科技部補助專題研究計畫 □成果報告 ■期中進度報告

高效能巨量資料分析系統之關鍵技術研發及其在電信流量管理之應用-子計畫二：具擷取保證與高效節能之電信巨量資料儲存設計

Design of Retrieval-Guaranteed and Energy-Efficient Big Data Storage Management for Telecom Traffic

計畫類別：□ 個別型計畫　　■ 整合型計畫
計畫編號：NSC 102－2221－E－030－010－MY3
執行期間：102 年 8 月 1 日至　105 年 7 月 31 日

執行機構及系所：輔仁大學資訊工程系

計畫主持人：林振緯
共同主持人：
計畫參與人員：蔡旻軒、謝才晧

成果報告類型(依經費核定清單規定繳交)：■精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
■涉及專利或其他智慧財產權，□一年■二年後可公開查詢

中　華　民　國　103　年　5　月　20　日

科技部專題研究計畫成果報告

具擷取保證與高效節能之電信巨量資料儲存設計

計畫編號： NSC 102-2221-E-030 -010 -MY3

執行期限： 民國 102 年 08 月 01 日到 105 年 07 月 31 日

中文摘要

本年度我們研究在雲端計算中心中，虛擬機遷移策略，以獲取資料的成本作為遷移策略的主要考量因素進而達到在設計巨量電信資料儲存系統時，能夠提高電信流量資料檔案存取的效益。此儲存系統以 Apache Hadoop 平台之檔案系統 HDFS (Hadoop Distributed File System) 為基礎 ，所提出的策略能夠增強其在資料存取、資料管理方面效能、及在電量節能等三方面之功效。

虛擬化的技術，可使實體機的硬體資源虛擬化，形成具有彈性的計算資源並可建立多個不同效能的虛擬機，當使用者提出請求時，雲端運算系統便依照使用者需求提供所需的虛擬機，讓使用者在虛擬機上建置應用程式，但應用程式所需的資料集是散佈在雲端運算系統各個實體機中的，因此每個虛擬機上的應用程式獲得其所需的資料集的時間皆不相同，尤其是資料密集(Data-Intensive)型的應用程式，會大量的讀取資料集，可能花費大量的時間在獲取資料集，而為了降低獲取資料集的整體時間，虛擬機是否能遷移至一個能盡可能降低資料傳輸成本的實體機上，則成為一個重要的議題。

我們將此問題定義為資料感知之虛擬機遷移問題，並設計了找到獲取資料總成本最小的實體機的公式，再透過此公式獲得最佳解，但，為了求得最佳解，將耗費許多的計算時間，因此我們提出以中心點為準的虛擬機遷移策略，結合尋找樹的中心點以及二分圖的模型兩個方法，可在多項式時間內得到近似解，並在最後的模擬實驗中，與另外幾個方法做比較，展現演算法中心點為準的虛擬機遷移策略的效能。

關鍵字：虛擬機遷移、資料中心、資料感知、幾何中心點及二分圖模型。

Abstract

In this year, we investigates the data-aware virtual machine (VM) migration in a cloud data center. We utilize the network topology structure of the cloud data center to design a median-based VM migration scheme. By moving the VM to reduce the average data access time of an application, the SLA requirement of the application can be satisfied. It is very possible that multiple VM may be migrated concurrently. This will introduce the physical machine contention problem to result in some VMs not to be migrated. We applies the bipartite graph modelling to migrate VMs as many as possible and minimize the total migration cost. Finally, we perform simulation

experiments to demonstrate the effectiveness of the proposed scheme.

Keywords: Virtual machine migration, Cloud data center, Geometric median, graph modelling.

一. 研究計畫之動機與目的

本年度我們研究如何提升巨量資料系統的存取效能。近年來雲端運算已成為巨量資料處理的首選平台。而在雲端系統中，虛擬機器與資料之間的距離是影響資料存取速度的關鍵。

雲端運算系統以提供使用者具有擴充性的資源為服務方式，服務的提供原則為按次付費使用，依據使用者所使用的資源量支付其費用，如租借一台虛擬機，以 Amazon EC2 為例，Amazon EC2 提供了 22 種不同規格的虛擬機，讓使用者選擇並租用，此種服務方式稱為基礎設施即為服務(IaaS)。

