存能力及頻寬速度等，透過網際網路提供給使用者[1]。

雲端運算提供三種服務模式，包括：基礎設施即服務(Infrastructure as a Service；IaaS)，平台即服務(Platform as a Service；PaaS)和軟體即服務(Software as a Service；SaaS)[1,11]。SaaS 提供雲端運算環境的服務模式給使用者，使用者透過網際網路使用雲端環境所提供的應用程式。SaaS 狹義的定義為雲端運算環境中提供

的服務，如電子信箱、社交網路與網頁遊戲等。廣義的 SaaS 不僅包含狹義的軟體即服務，更進一步涵蓋 PaaS 及 IaaS 的服務模式，如提供使用者網路系統桌面、防火牆、雲端平台與虛擬機器等，都是屬於 SaaS 的一部分。

PaaS 將使用者開發的應用程式或網頁部署在雲端運算環境，使用者不需自行架設開發平台，只要透過網路、瀏覽器或簡易的應用程序使用雲端供應商提供的開發環境，就可以進行應用軟體的開發或網站的建置。PaaS 是 SaaS 衍生出來的一種服務方式，目前已有許多企業或自由軟體協會推出各具特色的雲端平台環境，如 Google App Engine、Microsoft Azure 或 Amazon Web Services(AWS)等[2]。

IaaS 提供雲端運算環境中的硬體設備資源讓使用者使用，有時可稱為「公用運算」(Utility Computing)，也就是提供處理器、儲存硬碟、網路頻寬與其他資源的租用服務。換言之，IaaS 除了提供硬體設備資源，也提供運算能力及儲存能力，如 Amazon EC2(Elastic Compute Cloud)、Rackspace、Eucalyptus Cloud、OpenStack 以及 Hadoop 等[2]。

隨著雲端運算技術及應用不斷的進步，許多開放原始碼的雲端運算平台陸續被探討，本研究主要探討 OpenStack 平台。OpenStack 擁有自組開發的核心組件以及定義良好的 API(Application Programming Interface)，可以讓企業或雲端供應商針對各自需求建立屬於自己的公有雲或私有雲[3,7,8,12,17]，OpenStack 也相容 Amazon EC2 規格，因此可避免企業或雲端供應商面臨系統移植時衍伸的問題[3,8,15]，但是有關於 OpenStack 相關的學術研究議題仍較少被探討[9]。

在雲端運算平台環境下，企業或雲端供應商可以透過網頁圖形化使用者界面來管控虛擬機器(Virtual Machine)或配置用戶組的資源[6]。然而，目前 OpenStack 對於系統的監控與資源配置皆需由管理者以人為操作的方式設定。因此 OpenStack 的管理者在資源配置時，都需仰賴經驗以估計值來進行資源的配置。所以，OpenStack 在面對使用者提出服務需求時，無法有效進行系統的監控及提供資源動態配置[4]。以致可能造成無法有效配置適當的虛擬機器給服務需求，導致資源的浪費。

本研究在 OpenStack 雲端平台架構一套系統架構並提出一套虛擬機器配置策略。當使用者的需求從 SaaS 進入 OpenStack 時，將在排程器(Scheduler)的等待佇列(Waiting Queue)進行等待，排程器依先到先服務(First Come First Served；FCFS)排班演算法，將服務需求傳送給分派器(Assigner)。分派器經由代理人(Agent)進行機器資訊的蒐集，並透過最先合適(First Fit)配置策略，將使用者的需求與虛擬機器進行匹配，最後把服務需求分配到適當的虛擬機器執行。

在本文第 2 節將說明開放原始碼雲端運算平台、CPU 排班演算法及記憶體配置演算法，第 3 節說明本研究所提出的系統架構及虛擬機器配置策略，第四節則為範例說明，第 5 節為結論與未來研究。

