雲端主機之資料庫平衡負載

羅有隆　　賴民軒
朝陽科技大學　　朝陽科技大學
yllo@cyut.edu.tw　s10014618@cyut.edu.tw

摘要

網際網路的盛行帶動了雲端運算平台的發展，更導致了資料儲存管理機制在雲端運算的應用上逐漸邁向成熟。在雲端平台上每個資料庫主機往往必須同時服務多個資料庫應用系統，但在主機的資源有限下，就必須對所儲存的資料量，以及所需負擔的資料庫數目給予必須的考量，以期能夠達到能夠達到負載平衡，避免工作分配不均的情況下，造成服務品質不佳。然而，如何考量資料量與資料庫數目的均衡分配到各個主機，目前則尚未受到廣泛的關注。本研究報告，設計了五個將資料庫分配到主機的方法與理想分配差異的偏離公式，並透過實驗與偏離值的分析，來了解各個資料庫分配方法的優劣。而從我們的實驗結果來看，所提出的方法中，最好的分配，幾乎可以媲美最佳解的均衡分配方式。

關鍵詞：雲端平台、資料庫置放、平衡負載

Abstract

Cloud data storage management and applications have become more and more mature as Cloud computing is developed rapidly. Each database host in the cloud platform often has to service more than one database application system, however under the resource limitations of the host, evenly distributed databases into each sever is a big issue which has to be addressed. The database sizes and the number of databases must be taken into account for workload balance among database hosts. If too many data or databases are concentrated in only one or few database servers, the data skew occurs and may result poor quality of service. Currently, how to evenly allocating databases into hosts has not been concerned yet. In this paper, we will propose five database allocation approaches for distributing databases into hosts in the cloud platform. The equations used to evaluate the deviation of distribution results comparing to ideal are also provided in this report. After experimental study, the best one of our approaches can compete to the optimal solution for evenly distributed databases into hosts.

Keywords: cloud platform, database allocation, load balancing

1.前言

近來，雲端運算(cloud computing)應用受到高度的關注[7][12][18][19]，它利用"As A Service：AAS"的網路技術，以大量的運算或儲存資源(服務節點)提供使用者多元化的服務[21][24]。儲存雲端服務(Storage as a Services, StaaS)是雲端的重要服務項目之一，所有重要的應用程式及資料庫的服務都集中於此，其主要是提供線上儲存空間功能，讓使用者透過 Web-based 應用程式不論在任何地點、任何時間，透過任何可上網的裝置就可以方便的使用其儲存功能，而不需要另外建置及管理專屬的儲存設備系統，如圖 1。

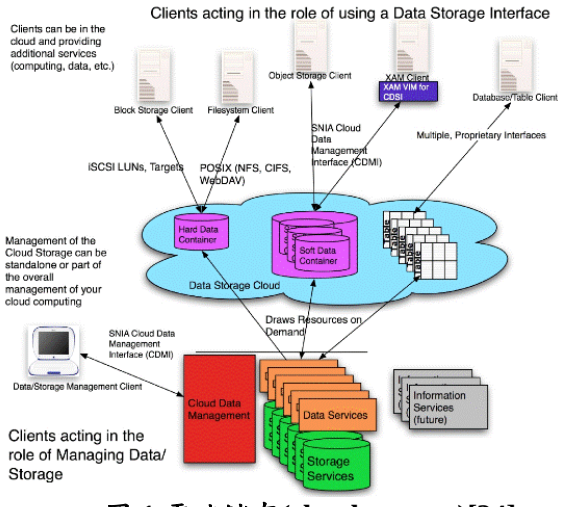

圖 1 雲端儲存(cloud storage)[24]

雲端儲存的核心即為儲存層，就如同雲端運算中的基礎設施即服務(Infrastructure as a Service, IaaS)，是由分散在不同區域的各類型儲存設備所組成，不論是 DAS，抑或 FC SAN、iSCSI 或 NAS 等 IP 儲存設備，皆可透過支援儲存虛擬化(storage virtualization)技術的集中化(consolidation)管理系統整合在一

起。透過管理系統，可以進行所有儲存設備的遠端監控、管理等作業，且必須達到不同儲存設備間的協同運作，並提供單一整合服務。在使用上，使用者不論身在何處或任何時間，只要透過 Web-based 應用程式，即可上網直接存取資料。由於雲端資料的龐大與多樣性，在雲端平台(cloud Platform)上的資料庫，是分散在眾多的主機之中，以服務成千上萬的使用者，且必須具有足夠的擴充性(scalable)與管理(management)能力，以即時面對任何特定的資料需求。因此，已有許多研究報告提出了在雲端平台上，如何做有效率的資料庫管理[6][10][11][13][16]。

一資料庫應用系統於資料庫伺服器(database server)中運作，以服務眾多的使用者，是相當耗費資源的，除了資料儲存空間與記憶體的需求外，還需要啟動資料庫引擎(database engine)來服務與管理資料庫系統。而實際的應用上，又往往是一實體主機同時服務多個資料庫應用系統，因此，如果過多的資料庫應用系統或過多的資料量同時於一主機中運作，共同分享資源，在主機資源有限的情況下，則各個資料庫運作效能將受到影響。然而，如何將資料庫大小不同的資料庫應用系統，有規劃的分配到雲端平台之各主機，以使得各主機所服務之資料庫應用系統之數目相近，而且所管理的總資料量亦相去不遠，達到主機間之平衡負載，使之有效率的服務眾多的使用者，至目前則是尚未受到討論。本研究報告將探討如何有效率的將資料庫分配到各個實體主機中，以求達各主機之負載平衡。