為了能夠實踐 IaaS 的服務，虛擬化技術[1]為其中關鍵的技術，透過虛擬化技術，將實體機的運算及儲存資源虛擬化，便可提供高於實體機數量的虛擬機。在應用程式的效能中，除了虛擬機所擁有運算能力以外，資料獲取速率也深深的影響應用程式的效能，在雲端系統中，應用程式所需要的資料集，會散佈到各個做為儲存節點的實體機中，當應用程式需要資料集時，虛擬機為了獲取這些資料集，便與做為儲存節點的實體機連線，此時，獲取資料的速率快慢將影響整體的執行效能，獲取資料的速率除了受雲端資料中心的網路速度決定以外，虛擬機與擺放資料集的實體機的距離也是個重要的影響因素，當虛擬機與擺放資料集的實體機的距離較遠時，便會使得獲取資料的時間提高，因而降低了應用程式的整體效能，且應用程式執行時所需要的資資料集往往不只一個，特別是當應用程式為資料密集型(Data-Intensive)的應用程式時，應用程式會需要讀取大量資料集，因此虛擬機將會頻繁的為應用程式獲取資料集，如虛擬機在網路拓樸的結構中的位置不佳，頻繁獲取資料集這行為將會花費大量的時間，導致應用程式的執行效能低落。

為了改善獲取資料的速率，我們依照網路拓樸的結構，設計了一個以中心點為準的虛擬機遷移策略(median-based VM migration scheme)，在這策略中，為了有效的改善資料感知(Data-Aware)之虛擬機遷移問題，我們將此問題轉換成已知的幾何問題: 尋找樹的中心點(the median finding of a tree)，在此幾合問題中，希望能從樹中找到的一中心點，此中心點到各節點的距離會是最小值，我們將此問題對應到資料感知之虛擬機遷移問題，因此我們將網路拓樸的結構轉換成樹的結構，並從這樹中找到的中心點，由原本的幾何問題的特性可得知，此中心點到各資料集的儲存節點的距離將會是最小值，但由於找到的中心點在網路拓樸的結構中，不一定會是可做為運算節點的實體機，此中心點可能為於雲端資料中心的交換器上，因此要進一步找鄰近的實體機做為虛擬機的運算節點，透過將虛擬機遷移到接近中心點的實體機，虛擬機獲得資料集的速率將大幅提升，進而提高應用程式整體的執行效率，且更容易滿足與使用者所訂下的服務水平協議(SLA)需求。

而在雲端資料中心，可能會有多個虛擬機同時需要進行遷移，且可能會有兩個或兩個以上的虛擬機皆選到同一台實體機作為遷移目標，如作為目標的實體機其資源無法負擔多台虛擬機的遷移，便產生了實體機的競爭問題，為了解決實體機的競爭問題，我們在所提出策略中加入了二分圖[2](bipartite graph)的模型，借由二分圖讓所有虛擬機盡可能的遷移的同時，也盡可能的降低了每個虛擬機的遷移成本。

二. 背景知識

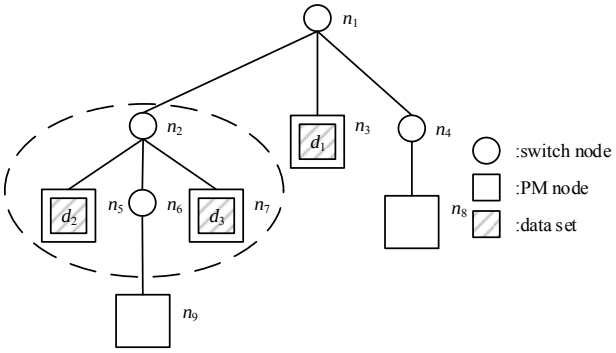
2.1. 系統模型

我們參考 Hadoop cluster[3]來設計我們的虛擬機遷移策略，Hadoop 是個可以實際在雲端資料中心執行的雲端運算系統，整個系統由兩個部分組成，主控節點(master node)以及數個作業節點(worker node)，主控節點負責管理作業節點，作業節點則是負責運行各虛擬機的實體機，且作業節點可以作為儲存節點儲存資料集。在 Hadoop cluster 的網路架構中，主控節點以及所有的作業節點是透過網路交換器(Ethernet switches)做連結的，而交換器經常採用的通信協定為生成樹協定(STP)[4]，因此，在邏輯上我們可以將 Hadoop cluster 的網路拓撲結構視為一個樹狀結構，借由樹狀結構，在遇到特定的問題時，我們便可尋找現有的幾何問題是否能夠對應，並用來解決此問題。