2. 文獻探討

在本節中將分別說明開放原始碼雲端運算平台、CPU 排班演算法及記憶體配置演算法。

2.1 開放原始碼雲端運算平台

開放原始碼雲端運算平台已逐漸被重視與應用，在本節中將針對實務性評價較高的 OpenStack、Eucalyptus Cloud 與 Hadoop 等進行說明。

2.1.1 OpenStack

OpenStack 是 2010 年 7 月由 Rackspace 和 NASA(National Aeronautics and Space Administration)共同發起的雲端運算及儲存計畫並合作研發的雲端運算平台，目前已有超過 150 家的企業支持 OpenStack，如 AMD、Intel、HP、Dell、PayPal 與 Cisco 等企業，國內則包括工研院與廣達電腦[3,7,8,12,17]。OpenStack 由多個不同核心功能的元件組成，包括：運算 Nova、物件儲存 Swift、區塊儲存 Cinder、網路通訊 Neutron(Network)、身分識別 Keystone、映像檔管理 Glance 及提供管理介面的儀表板 Horizon 等。圖 1 所示為 OpenStack 的核心架構，各元件功能說明如下。

- Nova：依使用者需求提供對虛擬機器部署及管理的功能。
- Swift：可擴展的分布式儲存平台，以防止單點故障的情況產生，可存放非結構化的資料。
- Cinder：整合運算套件，可讓管理者查看儲存設備的容量使用狀態。
- Neutron(Network)：提供網路連線能力。

- Keystone：提供所有元件或使用者進行身分的驗證及授權。
- Glance：提供虛擬機器映像檔案的管理，將常用的映像檔提供給 Nova。
- Horizon：提供所有元件模組化的網頁管理介面。

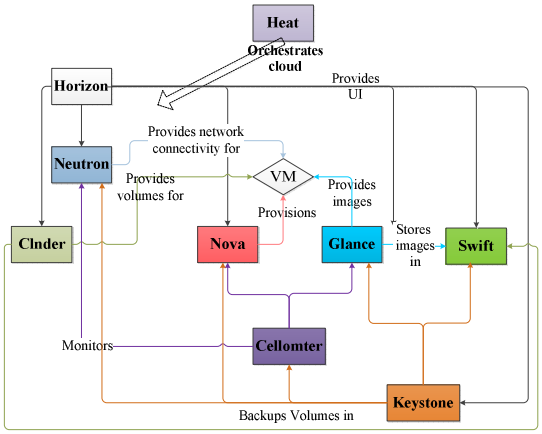

圖 1 OpenStack 核心架構[17]

2.1.2 Eucalyptus Cloud

Eucalyptus 是一種開放原始碼的軟體基礎結構，藉由計算叢集機器或工作站來實現彈性與實用的雲端運算平台。最初是美國加利福尼亞大學 Santa Barbara 計算機科學學院的研究，目前已被實現在商業應用上並發展為 Eucalyptus Systems Inc。雖然 Eucalyptus 已商業化，但仍以開放原始碼的方式來進行維護及開發。

Eucalyptus Cloud 系統核心組件是由 Cloud Controller(CLC)、Cluster Controller(CC)及 Node Controller(NC)組成。CLC 主要負責管理整個系統，也是使用者進入系統的端口；CC 負責控制及收集所有 NC 的訊息，並控制虛擬機器配置到各個 NC；NC 是虛擬化機器主要放置的節點，也是執行所有虛擬化過程的核心[6]。Eucalyptus Cloud 系統架構如圖 2 所示。

2.1.3 Hadoop

Hadoop 是 Apache 軟體基金會研發的開放原始碼平行運算編程工具和分散式檔案系統[5,16]，以 Java 程式語言建構而成，可以提供龐大資料進行分散式運算的環境。Hadoop 建立一個 Cluster 平台，利用 MapReduce 的概念將工作分配到多個 Cluster 進行平行運算[1,16]。

Hadoop 是處理與儲存大量資料的雲端運算平台，藉由平行分散式檔案處理策略，來獲得高效率的資料處理，包含 MapReduce 及 HDFS 兩個執行架構。MapReduce 是分散式程式的框架，程式透過 Map 及 Reduce 的方式進行分散及收斂的動作。HDFS 是 Master/Slave 架構，由兩種角色組成，分別為 Namenode 及 Datanode，Namenode 負責檔案系統中的檔案權限管理，如圖 3 中 Namenode 透過本身紀錄的 Metadata，管理檔案權限及副本檔案位置；Datanode 由數個節點所組成。當一個資料檔進入 HDFS 架構時，會被切割成數個較小的區塊(Block)並且儲存在不同的 Data node 中[1,16]。