本篇研究報告分為五個主要章節，除了第一章前言，對雲端架構做一簡單介紹，並提出我們的研究目的外，第二章將對現有與本研究相關之文獻做一摘要性的敘述，接著於第三章中說明我們所設計的將資料庫分配到主機的五個方法，並於第四章對我們的實驗結果進行分析討論。最後則是我們的結論。

2.文獻探討

雲端科技是利用虛擬化以及自動化等技術來創造和普及電腦中的各種運算資源，透過虛擬化技術將可以快速滿足使用者的使用需求，既不用去購買、安裝、配置其他新的設備。因此，雲端運算提供了三種服務類型，分別為：架構即服務(Infrastructure as a Service, IaaS)、平台即服務(Platform as a Service, PaaS)、軟體即服務(Software as a Service, SaaS)[24]。雲端運算是種能夠將動態伸縮的虛擬化資源，透過網路以服務的方式提供給使用者的運算服務，使用者不需要知道如何管理那些支援雲端運算的基礎設施，直接從雲端取得需要的功能，並取得完整的線上支援服務。

雲端的架構包含有全域主節點主機(global master node)與區域節點主機(local slave node)，而在資料庫的應用裡，區域節點主機主要是由實際存放資料的資料節點(data node)主機所組成。而虛擬化服務中之伺服器虛擬化，是將一台實體機器切分成多台伺服器，每台切 割分出之伺服器皆可執行各自致力之工作。因此，一台實體機器就可以服務多個資料庫應用系統，而一台實體機器的負載，就必須要有所儲存的資料量，以及所需負擔的資料庫數目的考量，在主機所能負擔的範圍內，發揮最大的效能，服務眾多的使用者。而如何讓構成雲端服務的龐大實體主機群中的每一主機獲得均衡的負載，至今尚未獲得討論。較類似的研究為 Mackey 等的研究報告探討了由許多小檔案(small files)所構成的詮釋資料(metadata)，如何分配到資源有限的雲端的資料節點(data nodes)中[9]。在這一章中，我們也將介紹 Best Fit Decreasing Strategy 之資料分配技術[8][11][20]，它將被我們應用於此研究中，以安排資料庫到資料節點的主機中。

2.1 Metadata Management for Small Files

Mackey 等在[9]研究報告中，提出如何在 Google 的 MapReduce[2] 與 Yahoo 的 Hadoop[26]兩應用於雲端之檔案管理系統中，有效率的管理眾多小檔案的詮釋資料(metadata)。檔案系統的管理是由一詮釋資料伺服器(metadata server)稱之為 Name Node 連結相當多數量的 I/O 節點稱之為 Data Nodes，如圖 2 所示。詮釋資料伺服器負責管理詮釋資料與檔案間的互動，資料節點負責資料的讀/寫與檔案的複製。詮釋資料必須存放在記憶體中，往往會造成伺服器處理的瓶頸。因此，系統限制了每位使用者的使用空間與檔案數。如圖 3，每個使用者被允許使用 7 個檔案(N=7)，以及 7GB 的空間(S=7GB)，而 User 2 與 User *n* 新進的檔案因已超過檔案數或空間大小，而不被允許。報告中也提出了將數個小檔案合併成一個檔案以讓檔案數減少(reduce)的方法，以及動態的允許使用者將超過限制的檔案處理完等的措施，以提高詮釋資料伺服器的效能。

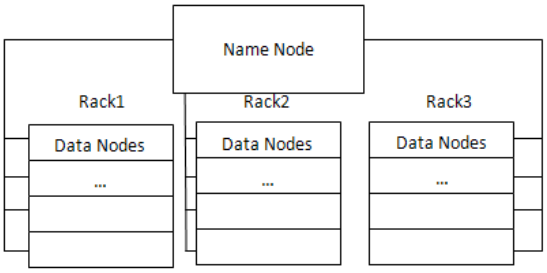

圖 2 詮釋資料伺服器連結多個 I/O 節點[9]

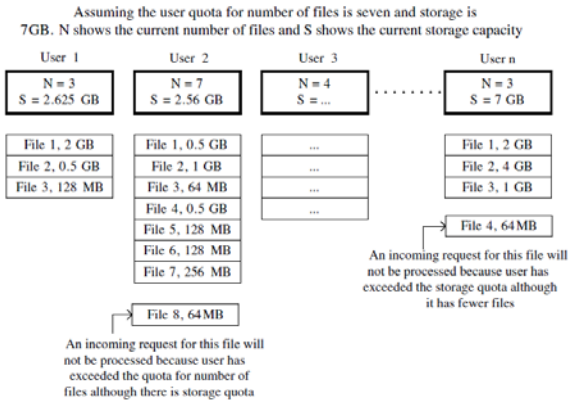

圖 3 檔案系統中每位使用者的檔案數與空間的限制 [9]

2.2 Best Fit Decreasing Strategy

當有許多不同大小的檔案或資料庫，希望能均勻的分配到幾部主機上時，這是個與裝箱(bin packing)演算法[4]類似的 NP-complete 問題[1]。Best Fit Decreasing Strategy[5]是質量大的優先的一種策略，它是目前被認為用來處理裝箱問題，並可以得到最接近最佳解的方法。Best Fit Decreasing Strategy 需要先將檔案(或箱子)由大到小排序，之後依續將它們置放到主機中，而每次皆放置到目前擁有最少資料量的主機，直到所有的檔案都分配結束。一個簡單的範例說明於圖 4，圖中排序好的檔案由大到小為 B1、B2、…、B8，依 Best Fit Decreasing Strategy 將此八個檔案置放至 P1 與 P2 兩主機中，B1 放入 P1 後，P2 所擁有的資料量小於 P1，因此 B2 被置放於 P2 中，而此時 P2 所擁有的資料量仍然小於 P1，再次將 B3 放入 P2 中，如果 P2 所擁有的資料量還是小於 P1，則繼續將 B4 也放入 P2，餘此類推。類似的方法，被廣泛的應用，如在[8][11][20]各不同的研究報告中。而我們這篇研究報告，也將應用 Best Fit Decreasing Strategy 來設計分配方法，用以分配資料庫至雲端的資料庫主機(資料節點)中，以期望可以均勻分配各主機的工作負載。