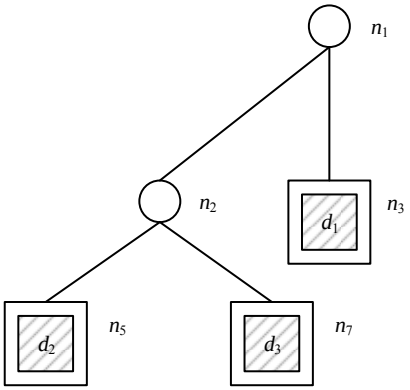
2.2. 文獻探討

虛擬機遷移已有非常多的相關研究，在[5]中，將第一合適(First Fit)演算法用在虛擬機遷移的問題中，將實體機的剩餘可用資源以 r 作表示，所有實體機的剩餘資源形成集合 R，假設一需遷移的虛擬機 v 其所需資源量為 n，第一合適在尋找實體機做遷移目標時，將從 R 中找到的第一個能夠滿足 v 所需的資源量的實體機做為遷移目標，此即為第一合適演算法，在[6]則是以最大剩餘資源做為標準，以能滿足虛擬機的需求且剩餘資源最大的實體機做為遷移目標，在[7]中提到依據實體機的工作量來動態管理虛擬機，一旦有實體機的工作量過大，便有機會違反 SLA，因此需要一個良好的實體機管理系統，能動態的找出需要將部分虛擬機遷移的實體機，以及尋找適合放置虛擬機的實體機，並依上述的為目標設計了一實體機管理系統，在[8]將 SLA 用應用程式的執行時間作為標準，當遇到需求高的 SLA 時，便需要去依照執行時間為基準來遷移虛擬機，最為適當的方法則是尋找高效能的實體機作為遷移目標，高效能的實體機能夠縮短應用程式的執行時間，這樣 SLA 便可以被滿足，然而資料中心的實體機其執行效能以及可用的資源皆不相同，又同時運作著多個虛擬機，因此在[8]所述的 SLA 的滿足方式是使用機率來表示的，假設虛擬機 v 要遷移至實體機 p 上，遷移的行為可能使的虛擬機上的應用程式有一定的機率 r 違反 SLA。

從上述的虛擬機遷移相關的方法中，發現並沒有討論到縮小獲取資料成本這方面的文獻資料，但獲取資料的成本對資料密集型的應用程式是一個非常重要的標準，因此我們便以這方向來進行相關的研究。



(a)



(b)

	$\min_{n\in T(n)} \sum_{m\in T(n)} w(m)\times d(n,m)$
n_1	(0×1)+(1×1)+(1×2)+(1×2)=5
n_2	(0×1)+(1×2)+(1×1)+(1×1)=4
n_3	(0×1)+(0×2)+(1×3)+(1×3)=6
n_5	(0×2)+(0×1)+(1×3)+(1×2)=5
n_7	(0×2)+(0×1)+(1×3)+(1×2)=5

(c)

圖.1 資料感知之虛擬機遷移範例(a)資料中心的拓撲結構(b)虛擬機 v_1 之資料獲取樹(c)以公式(2)尋找中心點

三. 虛擬機遷移策略

以中心點為基準的虛擬機遷移策略目四個要素: 收集資料集資訊、尋找中心點、決定作為目標的虛擬機以及二分圖的模組。

3.1. 收集資料集資訊

在雲端系統中，資料集是散佈在各個作為儲存節點的實體機上的，且每個應用程式有各自所需的資料集，而所需資料集的位於哪個儲存結點是可以預先知道的，在虛擬機編譯這應用程式時，編譯器便將所得到的資料集儲存位置的資訊收集好並存入應用程式的設定檔中，透過這些資訊我們便能知道資料集在整個雲端資料中心的為改進應用程式獲取資料集的時間，且能夠應對同時有多個虛擬機的遷移。此虛擬機遷移策略的組成有的所在位置。

如 2.1 所述，我們將資料中心的拓撲以樹來表示，之後我們使用資料集所在節點的資訊形成一子樹，並稱其為虛擬機 v 的資料獲取樹(data access tree)。