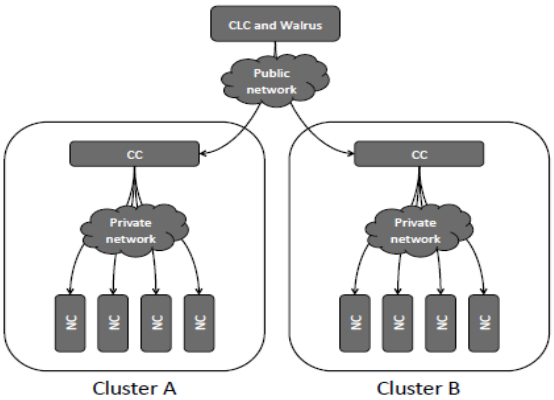

圖 2 Eucalyptus Cloud 系統架構[6]

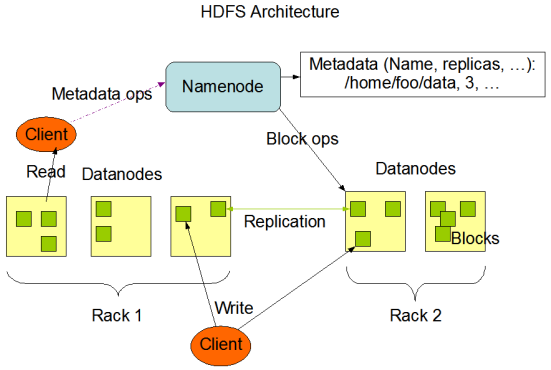

圖 3 HDFS 架構[1]

2.2 CPU 排班演算法

CPU 排班演算法是指在單核心處理器的環境，處理器無法同時處理多個程序(Process)，因此當多個程序同時進入並等待執行時，需從暫存佇列選擇一個程序來執行，剩餘程序則在 I/O 暫存佇列，並且等待 CPU 釋放出資源時，再從 I/O 佇列中選取下一個程序，此依序排班等待 CPU 執行的模式，稱之為不可搶先(Non-preemptive)機制。另一種排班模式為可搶先(Preemptive)機制，當一個更高優先層級

的程序進入 I/O 暫存佇列，則會中斷 CPU 原先執行的程序，並執行高優先層級的程序，原本 CPU 執行的程序將被放在 I/O 暫存佇列中[2]。本小節將說明不同特性的 CPU 排班演算法[2]。

2.2.1 先到先服務(First Come First Served；FCFS)排班演算法

先到先服務排班演算法是以先進入暫存佇列的程序先進行服務，因此將優先進入的程序分配給 CPU 執行，直到 CPU 執行完程序時，再把下一個優先進入的程序分配給 CPU。由於每個程序執行的時間不盡相同，因此下一個程序的等待時間，必須依據上一個程序執行的時間而定，因此當上一個程序執行時間過長，可能會造成 CPU 無法快速處理後續的程序，導致影響整體的效率[2,10]。

2.2.2 依序循環(Round-Robin；RR)排班演算法

依序循環排班演算法在排班時加入時間量(Time Quantum)的規則，將 I/O 暫存佇列的所有程序視為一個環狀佇列，以固定的時間量讓每個程序都能被 CPU 執行。因此時間量的長短將影響整體執行的效能，如果時間量太短，程序將經常進行轉換，導致執行效率不佳[2,10]。

2.2.3 優先(Priority)排班演算法

每個程序都有一個優先權值，以優先權值來決定於程序的執行順序。從暫存佇列中最高優先權的程序分配給 CPU 執行，當遇到相同優先權值的程序時，以先到先服務排班演算法進行分配。優先排班演算法的缺點是會使得低優先權值的程序一直被搶先執行，形成低優先權值的程序發生飢餓(Starvation)的情況[2,10]。