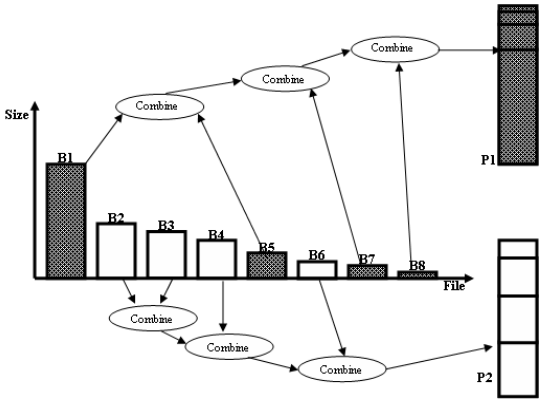

圖 4 Best Fit Decreasing Strategy

2.3 Round-Robin Scheduling

Round-Robin Scheduling 是使用在 CPU 排程用途上[14]，在許多的用途上也都是有運用到 Round-Robin Scheduling 類似的方法[15][3][23]，不管是使用在平行資料庫分配使用，或者是在 CPU 排程用途上，Round-Robin Scheduling 都是一個非常基本，更是非常實用的方法。如圖 5 [23]是 Round-Robin Scheduling 的使用方式，CPU 起點從編號 0 的工作開始執行，之後往編號 1 的工作執行，餘此類推，當執行完編號 5 的工作之後又重新從編號 0 的工作開始執行。這樣的方法固然簡單，卻是最能有效均勻分配 CPU 執行時間的演算法。如果將 Round-Robin Scheduling 的概念，運用在分配資料庫給主機的話，資料庫依序分配給第一到最後一部主機，再從重新從第一部分配起，直到所有資料庫都分配完為止。如果不考慮資料庫大小，就主機所分配到的資料庫庫數目來說，一定會是最佳解。

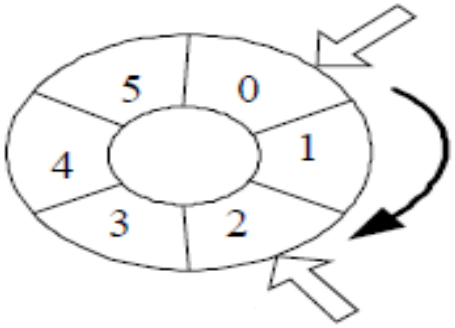

圖 5 Round-Robin Scheduling [23]

3. 資料分配方法設計

每個資料庫的執行，都需要啟動一個資料庫引擎(database engine)來為其服務與管理，但如果一部實體主機要服務的資料庫太多，必須啟動很多的資料庫引擎，在主機資源有限的情形下，會影響到主機的效能。同樣的，主機如

果被分配到太多的資料庫資料，對資料庫的操作，也是要耗費較多的系統資源與時間。因此，我們希望每個主機都盡量被分配到相近的資料大小，以及差不多一樣數目的資料庫，以使每部主機的工作負載量，可以相近，避免有瓶頸節點的產生。我們改進先前於[27]所提出的四種分配方式，考量資料庫大小與數量，先將各資料庫依資料量由大到小排序，以做為分配到主機時的順序，並重新設計了以下五種資料庫配置的方法，用來探討分配的均衡狀況。

3.1 Round-Robin Allocation

Round-Robin Allocation 是利用 Round-Robin scheduling 的概念，將多個資料庫依序從第一部主機分配到最後一部主機，然後循環的再從第一部主機開始，將尚未分配的資料庫分配到各主機，一直到所有資料庫分配完畢為止，圖 6 為簡單的分配表示。此分配方式可以保證主機所分配到的資料庫數目，是在最佳解的狀況。但是，卻無法確保各主機所分配到的總資料量，會是均衡的。

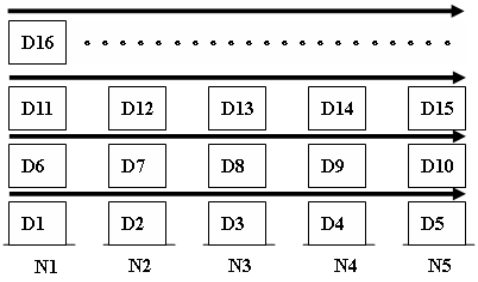

圖 6 **Round-Robin Allocation**

3.2 Z-Distribution

Z-Distribution 是將資料庫依序從第一部主機分配至最後一部主機，接著再從最後一台主機分配回來第一部主機，以'Z'字型反覆的執行到分配完所有資料庫為止為止，如圖 7 所示。此方法能比 Round-Robin Allocation 將資料更均衡的分配到主機，以避免較多資料庫量分配至第一台主機。