以圖.1 為例，圖.1(a)中，我們假設資料中心有五個實體機且透過四個交換器連接在一起，節點 n_1、n_2、n_4 及 n_6 為交換器，節點 n_3、n_5、n_7、n_8 及 n_9 為實體機，虛擬機 v_1 的應用程式所需的資料集存在節點 n_3、n_5 及 n_7 上，依據上述資訊 v_1 的資料獲取樹如圖.1(b)。

n_1為資料獲取樹的根節點，而為了考量獲取資料集的資料獲取時間，我們為此樹加入權重 $w()$，葉節點的權重 $w()$ 為 1，其餘的 $w()$則為 0。

3.2. 尋找中心點

Theorem 1: 在一虛擬機的資料獲取樹中所找到的中心點，其獲得應用程式所需的資料集的獲取成本為最小。

Proof. 首先，我們引用[9]中，對樹的中心點的定義如下。

Definition 1. 一樹 T，其中心點 n 到 T 中所有節點的距離總合為最小。

$$\min_{n\in T(N)} \sum_{m\in T(N)} d(n,m) \qquad (1)$$

$T(N)$為樹 T 的所有節點的集合，$d(n,m)$為節點 n 與 m 之間的距離也就是交換器的數量。

為了避免尋找中心點時，非儲存節點的影響，我們在這公式中加入了權重的概念，將公式變為:

$$\min_{n\in T(N)} \sum_{m\in T(N)} w(m)\times d(n,m) \qquad (2)$$

$w(m)$為節點 m 的權重，當 m 為儲存節點時，$w(m)$為 1，當 m 為交換器時，$w(m)$為 0。

由公式(2)可知，當一節點並非為儲存節點，尋找中心點時就不會將其他節點與此節點的距離列入計算，會列入計算的只有據有權重的儲存節點，透過這樣的方式，在從資料獲取樹尋找中心點時，便不會受到非儲存節點的影響。

Definition 2. 雲端資料中心的節點是透過交換器來做連接的，因此資料取成本可以用儲存節點到資料集之間經過的交換器數量來表示。

因此公式(2)所找到的中心點可讓應用程式的資料獲取成本達到最小，Theorem 1 得證。

Theorem 1 用於在樹中找到中心點，而在[9]中說明樹的結構具有良好的數學性質，可以在線性時間內找到中心點，透過這性質我們設計了一個可在線性時間內找到樹的中心點的演之演算法，如圖.2。

Input: Given a data access tree *Tv* of a VM *v*.
Output: The median of *Tv*.
1: *Q* ← empty /* *Q* is a queue.*/
2: *W_Tv* ← 0 /* The total weight of *Tv*. */
3: **for** each node *n* of *Tv* **do**
4: **if** *w*(*n*)=1 **then**
5: Add *n* in *Q*.
6: *W_Tv* ← *W_Tv*+1
7: **end if**
8: **end for**
9: **while** (*Q* is not empty)
10: Delete a node *n* from *Q*.
11: *p* ← the parent of node *n*.
12: *w*(*p*) ← *w*(*p*)+*w*(*i*)
13: **if** *w*(*p*) ≥ 1/2**W_Tv* **then**
14: *p* is the median of *Tv*.
15: **break**
16: **end if**
17: Delete node *n* from *Tv*.
18: **if** node *p* becomes a terminal node of *Tv* **then**
19: Add *p* in *Q*.
20: **end if**
21: **end while**

圖.2 可在線性時間內找到樹的中心點

3.3. 決定作為目標的虛擬機

在決定虛擬機的目標實體機前，我們有以下三個 lemma。

Lemma 1. 已知資料獲取樹的中心點在獲取資料集時具有最小的資料獲取總成本，因此，中心點也可以盡可能的減少最大的資料獲取成本。

Lemma 2. 如果虛擬機遷移到離中心點近的實體機上時，最大資料獲取成本非常有可能可以滿足在虛擬機上執行的應用程式的 SLA。

Lemma 3. 假設在虛擬機 v 上執行的應用程式其 SLA 為要在 s 個交換器的

距離內獲得資料集，如果所找到的中心點的最大資料獲取成本為 h 個交換器，則表示只要能夠在 s-h 個交換器的距離內找到可用的實體機，將此實體機作為遷移目標，則可以滿足虛擬機 v 的 SLA 需求。