2.2.4 最短優先(Shortest Job First；SJF)排班演算法

從暫存佇列中挑選執行時間最短的程序，優先分配給 CPU 執行，直到暫存佇列的程序被分配完為止。遇到時間相同的程序，則使用先到先服務排班演算法進行分配。最短優先排班演算法的缺點是會使執行時間長的程序一直被搶先執行，形成執行時間長的程序一直無法被 CPU 執行，導致發生飢餓的情況[2,10]。

2.3 記憶體配置策略

當程序進入作業系統時，作業系統考慮每個程序所需的記憶體及系統內閒置的記憶體空間，再決定該程序可被分配的記憶體。當程序被系統執行完成時，該程序會釋放原佔有的記憶體，被釋放的記憶體空間將被併入到可用區間集合裡。在本節中將說明三種最常被用來執行記憶體配置策略的方法[10]。

2.3.1 最先合適(First Fit)

最先合適配置策略是從可用記憶體空間列表(Free Memory Space List)的初始位置或前一個搜尋的結束位置開始搜尋，當找到第一個能滿足程序的記憶體空間需求，便將此記憶體的空間進行配置。最先合適配置策略的優點在於能夠快速地將記憶體需求進行配置，並提高記憶體的使用率；缺點是每次都以優先找到第一個合適的記憶體空間進行配置，但因記憶體的空間大小未經排序，所以容易造成記憶體空間出現內部碎片的現象[10]。

2.3.2 最好合適(Best Fit)

最好合適配置策略是從可用記憶體空間列表找出所有能滿足程序需求的記憶體空間，並將會產生最小剩餘空間的記憶體配置給程序需求。由於最好合適配置策略必須搜尋整個串列，直到找出滿足程序需求且為最小剩餘的記憶體空間，因此最好合適配置策略對記憶體能夠達到更好的使用率，但在分配速度上略遜於最先合適配置策略。除此之外，若原切割的記憶體區塊不符合程序需求，則極易產生外部碎片的狀況[10]。

2.3.3 最差合適(Worst Fit)

最差合適配置策略從可用記憶體空間列表找出所有能滿足程序需求的記憶體空間，並配置最大的剩餘空間給程序需求。由於最差合適配置策略必須搜尋整個串列，直到找出滿足程序需求且是最大的記憶體空間，因此最差合適配置策略不管是在記憶體使用率或者分配速度上都遜於最先及最好合適的配置方法，並且容易造成內部碎片的產生。但對於需求會增加，又不適合重新分配的程序，反而較合適最差合適配置策略[10]。

3. 系統設計

在本節中將說明本研究所設計的 OpenStack 系統架構與 OpenStack 雲端平台虛擬機器配置策略。

3.1 OpenStack 系統架構

本研究探討基於 OpenStack 雲端平台虛擬機器資源配置策略，提出適用於 OpenStack 雲端平台的架構。當使用者的需求進入到 SaaS 時，將根據不同性質的服務需求(Service Request；SR)分類到不同種類的服務區塊，並且傳送到 PaaS。當服務需求進入到 PaaS 時，PaaS 的排程器(Scheduler)會將需求放置到等待佇列(Waiting Queue)，並依據先到先服務排班演算法傳送給 PaaS 與 IaaS 之間的分派器(Assigner)。

代理人(Agent)以 Polling 的方式，從 IaaS 蒐集各實體主機(Server)與各實體主機中虛擬機器相關的資源資訊。代理人所搜集的資源資訊將依遞增的方式進行排序，並記錄在資料表中，包括 CPU 處理能力與 Memory 的需求量等。分派器則採用最先合適配置策略，依據排序後資料表中的CPU處理能力與Memory需求量，與使用者所提出的服務需求進行配置，當找到第一個合適的虛擬機器時，即將需求分配給該虛擬機器執行。所有在等待佇列中等待被執行的程序將依序被分派執行，直到等待佇列中所有需求被分配完為止，如圖 5 所示。

由於虛擬機器資源的排序結果將影響配置策略的成效，因此虛擬機器的資源資訊必須先行進行排序[15]。在本研究中，為了避免小的服務需求被配置到較大的虛擬機器資源，導致資源的浪費，因此採用最先合適配置策略，依據最先合適之特性，將虛擬機器資源以遞增的方式進行排序，並將第一個符合服務需求的虛擬機器配置給需求。