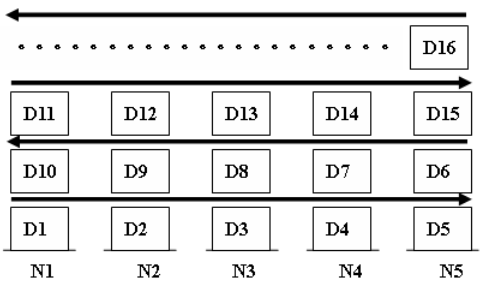

圖 7 Z-Distribution

3.3 Best Fit Decreasing Strategy with Size First

此方法是應用 Best Fit Decreasing Strategy[5]，將資料庫依資料量(size)由大到小做排序，然後依序將目前資料量最大的資料庫，分配到目前擁有資料量最少的主機中。若有資料量一樣的多部主機，則以擁有資料庫數目少的優先被分配。此方法優先考量的是讓每部主機擁有最接近的資料量負載，但無法保證每部主機所被分配到的資料庫數目是最相近，也就是無法避免可能有部份主機被分配到為數較多的小型資料庫。

3.4 Best Fit Decreasing Strategy with Limited Number of Databases

此方法相似於 Best Fit Decreasing Strategy with Size First，但是限制了資料庫分配到主機的數量不能高於平均數目，例如：總資料庫數 100 個、主機 10 台，則平均分配到各主機應該為 10 個資料庫，我們據此將每台主機可分配到的資料庫數限制為 10，滿了則再在分配，而改分配到目前資料庫數目未滿，且被分配到資料量最小的主機。我們希望此方法可以有如同 Round-Robin Allocation 與 Z-Distribution 一樣的均勻分配資料庫數目，但各主機所分配到的資料量能較前兩者來得均衡。

3.5 Quantity Ratio Allocation

Quantity Ratio Allocation 的設計概念為，因資料庫依資料量大小的順序依序的做分配到主機的考量，當兩主機已分配有相同數目的資料庫時，應優先分配給擁有資料量少(或與平均應分配量差距大)的主機，以達資料量的平衡。或當兩主機被分配有相同資料量時，應優先分配給已擁有較多資料庫數(或與平均應分配資料庫數差距小)的主機，理由是資料庫大小是排序好的，如此可以將後面較小的多個資料庫，分配給目前資料庫數少的主機，以達平衡分配之目的。因此，我們設計了公式(1)，以計算出 Quantity Ratio (QR)，以做為分配資料庫的依據。 公式中，NDB_{avg} 是每台主機所應該分配到的資料庫數目平均值，也就是設定的資料庫總數目除以設定的總主機數，NDB_i 則是目前第 i 台主機所分配到的資料庫數目，當相減之後則是第 i 台主機的剩餘能置放的資料庫數目。DB_{avg} 是表示每台主機應該被分配到的資料庫資料量平均值，則為設定的資料庫

總資料量除以設定的主機總數，DB_i則是目前第 i 台主機所分配到的資料庫資料量，當與 DB_{avg} 相減後則為第 i 台主機可以置放的剩餘資料庫量。如此，資料庫將優先分配到 *QR* 值大的主機。而此設計，我們仍然將其資料庫數目限制為平均數量，也就是主機中資料庫數目達到平均數量，將不再被分配，如此，公式(1)中之分母可以避開值為 0 的情形出現。

$$\mathrm{QR} = \frac{DB_{avg} - DB_i}{NDB_{avg} - NDB_i} \quad \ldots (1)$$

以範例說明：假設 A 主機被置放單一個資料庫量 100GB，資料庫數目為一個，D 主機置放五個資料庫，資料量為 90GB 資料庫數目為五個。如果希望每個主機能被平均分配到 10 個資料庫與 150GB 的資料量。如此，A 主機尚餘 9 個資料庫數目與 50GB 資料容量可以繼續分配資料庫進來，則 *QR* 值為 50/9=5.6。而 D 主機尚餘 5 個資料庫數目與 60GB 資料容量可以繼續分配資料庫進來，則 *QR* 值為 60/5=12。如此，下一個資料庫會被分配到 D 主機。這樣分配方式能盡量讓較大的資料庫量與較小的資料庫量分配在同一台主機上。

4.實驗分析

為了瞭解我們所設計的五個資料分配方式的優劣，我們參考[27]的實驗模型，重新設計更理想的公式，設計了一系列的模擬實驗，以做為討論分析。

4.1 實驗模型

在接下來的模擬實驗中，我們設計有 1000 個資料庫，總共有 100TB 的資料量，將被分配到 100 個資料節點主機。又假設每個資料庫所擁有的資料量並不相等，而是呈現常態分配(normal distribution)，使得每個資料庫有大有小，其大小的分佈情形，將由齊夫定律(Zipf-like distribution 函式) [17][22] 所模擬出來，而它的計算函式如公式(2)所示。

$$|B_i| = \frac{|R|}{i^{Z_b} \sum_{j=1}^{b} \frac{1}{j^{Z_b}}} \quad \ldots (2)$$

公式中，$|B_i|$代表第 i 個資料庫的資料量；|R|指加總所有資料庫的總資料量(實驗中為 100TB)；b 為資料庫的總數目(實驗中為 1000)；Z_b 為資料歪斜(skew)程度的係數，其值介於 0.0 與 1.0 間。當 Z_b=0 時，每一資料庫將擁有一樣大的資料量，而當 Z_b=1 時，代表資料庫大的很大，小的很小，其差異非常的大。例如：圖 8 是將歪斜程度(Z_b)設為 0.0(綠)、0.5(藍)與 1.0(紅)來模擬 1000 個資料庫的大小分佈情形。其中，資料歪斜程度為 0(Z_b =0)時，每個資料庫大小都相同為 100GB。而在資料歪斜非常嚴重的情形下(Z_b =1.0)，最大的一個資料庫擁有約 13TB 的資料，而最小的資料庫則約有 13GB 的資料，大小差距約有 1000 倍。

