服務節點選擇機制應用於階層式雲端運算環境

江茂綸
朝陽科技大學
資訊與通訊系
助理教授
mlchiang@cyut.edu.tw

張庭毅
彰化師範大學
數位學習研究所
副教授
tychang@cc.ncue.edu.tw

陳彥儒
朝陽科技大學
資訊與通訊系
研究生
s9830617@cyut.edu.tw

摘要

由於網際網路的蓬勃發展及電腦硬體的快速成長，使得現行的網路應用服務需求日漸增加，進而造成網路服務供應商的龐大負擔。因此，為了要應付大量的客戶需求，網路服務供應商必需提昇相關硬體的計算能力及網路頻寬。然而，傳統的分散式系統無法因應現今龐大的應用服務需求。因此 Google 在 2007 年提出以分散式系統為基礎的網格運算新概念，稱之為雲端運算。在過去的研究中，有提出一個雲端運算服務區塊的架構，但是並沒有對節點的選擇作適當的管理。本研究提出一個使用 MDR(Multi-Dimensional Ranking)為基礎的雲端服務節點選擇機制 CSNEM(Cloud Service Node Election Mechanism)，除了選出區塊中適合提供工作分配的叢集頭節點外，並藉由 MDR 的方法讓系統在選擇服務節點進行工作時可以降低總執行時間，進而提昇服務效率。

關鍵字：雲端運算、叢集頭、分散式系統、節點選擇機制

Abstract

Due to the rapid growth of Internet and computer hardware, the demand of Internet application is getting growing, resulting in a large overhead for Internet Service Provider (ISP). To cope with a lot of user's demand, Internet Service Provider needs to improve the computational ability of hardware and network bandwidth. As a result, the traditional distributed system can not satisfy these huge demands for Internet application. A new concept of grid computing which is based on the distributed system is proposed by Google at 2007, called as cloud computing. However, the election management did not be proposed appropriately in a cloud computing service block structure in the past. Therefore, the proposed CSNEM (Cloud Service Node Election Mechanism) is base on MDR (Multi-Dimensional Ranking) to select appropriate cluster head nodes and service nodes in service blocks. Furthermore, the total execution time and performance can be improved.

Keyword: Cloud computing, Cluster head, Distributed system, Node election mechanism

1. 簡介

隨著電腦硬體設備快速的演進，微處理器的速度越來越快，成本也越來越低，讓一般無法負擔起超級電腦的中小企業，可以利用多台電腦透過建置叢集運算的方式來提供更多的運算能力，而這將是未來的主要趨勢。然而，資訊科技與網際網路的持續發展所帶動使用者對於網路服務需求量的增加，網路服務供應商漸漸無法完全滿足這些多樣化且巨大的網路服務需求。而為了達到使用者多元及大量的需求，Google 於 2007 年提出了以分散式系統[2,9]為基礎架構的網格運算新概念，稱之為

雲端運算[3,4,7,8]。此概念以使用者為導向，透過供應商建置在世界各地的運算節點，來提供大量的網路服務、運算能力及平台的租賃。其雲端運算的主要模式大致上分為三種，分別是 SaaS (Software as a Service)、PaaS (Platform as a Service) 及 IaaS (Infrastructure as a Service)；其中 SaaS 主要是讓使用者不需要安裝任何應用服務軟體，即可以隨時使用任何可上網的裝置來使用網路服務；而 PaaS 是 SaaS 的延伸，方便使用者在主機沒有軟體和 OS 等平台時，可以利用供應商所提供的平台，進而大量節省維護與故障處理的時間。此外，IaaS 模式則是提供一般中小企業租賃俱龐大的運算能力的基礎設施，透過向 IaaS 供應商來租借這些設備，企業只需要負擔相對的價格即可，將可以降低整體成本。