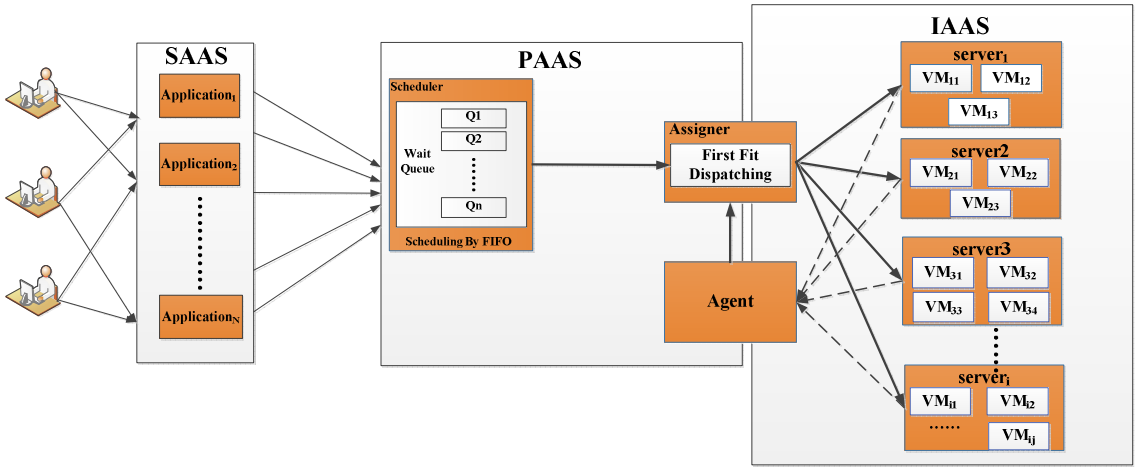

圖 5 OpenStack 雲端平台系統架構

3.2 OpenStack 雲端平台虛擬機器配置策略

當服務需求從排程器中的等待佇列傳送到分派器時，分派器會從代理人得到 IaaS 已排序後的虛擬機器資料表，透過遞增排序的最先合適記憶體配置策略開始進行服務需求的分配，以下是 RAS 進行服務需求配置的步驟：

[步驟一]： 分派器由虛擬機器資源配置表篩選目前為空閒的虛擬機器，並將篩選結果根據 CPU 處理速度依遞增順序進行排序。

[步驟二]： 從排程器接收到的服務需求，使用先到先服務排班演算法，將優先到達的服務需求根據 CPU 處理速度與[步驟一]篩選出來目前為空閒狀態之虛擬機器資源表以最先合適配置策略進行服務需求的配置。服務需求的配置有兩種情況：

(1) 找到符合服務需求中 CPU 處理速度的虛擬機器，並判斷其記憶體是否能滿足服務需求的需求，若能滿足則進行[步驟三]。若無法滿足，則找下一個虛擬機器進行匹配。若所有現有空閒及符合其 CPU 處理速度需求的虛擬機器，都無法滿足記憶體需求，則將該服務需求放置到等待佇列的最後，等待下一次的服務需求重新被配置。

(2) 未找到符合服務需求中 CPU 處理速度的虛擬機器，代表目前該服務需求無法被執行，該服務需求將被放置到等待佇列的最後，等待下一次服務需求重新被配置。

[步驟三]：將服務需求配置給[步驟二]所選到符合服務需求的虛擬機器進行執行，更改虛擬機器資源配置表的狀態，並將該虛擬機器從資源列表中移出。接著，重複[步驟二]進行下一個服務需求的配置，直到等待佇列中所有服務需求分配完畢為止。