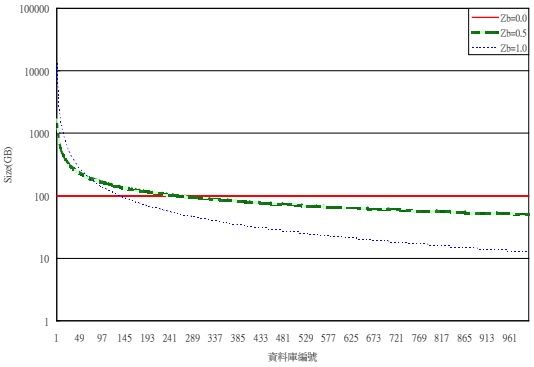

圖 8 Zipf-like distribution 分佈情形

此外，為了比較各分配方法將資料分配得是否均勻，讓各主機在總資料量與資料庫數量，可以有相近的平衡負載，我們設計了公式(3)、公式(4)與公式(5)，以用來計算分配結果偏離了理想的均衡負載有多大的距離。

$$D_{size} = \sum_{i=1}^{n} \left| DB_i - DB_{avg} \right| \quad \ldots (3)$$

$$N_{NoOfDB} = \sum_{i=1}^{n} \left| NDB_i - NDB_{avg} \right| \quad \ldots (4)$$

$$DR = \sqrt{\left| \frac{D_{size}}{D_{total}} \right|^2 + \left| \frac{N_{NoOfDB}}{N_{total}} \right|^2} \quad \ldots (5)$$

其中，公式(3)中，D_{size} 為總資料量偏離距離，亦即每部主機實際被分配到的資料量與主機平衡負載該有資料量的差，取絕對值之加總；DB_i 為第 i 部主機被分配到的資料量，DB_{avg} 為每部主機平衡負載該有的平均資料量，n 則為主機的數目。公式(4)中，N_{NoOfDB} 為資料庫數目的偏離距離，也就是每部主機實際被分配到資料庫數目與平衡負載資料庫數目的差，取絕對值之加總；NBD_i 為第 i 部主機被分配到的資料庫數目，NDB_{avg} 是各主機平衡負載的資料庫

數目，n 仍然為主機的數目。公式(5)中，DR 為總偏離率(Deviation Ratio)，D_{total} 為總資料量，N_{total} 為總資料庫數目。因此，公式(5)總偏離率 DR 是考量以資料量偏離率與資料庫數目偏離率為座標之二維空間距離，其值介於 0 與 $\sqrt{2}$ 之間，值越小表示偏離情況越輕微，值為 0 時表示均衡分配完全沒有偏離。DR 值同時考量了資料庫量與資料庫數目，可以更具體用來的判斷資料庫分配方法的優劣。

實驗參數整理於表 1。此外，每個資料庫大小$|B_i|$將取決於歪斜程度 Z_b，主機分配到的資料量 DB_i 與資料庫數目 NDB_i，將由實驗決定，而主機平衡負載之資料量 DB_{avg} 與資料庫數目 NDB_{avg} 由計算可得，分別為 1TB 與 10 個資料庫。實驗過程將資料庫資料量歪斜程度(Z_b)由 0.1 到 1.0 的變化，透過 Round-Robin Allaction(簡稱 Round-Robin)、Z-Distribution、Best Fit Decreasing Strategy with Size First(簡稱 BestFit_Size)、Best Fit Decreasing Strategy with Limited Number of Databases (簡稱 BestFit_NDB)、以及 Quantity Ratio Allocation (簡稱為 QR)等五個配置法逐一將資料庫分配給主機，以了解各分配方法的優劣。每個實驗都是逐步提高資料庫資料量的歪斜程度(Z_b)，由 0.1 到 1.0，以檢視各資料庫分配方法的能力與表現。實驗結果與討論分析於下面各節。

表 1 實驗參數

參數名稱	參數設定
主機(n)	100 台
資料庫數目(b)	1000 個
資料庫總資料量($\|R\|$)	100TB
歪斜程度(Z_b)	0.1~1.0

4.2 資料量偏離分析

資料量偏離分析，是探討各主機被分配到的資料量偏離平衡負載的情形，分配結果在經過透過公式(3)的統計，實驗結果呈現於圖 9，圖中縱軸是總資料量偏離距離 D_{size} 的值。由圖 9 我們發現，BestFit_Size (Best Fit Decreasing Strategy with Size First)因為就是優先從均衡分配資料量來考量，它表現得最好，資料量偏離距離最小。表現次佳的為 BestFit_NDB (Best Fit Decreasing Strategy with Limited Number of Databases)與 QR (Quantity Ratio Allocation)，兩者的偏離曲線幾乎重疊，相當的接近。 而 Round-Robin Allocation 表現得最差，因為它只專注於均衡分配資料庫數目，完全沒有考慮資料量要均衡配。Z-Distribution 雖然保留了 Round-Robin 均衡分配資料庫數目的優點，而嘗試改善它的缺點，但相較於其他分配方式，它的改善幅度是有限的。