透過以上的 lemma，我們可以知道，要將虛擬機遷移到能夠滿足虛擬機 v 上的應用程式的 SLA 需求的實體機，但能夠滿足 SLA 需求的實體機可能不只有一個，而是有兩個或兩個以上的候選實體機，因此我們要從這些候選實體機中找到適合擺放虛擬機的實體機來作為目標實體機。

以圖.1(a)為例，已知虛擬機 v_1 的資料獲取樹的中心點位於節點 n_2，此中心點的最大資料獲取成本為兩個交換器的距離，假設 v_1 上的應用程式的 SLA 需求為 3 個交換器的距離，在這樣的情況下，能夠滿足 v_1 的實體機必須要在圖. 1(a)中，距離節點 n_2 1 個交換器以內的範圍，因此，節點 n_5 與 n_7 可以做為 v_1 的目標實體機，如圖.1(a)中的虛線所示，而節點 n_2 與 n_6 則不能做為目標實體機，因為其為交換器而非實體機。

3.4. 二分圖的模組

在雲端資料中心中，很有可能在同一個時間有多個虛擬機要進行遷移，所以也很有可能在同一時間有兩個或兩個以上的虛擬機選到同一台實體機做為遷移目標，而一旦被選為目標的實體機其資源無法負擔將要遷移過來的虛擬機，便會有一部分的虛擬機不能做遷移，即為實體機的競爭問題，前述的方法只適合解決單一的虛擬機遷移，為了解決虛擬機競爭的問題，我們使用以下的步驟建立二分圖(Bipartite Graph) BG=((U, V), E)來表示所有可行的遷移狀況。

Definition 3. 一權重二分圖 BG = ((U, V), E)，此二分圖由兩個不同的節點集合 U 跟 V 組成，透過共有的 cost c_{ij} 建立邊(u_i, v_j)∈E，將節點 u_i ∈U 及 v_j ∈V 連接起來形成二分圖。

1) 我們假設集合 V 為那些同時要進行遷移的虛擬機的集合，v∈V 並且有自己的目標實體機的集合，我們將所有的目標實體機的集合做聯集並稱為 P。
2) V 跟 P 為 BG 中兩個不同節點集合，將所有在 V 中的虛擬機列在 BG 的左側，右側則為 P 之中所有的目標實體機。
3) 如果 p∈P 為 v∈V 的目標實體機，則在 BG 中有一邊(v, p)，將 v 跟 p 連結起來，邊(v, p)同時也與邊的權重有關，由虛擬機 v 遷移到實體機 p 的遷移成本決定，如圖 1.(c)中，虛擬機 v_1 遷移至實體機 p_1 的遷移成本為 1，而如果 v 與 p 之間沒有虛擬機的遷移關係的話則以 N/A 作為表示，遷移成本的定義如下。

Definition 4. 虛擬機 v 的遷移成本為 v 遷移到目標實體機時，所經過的交換器的數量。

有許多演算法可在多項式時間內解決二分圖問題，因此，透過二分圖我們便可盡可能的遷移虛擬機以及讓總遷移成本減少。

四. 效能評估

在此章，我們透過數個模擬實驗來評估我們所以提出的資料感知之虛擬機遷移策略，從第三章可得知，此虛擬機遷移策略是基於尋找中心點及二

分圖的模型而產生的，且能在線性時間內找到合適的實體機做為遷移目標，而我們將所提出來的虛擬機遷移策略簡稱為MDBG策略，並使用MatLab[10]來實作我們的策略並與其他幾種現有的虛擬機遷移策略做比較。

4.1. 模擬環境

在模擬實驗中，我們參考[11]在雲端運算中心散佈 100 個實體機節點，並在一個 500×500 平方單位的正方型平面上，將 50 個 rack 擺放進資料中心，這些 rack 透過交換器連接著形成一樹狀結構，rack 由部分的實體機節點組成，其中有一 rack 被設為樹根(root)，由此 root rack 組織其他的 rack 形成一個樹高為 10 的樹狀結構，建立好樹狀結構後，我們則隨機將 100 個實體機節點散佈到這些 rack 上。