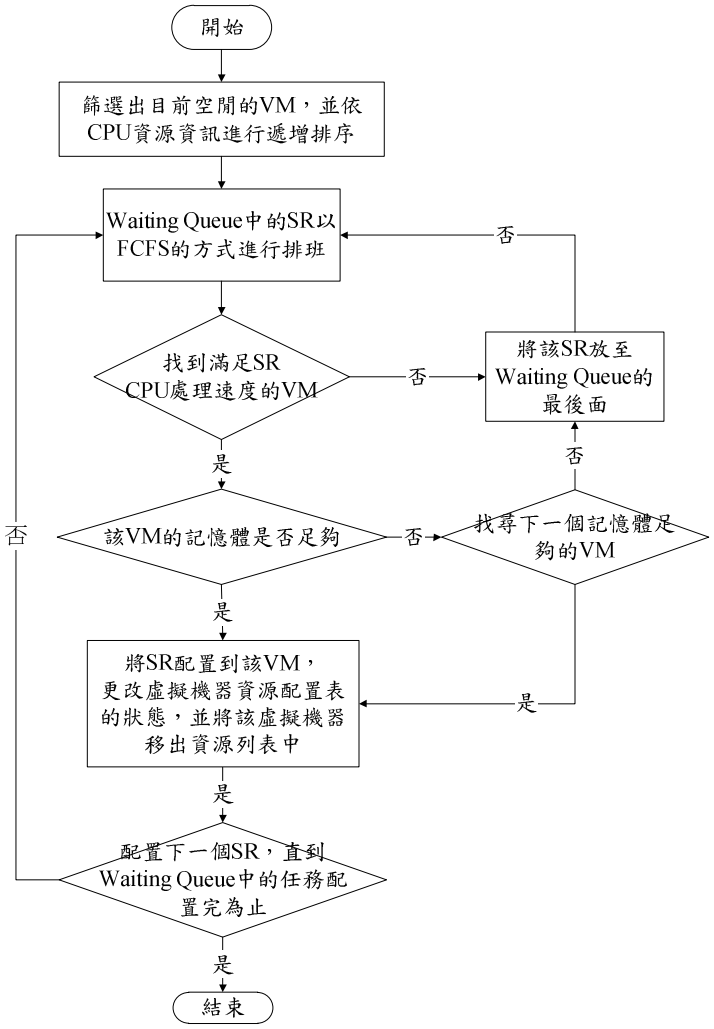

圖 6 OpenStack 雲端平台虛擬機器配置策略流程圖

4. 範例說明

在本節中，將以範例說明本研究所提出架構在 OpenStack 雲端運算平台的虛擬機器配置策略。針對範例的說明，假設：

(1) 排程器裡等待佇列中的服務需求，可以經由分派器配置給 IaaS 中的虛擬機器執行；

(2) 在初始環境的狀態，所有虛擬機器都是閒置的狀態。

在範例中，所有的服務需求會被放置在服務需求等待佇列中。如表 1 所示，在服務需求等待佇列中共有七個需求，其中 SR_1 為第一個進入等待佇列的需求。在服務需求表會記錄所有服務需求所需要的 CPU 執行速度及記憶體需求量，如表 2 所示。

表 1 服務需求等待佇列

SR_7	SR_6	SR_5	SR_4	SR_3	SR_2	SR_1

表 2 服務需求表

Service Requests	CPU(MB/s)	記憶體(MB)
SR_1	540	1024
SR_2	200	512
SR_3	350	1024
SR_4	300	256
SR_5	150	512
SR_6	500	1024
SR_7	370	512

代理人所搜集的資源資訊,包括 CPU 處理速度與記憶體剩餘量，將依遞增的方式進行排序，並記錄在資料表中，如表 3 虛擬機器資源配置表所示。除此之外，在虛擬機器資源配置表中亦紀錄虛擬機器的忙碌狀態，當 Used Flag=1，表示虛擬機器處於忙碌狀態；Flag=0，表示虛擬機器處於閒置狀態。如果虛擬機器已被分配到服務需求，則會在虛擬機器資源配置表的 Assigned 欄位註記服務需求的 ID。

表 3 虛擬機器資源配置表

VM_{ID}	CPU(MB/s)	Memory(MB)	Used Flag	Assigned
VM_{31}	150	512	0	
VM_{11}	200	1024	0	
VM_{12}	300	2048	0	
VM_{21}	350	1024	0	
VM_{33}	450	512	0	
VM_{32}	550	1024	1	
VM_{22}	600	2048	0	