此外，五個方法都受資料庫大小歪斜程度(Z_b)的影響，曲線逐步上升，代表歪斜程度高，要平衡主機的負載就越來越難。甚至在 Z_b=1.0 時，資料量偏離距離會接近 60TB 或以上，其原因我們可以由圖 8 的 Zipf-Like 分佈情形與表 2 的最大資料庫與最小資料庫來分析說明。表 2 中，當歪斜程度 Z_b=0.0 時，表示最大資料庫與最小資料庫所擁有的資料量都是相同的，皆為 100GB。但是當 Z_b 越來越大時，最大資料庫與最小資料庫的資料量差異也越來越大，在 Z_b=1.0 時，最大資料庫為 13.36TB，但是最小資料庫為 0.013TB(=13GB)，兩者差異超過 1000 倍。無論用何種分配方式，一主機若被分配到最大的資料庫，其資料量就遠大於負載平衡時的 1TB，也已經產生了超過 12TB 的偏離距離，無怪乎 100 部主機加總的偏離距離會接近 60TB 或以上，甚至 Round-Robin 在 Z_b=1.0 時，更超過了 70TB。

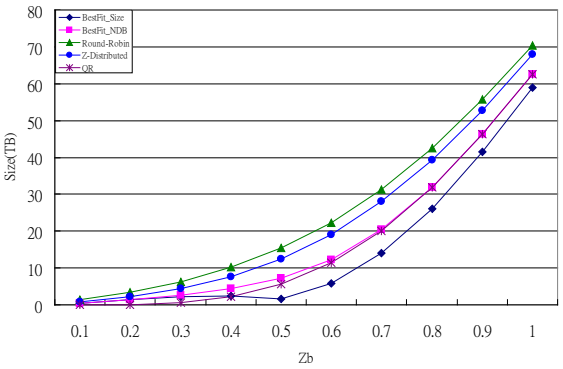

圖 9 資料量偏離分析

表 2 Zipf-like distribution 最大與最小資料庫

Z_b	最大資料庫(TB)	最小資料庫(TB)
0.0	0.10	0.100
0.1	0.18	0.090
0.2	0.32	0.080
0.3	0.56	0.070
0.4	0.96	0.061
0.5	1.62	0.051
0.6	2.65	0.042
0.7	4.22	0.034
0.8	6.46	0.026
0.9	9.50	0.019
1.0	13.36	0.013

4.3 資料庫數目偏離分析

資料庫數目偏離分析，探討各主機被分配到的資料庫數目偏離平衡負載的情形，分配結果再透過公式(4)的統計，實驗結果呈現於圖 10。平衡負載的最佳情況下，每一主機被分配到的資料庫數目應該為 10。從圖 10 中我們可以看出，除了 BestFit_Size 優先從均衡分配資料量來考量，而沒有考慮均衡每個主機的資料庫數目，因此表現較差外，其他四個方法都能把資料庫數目均衡的分配到各主機，使得資料庫數目偏離值(N_{NoOfDB})為 0。

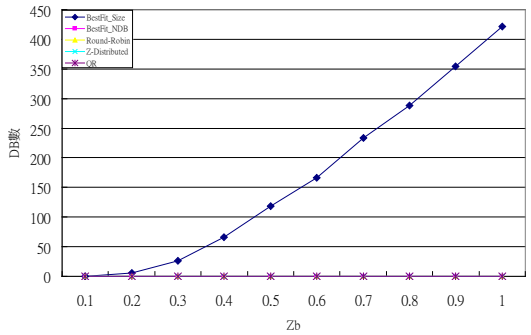

圖 10 資料庫數目偏離分析

4.4 總偏離率分析

以上兩種實驗皆顯示單方面考量的平衡負載偏離情形，如果 資料庫資料量以及資料庫數目則都是非常重要的因素，則應該將兩種因素結合在一起同時考量，公式(5)因此而被設計出來。公式(5)以考量資料庫資料量與資料庫數目之偏離率做為二維座標，計算其與原點的距離來表示總偏離率，距離越小，表示分配得越均衡。此實驗比較了五種分配方法的總偏離率，實驗結果呈現於圖 11 中。從圖 11 中，BestFit_NDB 與 QR 對資料庫的平衡分配效果最好，而且兩方法的成效相當接近。其次則是 BestFit_Size 與 Z-Distribution，兩方法的平衡效果也是相當的接近。而 Round-Robin 則是表現最差的，主要還是資料量無法做適當均衡的分配考量，光靠均衡的資料庫數目是無法有效降低總偏離率。此外，BestFit_NDB 在資料量歪斜程度(Z_b)超過 0.9 之後，總偏離率越來越高，甚至在 1.0 時是五個方法中最差的，主要是因為，資料庫的資料量差異過大，也造成資料庫數目分配的嚴重失衡，因此造成總偏離率不佳的結果。

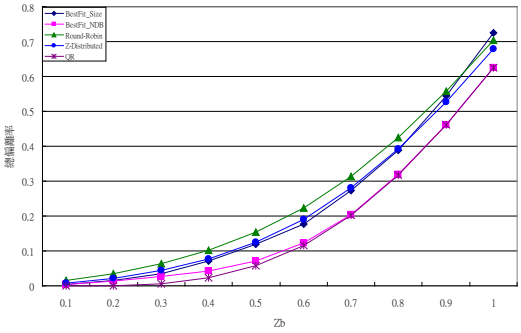

圖 11 總偏離率分析

4.5 資料庫數目的影響

在此實驗當中，我們使用當總資料庫數目不同的情況下，探討對五個分配方法的影響，表 3 是我們所設定的實驗參考數據，除了歪斜程度(Z_b)固定設為 0.5，以及資料庫數目由 500 變化到 1300 個外，其餘實驗參數設定皆與前面的實驗相同。實驗結果呈現在圖 12 中，由圖可觀察到，所有方法都會受到資料庫數目的增加，使得總偏離率都呈現縮小的趨勢，這是因為當資料的總量是固定的時候(100TB)，資料庫數目增加，意味著每個資料庫的資料量會減少，這是有益於資料庫的均衡分配。此實驗結果仍然是以 BestFit_NDB 與 QR 對資料庫的均衡分配效果最佳，且 QR 又略優於 BestFit_NDB。而其他三個方法表現較為遜色，此外，當資料庫數目增加，對 BestFit_Size 分配方式的幫助更明顯於其他兩方法，當資料庫數目逐漸增加後，BestFit_Size 也逐漸優於 Round-Robin 與 Z-Distribution。