基於上述的資料中心架構，我們設定了以下的參數。

- 每個實體機其可用的運算資源取三項做為代表，可用的 CPU 頻率(GHz)、可用的記憶體空間(GB)及可用的儲存空間(GB)，其資源量的區間為[(6,4,500),(12,32,3000)]，每台實體機的三項可用資源在這區間中依照亂數做決定。
- 同時要遷移的虛擬機數量從 10 個增加到 50 個。
- 每一個需要遷移的虛擬機其各資源的需求量參考 Amazon EC2[12]。
- 在模擬雲端資料中心時，我們將網路拓撲結構形成的樹的樹高設為 10，在需要遷移的虛擬機上執行的應用程式其 SLA 需求則從 SLA 區間[1, 10]隨機產生其所需要的值。
- 以上的參數每回將會執行 10 次。我們依資料獲取總成本及演算法執行時間兩個部分進行效能評估。

4.2. 模擬結果

在模擬結果中，我們將所提出的 MDBG 策略與另外三種虛擬機遷移演算法策略做比較: 最佳化虛擬遷移策略(OPT)、第一合適虛擬機遷移策略(FF)以及隨機選擇虛擬機遷移策略(RND)；OPT 將整個網路拓撲所形成的樹 $T(N)$，透過公式(2)將所有的節點都進行分析，分析當前的節點獲取所有資料集的總成本為何，再從中找到總成本最低的一個節點，讓此節點做為虛擬機遷移的目標實體機；FF 策略則是依照虛擬機的需求對節點的資源進行分析，一旦節點的資源可以滿足虛擬機的需求，就直接將此節點做為虛擬機遷移的目標實體機；RND 策略就如名字，隨機選擇一個實體機，只要符合虛擬機的資源需求，就將此實體機做為目標實體機來進行遷移，因此可以迅速的找到目標實體機；此外，OPT、FF 以及 RND 策略不像所提出的 MDBG 策略，它們並沒有有效的去解決實體機競爭的問題，如果虛擬機 i 跟 j 同時選擇實體機 p 做為它們的遷移目標，而實體機 p 卻沒有足夠的資源可以同時讓兩台虛擬機進行遷移，為了能夠解決多個虛擬機遷移的實體機競爭問題，虛擬機遷移的演算法將會重新再執行一次，會浪費些不必要的時間，進而降低整體的執行效率。

圖.3(a)中所展示的為各虛擬機遷移策略獲取資料的總成本之比較，如我們所預期的，OPT 策略其總成本為最低的，而我們所提出的 MDBG 策略也只有比 OPT 策略的總成本高出 2%，與

FF 以及 RND 兩個策略的總成本做比較，則是各自少了 46%跟 47%。

圖.3(b) 所示的為各虛擬機遷移策略執行時間的成本之比較，OPT 將會從計算中心裡找到最佳的遷移目標，但將會花費大量的找尋時間，而 MDBG 會在線性時間找到中心點並解