接著，將依據表 1、2 及 3 的資料執行虛擬機器配置策略。

[步驟一]：排程器以先到先服務排班演算法將表 1 等待佇列中的服務需求進行排班，其中 SR_1 是第一個進入等待佇列的服務需求。接著，分派器率先以 SR_1 的需求進行虛擬機器資源配置。分派器透過代理人蒐集的虛擬機器資源配置表(表 3)可以得知 VM_{32} 剩餘的 CPU 處理速度 550MB/s 與 Memory 剩餘量 1024MB 正符合 SR_1 的 CPU 需求 540 MB/s 與 Memory 需求 1024MB，因此分派器從虛擬機器資源配置表中可得知第一個適合 SR_1 的虛擬機器為 VM_{32}。所以分派器將 VM_{32} 配置給 SR_1，並且更新虛擬機器資源配置表，將 VM_{32} 的 Used Flag 更新為 1，Assigned 欄位標註目前正執行 SR_1，如表 4 所示。因 VM_{32} 虛擬機器已被配置執行 SR_1 服務需求，所以將其由虛擬機器資源配置表刪除。

表 4 虛擬機器資源配置表

(第一次服務需求分配)

VM_{ID}	CPU(MB/s)	Memory(MB)	Used Flag	Assigned
VM_{31}	150	512	0	
VM_{11}	200	1024	0	
VM_{12}	300	2048	0	
VM_{21}	350	1024	0	
VM_{33}	450	512	0	
VM_{32}	550	1024	1	SR_1
VM_{22}	600	2048	0	

[步驟二]：接續配置下一個服務需求 SR_2，[步驟二]的配置流程與[步驟一]相同。由表 3 可以得知 VM_{11} 剩餘的 CPU 處理速度 200MB/s 與 Memory 剩餘量 1024MB 剛好符合 SR_2 的 CPU 需求 200 MB/s 與 Memory 需求 1024MB，因此將 VM_{11} 配置給 SR_2，並且更新虛擬機器資源配置表,將 Used Flag 更新為 1,Assigned 欄位標註目前正執行 SR_1，如表 5 所示。因為 VM_{11} 已經被配置執行 SR_2,所以將其由虛擬機器資源配置表刪除。後續等待佇列中的服務需求，將依照[步驟一]的流程繼續執行，直到等待佇列中的服務需求被執行完畢。

表 5 虛擬機器資源配置表

(第二次服務需求分配)

VM_{ID}	CPU(MB/s)	Memory(MB)	Used Flag	Assigned
VM_{31}	150	512	0	
VM_{11}	200	1024	1	SR_2
VM_{12}	300	2048	0	
VM_{21}	350	1024	0	
VM_{33}	450	512	0	
VM_{22}	600	2048	0	

[步驟三]：當所有的虛擬機器都被配置去執行服務需求時，將形成虛擬機器配置完成表，如表 6 所示。在虛擬機器配置完成表中記錄已被分配到服務需求的虛擬機器，當虛擬機器完成服務需求後，將會回到虛擬機器資源配置表中，等待下一個服務需求進入，進行服務需求與虛擬機器的資源配置。

表 6 虛擬機器配置完成表

VM_{ID}	CPU(MB/s)	Memory(MB)	Used Flag	Assigned
VM_{31}	150	512	1	SR_5
VM_{11}	200	1024	1	SR_2
VM_{12}	300	2048	1	SR_4
VM_{21}	350	1024	1	SR_3
VM_{33}	450	512	1	SR_7
VM_{32}	550	1024	1	SR_1
VM_{22}	600	2048	1	SR_6

5.結論及未來工作

由於目前常見的開放原始碼雲端平台必須透過不同平台的整合，才能符合使用者需求，使得雲端運算環境建置的成本相對升高。因此本研究建構適用於 OpenStack 雲端平台的系統架構，並且提出一個適用於 PaaS 與 IaaS 之間的 OpenStack 雲端平台虛擬機器配置策略。讓使用者所需要的服務需求，經由 OpenStack 雲端平台系統架構及 OpenStack 雲端平台虛擬機器配置策略，有效完成服務需求的配置。OpenStack 雲端平台虛擬機器配置策略經由代理人集結所有虛擬機器的資訊以及服務需求的資訊，分派者再以最先合適配置演算法進行比對，將服務需求配置到第一個合適的虛擬機器中執行，以達到服務需求與虛擬資源有效之配置，並降低雲端運算 IaaS 中實體與虛擬資源的浪費。