表 3 各資料庫數量實驗參考

參數名稱	參數設定
主機(n)	100 台
資料庫數目(b)	500 個~1300 個
資料庫總資料量($\|R\|$)	100TB
歪斜程度(Z_b)	0.5

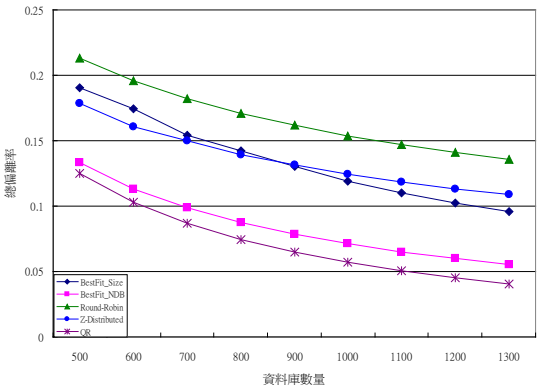

圖 12 資料庫數目的影響分析

4.6 最佳解的探討

從前面的實驗結果，BestFit_NDB 與 QR 兩分配效果最好，而 QR 又略優於 BestFit_NDB，但是是否還有改善的空間？ 為了解持續改善的可能性，本節中我們探討了 QR 的總偏離率與最佳解有多少的差異。但是，為此問題尋求最佳解是個 NP-hard 問題[4]，如果以前面的實驗參數來進行，必須考量所有可能組合才能求得最佳解，當每個資料庫都有 100 個主機可以選擇時，以 1000 個資料庫，則所有可能組合將有 100^{1000} 種可以考量，這若在現在一般電腦上執行，幾乎是不可能的任務。 因此，我們降低了參數的值，僅以 20 個資料庫數目與 4 部主機來進行，其可能組合仍然達到 4^{20}(=1099511627776)種組合。資料庫的總資料量則設定為 10TB，QR 分配與最佳解的總偏離率實驗結果如圖 13 所示，我們發現兩者的曲線相當接近，幾乎重疊。由此實驗的結果表示，如果沒有突破性的思考，僅考慮資料庫如果分配到主機，已經沒有什麼改善的空間。

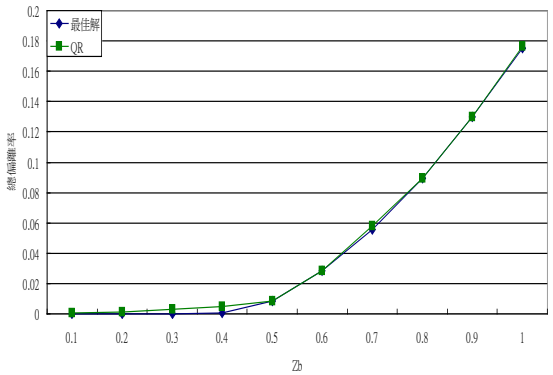

圖 13 QR 與最佳解的比較

5.結論

儲存雲端服務(Storage as a Services, SaaS)是雲端平台的重要服務項目之一，所有重要的應用程式及資料庫的服務都集中於此。在雲端平台(cloud Platform)上的資料庫，是分散在眾多的主機之中，以服務成千上萬的使用者。一資料庫應用系統於資料庫伺服器(database server)中運作，以服務眾多的使用者，是相當耗費資源的，除了資料儲存空間的需求外，還需要啟動資料庫引擎(database engine)來服務與管理資料庫系統。而實際的應用上，又往往是一實體主機同時服務多個資料庫應用系統，因此，如果過多的資料庫應用系統或過多的資料量同時於一主機中運作，共同分享資源，在主機資源有限的情況下，則各個資料庫運作效能將受到影響。本研究報告設計了五個資料庫分配的方法─Round-Robin Allaction、Z-Distribution、Best Fit Decreasing Strategy with Size First、Best Fit Decreasing Strategy with Limited Number of Databases、以及 Quantity Ratio Allocation，用於將資料庫分配到各個實體主機中，希望在資料量與資料庫數目可以達各主機之負載平衡。而我們經過實驗驗證，如果主機有資源上的嚴格限制，運作著重在資料庫數目的均衡分配，而不考量資料量的平衡負擔，則除了 Best Fit Decreasing Strategy with Size First 外，其他四個分配方法皆可以達到最佳的效果。而如果在主機資料容量必須嚴格控管的情形下，Best Fit Decreasing Strategy with Size First 與 Quantity Ratio Allocation 會有不錯的分配效果。但是，如果在資料庫數量與資料量都必須考量的情形下，我們設計了結合資料庫數目偏離率與資料量偏離率的總偏離率計算公式，來計算資料庫分配結果與完全均衡的距離，具次判斷分配的優劣。而我們實驗結果顯示，Quantity Ratio Allocation 可以把資料庫分配得最均衡，其次是 Best Fit Decreasing Strategy with Limited Number of Databases，但很接近 Quantity Ratio Allocation。最後我們也探討所設計的 Quantity Ratio Allocation 分配方法與最佳解的差異，發現兩者是相當接近的。

參考文獻

[1] S.A. Cook, “The Complexity of Theorem Proving Procedures,” *In proceedings of 3rd Annual ACM Symposium on the Theory of*

Computing, New York: ACM. 1971: 151-158.