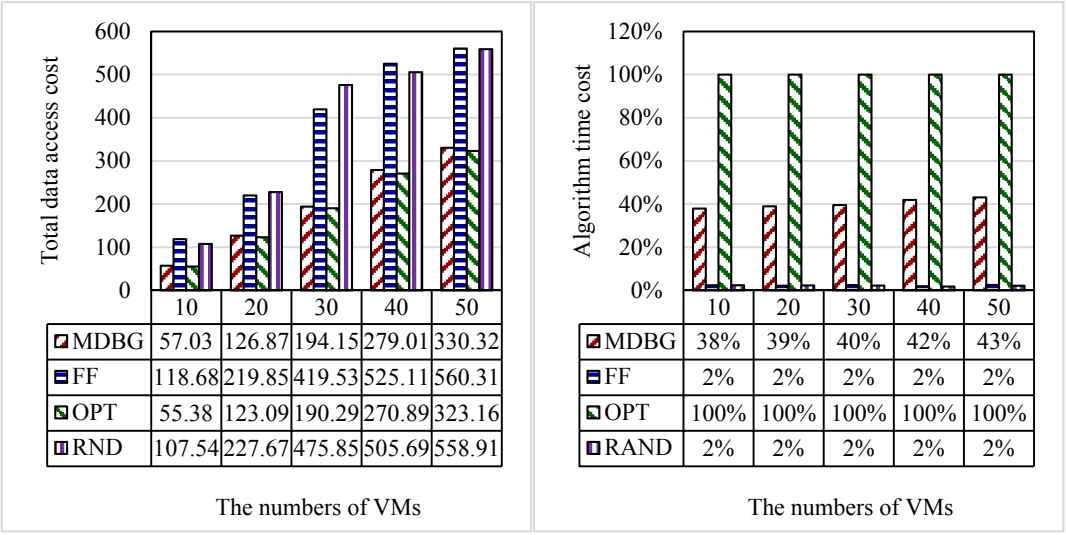


圖.3 MDBG 與其他遷移策略之比較(a)資料獲取總成本(b)演算法執行時間比較

決實體機的競爭問題，且所花費的時間大約為 OPT 的 59%，FF 及 RND 兩個演算法的執行時間則非常的低。

五. 結論

本研究計畫提出了資料感知的虛擬機搬移機制，用在雲端運算系統加快資料存取的速度，進而提升巨量資料處理的執行效率。在雲端資料中心的資料感知之虛擬機遷移問題，所提出的虛擬機遷移策略是基於尋找樹的中心點以及二分圖的模形兩個方法，並將此策略稱為 MDBG。在 MDBG 策略中，我們使用雲端資料中心的網路拓撲結構以及應用程式在執行時所需要用的資料集的散佈資訊，找到由這些資訊形成的樹的中心點，藉此獲得最佳的遷移目標，透過將虛擬機遷移到與中心點鄰近的實體機，盡可能的讓應用程式在執行時要獲得資料集的獲取時間減少，這點對於需要大量獲取資料集的資料密集類型的應用程式特別重要，再進一步的，MDBG 策略考量了多個虛擬機需要同時進行遷移的情況，這情況我們則是使用二分圖的模型來解決，藉由二分圖的模型可以讓能夠同時遷移的虛擬機的數量最大化，以及能讓每個要遷移的虛擬機其獲取資料的總成本是最小的，並在最後的模擬結果中證明 MDBG 策略可以有效的改善資料感知之虛擬機遷移問題，與最佳化遷移成本的 OPT 相比，我們只有增加大約 2%的的獲取資料總成本，與第一合適的 FF 以及隨機選擇的 RND 兩個方法相比，一個減少 46%，另一個則減少 47%的獲取資料總成本，在演算法執行時間的部分，MDBG 的執行時間大約只有 OPT 的 59%，證明此策略的可用性。

參考文獻

[1] (2014) The Xen Project, The Powerful Open Source Industry Standard for

Virtualization. [Online]. Available: http://xen.org

[2] D. B. West, *Introduction to Graph Theory*, 2nd ed., Prentice Hall, 2000.

[3] (2014) Hadoop - Apach Hadoop. [Online]. Available: http://hadoop.apache.org

[4] IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges, IEEE 802.1D Std., 2004.

[5] M. Bichler, T. Setzer, and B. Speitkamp, "Capacity Planning for Virtualized Servers," in *Proc. WITS*, vol. 1, Nov. 2007.

[6] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application Performance Management in Virtualized Server Environments," in *Proc. NOMS*, pp. 373-381, Apr. 2006.

[7] M. Andreolini, S. Casolari, M. Colajanni, and M. Messori. "Dynamic Load Management of Virtual Machines in Cloud Architectures," in *Proc. Conf. CloudComp*, vol. 34, pp. 201-214, 2010.

[8] S. Das, M. Kagan, and D. Crupnicoff, "Faster and Efficient VM Migrations for Improving SLA and ROI in Cloud Infrastructures," in *Proc. Int. Teletraffic Congress*, Sep. 2010.

[9] S. K. S. Gupta and P. K. Srimani, "Adaptive Core Selection and Migration Method for Multicast Routing in Mobile Ad Hoc Networks," in *Proc. IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 1, pp. 27-38, Jan. 2003.

[10] S. Kavulya, J. Tany, R. Gandhi, and P. Narasimhan, "An Analysis of Traces from a Production MapReduce Cluster," in *Proc. IEEE International Conference on Cluster, Cloud and Grid Computing*, pp. 94-103, May. 2010.

[11] (2014) MathWorks - MATLAB and Simulink for Technical Computing. [Online]. Available: http://www.mathworks.com

[12] (2014) Amazon Web Services - Amazon Elastic Compute Cloud (EC2). [Online]. Available: http://aws.amazon.com/ec2