然而，本研究所提出的虛擬機器配置策略雖然透過將資源進行排序列表，以降低最先合適配置演算法可能產生的資源浪費問題，但並未考慮實體機器可能各自擁有不同數量或不同資源剩餘量的虛擬機器，因此可能會出現負載不平衡的情況。因此在未來的研究中，將考量各個實體機器或實體機器叢集下虛擬機器的使用情況，藉以改良本研究所提出的資源配置策略，期能達到資源有效配置及虛擬機器使用的負載平衡。

致謝

這篇論文是國科會計畫 NSC102-2221-E-324 -008 研究成果的一部份，在此我們感謝國科會經費支持這個計畫的研究。

參考文獻

[1] 王淑卿、李金鳳、江茂綸、曾莉雅、留旻懋、洪瑋修、簡輝銘、毛上仁，“建構雲端資源管理整合平台-以 Hadoop 為例，”***第七屆資訊科技國際研討會***，2013。

[2] 陳慶維，***在階層是雲端運算網路架構下以負載平衡為導向之排程機制***，**碩士論文，朝陽科技大學資訊管理系碩士班，台中市**，2011。

[3] 黃智國，***以 OpenStack 實現自主 IaaS 的實驗***，碩士論文，朝陽科技大學資訊管理系碩士班，台中市，2012。

[4] Corradi, A., Fanelli, M., and Foschini, L., “VM consolidation: A Real Case based on OpenStack Cloud,” ***Future Generation Computer Systems***, pp.118-127, 2012.

[5] Keller, G., Tighe, M., Lutfiyya, H., and Bauer, M., “An Analysis of First Fit Heuristics for the Virtual Machine Relocation Problem,” ***2012 8th International Conference and 2012 Workshop on Systems Virtualization Management (SVM) Network and Service Management (CNSM)***, pp.406-413, 2012.

[6] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D., “The Eucalyptus Open-source Cloud-computing System,” ***9th IEEE/ACM International Symposium on Cluster Computing and the Grid***, pp.124-131, 2009.

[7] Wuhib, F., Stadler, R., and Lindgren, H., “Dynamic Resource Allocation with Management Objectives-Implementation for an OpenStackCloud,” ***2012 8th International Conference and 2012 Workshop on Systems Virtualization Management (SVM) Network and Service Management (CNSM)***, pp. 309-315, 2012.

[8] Wen, X., Gu, G., Li, Q., Gao, Y., and Zhang, X., “Comparison of Open-Source Cloud Management Platforms: OpenStack and OpenNebula,” ***2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)***, pp. 2457-2461, 2012.

[9] Xu, L., and Yang, J., “A Management Platform for Eucalyptus based IAAS,” ***2011 IEEE International Conference on Cloud Computing and Intelligence***

Systems (CCIS), pp.193-197, 2011.

[10] Silberschatz, A., Galvin, P. B., and Gagne, G., ***Operating System Concepts***, J. Wiley and Sons, 2009.

[11] 黃重憲, "淺談雲端運算," http://www.cc.ntu.edu.tw/chinese/epaper/0008/20090320_8008.htm.

[12] 雲端開放大革：OpenStack OpenStack, http://www.ithome.com.tw/itadm/article.php?c=81098.

[13] 雲端開發專欄（二）：5 分鐘幫你搞懂 http://news.networkmagazine.com.tw/magazine/2012/07/05/40782/.

[14] Eucalyptus Cloud, http://www.eucalyptus.com/docs

[15] Eucalyptus, http://www.oschina.net/p/eucalyptus

[16] Hadoop, http://hadoop.apache.org/.

[17] OpenStack, http://docs.openstack.org/essex/openstack-compute/admin/content.