[2] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” *In Proceedings of the 6th Symposium on Operating System* Design and Implementation (OSDI 04), Usenix Assoc., pages 137–150, 2004.

[3] D.J. DeWitt, J. Gray, “Parallel Database Systems: The Future of High Performance Database Systems,” Comm. ACM 35 (6), 85 - 98, 1992.

[4] M.R. Garey and D.S. Johnson, Computers and Intractability: *A Guide to the Theory of NP Completeness*, Freeman, San Francisco, 1979.

[5] D.S. Johnson, “Near-optimal bin packing algorithms,” *Ph.D. Thesis, MIT*, Cambridge, MA, 1973.

[6] J.L. Johnson, “SQL in the Clouds,” *Computing in Science and Engineering*, pp. 12-28, July/August, 2009.

[7] Y. Kakuda, H. Yukitomo, S. Kusumoto, and T. Kikuno, "Scientific Computing in the Cloud, " *IEEE Design & Test*, Vol. 12, Issue 3, IEEE Computer Society Press, pp. 34-43, May 2010.

[8] M. Kitsuregawa and Y. Ogawa. “Bucket spreading parallel hash: A new, robust, parallel hash join method for data skew in the super database computer (SDC),” *In Proc. of 16th Int ’ l Conf. on VLDB*, pages 210-221, Brisbane, Australia, August 1990.

[9] G. Mackey, S. Sehrish and J. Wang, "Improving metadata management for small files in HDFS," *CLUSTER ‘09. IEEE International Conference on Cluster Computing and Workshops*, September, 2009.

[10] V. Mateljan, D. Cisic, and D. Ogrizovic, “Cloud Database-as-a-Service (DaaS) – ROI,” *proceedings of the 33rd International Convention MIPRO*, pp. 1185-1188, May 2010.

[11] Z. Mian and Z. Nong, “The Study of Multimedia Data Model Technology Based on Cloud Computing,” *The 2nd International Conference on Signal Processing Systems (ICSPS)*, pp. V3-743-V3-746, July 2010.

[12] A. Michael, F. Armando, G. Rean, A. D. Joseph, K. Randy, K. Andy, L. Gunho, P. David, R. Ariel, S. Ion, and Z. Matei, “A View of Cloud Computing,” *Communications of the ACM*, Vol.53, No. 4, pp. 50-58, 2010.

[13] J. Rogers,O. Papaemmanouil, and U. Cetintemel, "A Generic Auto-Provisioning Framework for Cloud Databases," *IEEE 26th International Conference on Data Engineering Workshops (ICDEW)*, pp. 63-68, 2010.

[14] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, John Wiley & Sons, Inc., seventh edition. pp. 164-166, 2005.

[15] T. Stöhr, H. Märtens, E. Rahm, “Multi-Dimensional Database Allocation for Parallel Data Warehouses,” Proc. 26th VLDB Conference, Cairo, Egypt, Sep. 2000.

[16] N.E. Taylor and Z.G. Ives, “Reliable Storage and Querying for Collaborative Data Sharing Systems,” *IEEE 26th International Conference on Data Engineering (ICDE)*, pp. 40-51, 2010.

[17] C. Turbyfill. ”Comparative Benchmark of Relational Database System,”. PhD thesis, Cornell University, September 1987.

[18] M.A. Vouk, “Cloud Computing- Issues, Research and Implementations,” *the 30th International Conference on Information Technology Interfaces*, pp. 31-40, June 23-26, 2008.

[19] E. Walker, W. Brisken, and J. Romney, “To Lease or Not to Lease from Storage Clouds,” *Computer*, Vol. 43, Issue 4, IEEE Computer Society Press, pp. 6-9, April 2010.

[20] J.L. Wolf, D.M. Dias, P.S. Yu, and J. Turek, “Comparative performance of parallel join algorithms,” *In Proc. of Int’l Conf. on Parallel and Distributed Information Systems*, pages 78-88, Miami, Florida, December 1991.

[21] S. Zhang, S. Zhang, X. Chen, and X. Huo, “Cloud Computing Research and Development Trend,” the 2nd International Conference on Future Networks, pp. 93-97, Jan. 2010.

[22] G.K. Zipf, “Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology,” Addison-Welsey, Reading, MA. 1949.

[23] Y.F., Zheng, C., Shao, “An efficient round-robin algorithm for combined input-crosspoint-queued switches,” *In: Dini P, ed. Proc. of the IEEE ICAS/ICNS, Papeete: IEEE Computer Society,* pp. 23−28.2005.

[24] Wu, J.; Ping, L.; Ge, X.; Wang, Y.; Fu, J. Cloud storage as the infrastructure of cloud computing. In Proceedings of the International Conference on Intelligent Computing and Cognitive Informatics, Kuala Lumpur, Malaysia, 22–23 June, 2010; pp.380–383.

[25] Gartner Says Cloud Computing Will Be as Influential as E-business, June 2008, (http://www.gartner.com/it/page.jsp?id=707508)

[26] The Apache Software What Is Apache Hadoop, 2012, (http://hadoop.apache.org/)

[27] 羅有隆、賴民軒，「雲端資料庫主機負載平衡之研究」，2012資訊技術應用與管理研討會論文集，義守大學，高雄，CD-ROM。