此外，由於網際網路的多媒體化，如部落格(Blog)、YouTube 等越來越多的網站不僅提供文字瀏覽，更進一步提供照片以及影音資訊，使得越來越多的網站都需不斷的提升網站瀏覽品質及擴充實體設備，以提供使用者更多元化的應用與更流暢的使用環境。而這樣無窮的商機也吸引了各大廠商的加入，如 Google、Microsoft、Yahoo、Amazon…等都加入這塊領域。然而，雲端運算的出現也延伸出了很多不同的問題，如 2009 年下旬發生了幾起令人矚目的雲端運算災難事件，包括：Sidekick 服務中斷、Amazon EC2 [11]遭到阻斷服務攻擊，以及 Google 電子郵件服務中斷等。因此像是容錯處理(Fault Tolerant Processing)、故障診斷協議(Fault Diagnosis Agreement problem)、平行處理 (Parallel Processing)、負載平衡(Load Balancing)及服務品質(Quality of Service)等議題，都是需要進一步進行探討及解決。

本研究將針對服務品質進行探討，其最主要的原因為雲端運算大都是以服務為導向的架構；然而，網路使用者的大量增加，在服務品質上的提升將是迫切需要改善的問題。因此，本研究將提出一個可以提高雲端運算環境服務品質的叢集頭選擇機制，利用 MDR (Multi-Dimensional Ranking)[6]分別選出叢集頭節點 (Cluster head) 及支援節點 (Support node)[5]，讓使用者所提交的服務可以透過有效率的快速分配，以減少工作的執行時間來提昇雲端運算的服務效率。

本文第二節為文獻探討，首先說明目前跨足雲端運算的國際大廠所提供之相關服務之探討，並說明 MDR 函式以及過去相關之研究；而在第三節中，將說明本研究所提出之叢集頭選擇機制；而實例探討將展示於第四節；最後，第五節為結論與未來工作。

2. 文獻探討

目前跨足雲端運算架構的國際大廠[16]，包括：Google、IBM、Microsoft、Yahoo 及 Amazon 等企業，都對雲端運算這塊新的環境充滿了期許，也作了很多進入的準備。而在美國網路搜尋市場占有率第一名的 Google，其實很早就將雲端運算概念應用在自家所提供的服務上，諸如 Gmail，YouTube，Google Docs，Google Talk，Google Calendar 及 Google Gadget 等 [13]。此外，Google 於 2007 年 10 月與 IBM 合資超過 1,500 萬美元建立 Google 101 大型資料運算中心，並在 2008 年將雲端運算定位為未來的發展策略，而這點可從 Google 進軍通訊產業而推出的 G-phone 看出點端倪。因此從 Google 積極大舉佈局雲端應用的動作下，相信在加強架構「端」連到「雲」的完整的商業模式後，是很有機會在未來市場繼續保持領先地位。

除了 Google 的積極發展外，IBM 也主推 Blue Cloud「藍雲計畫」來進軍雲端運算，其計畫的主要切入點不在於如何提供使用端各種服務，IBM 更專注於如何提供雲端運算所需

的硬體設備與管理軟體，允許企業將運算任務分成不同組件，並透過有效率的電腦系統執行，解決企業尖鋒及離鋒時間的系統負荷量問題。同時結合 Google 在全球數個城市建立雲端運算中心，除了提供容量大的免費信箱之外，還推出網路平台之應用功能等，如 Google 文件，不但可以提供線上編輯文件，同時也能將所編輯的文件存放於帳戶內的空間。

然而，Microsoft 提出不同於上列所提的運作模式，其主要在雲端的策略是「Software + Service」。其所推出的新作業系統「Azure」[12]，將結合 Live Mesh 開發新功能，並整合各種 Live Services, 像是 Windows Live Messenger、Live Search 等。此外，Azure 的另一項用途則是讓軟體開發者所撰寫的程式可以直接在微軟資料中心上線，不需透過公司裡的伺服器。也就是說，Azure 就像是微軟線上服務的地基，扎穩微軟邁向雲端之路。

針對雲端運算服務，Yahoo 也提出了雲端運算架構 Hadoop[14]，主要是應用在自家搜尋服務的兩千台伺服器上，藉以處理超過 5 Petabytes 的網頁內容，進而建立整個網際網路的網頁索引資料。此外，Yahoo 的雲端產品定位為 Consumer Cloud Computing，主要提供 Yahoo! Live，Yahoo! One Connect 及 News Globe 等線上訊息服務。而即將正式開放的 Yahoo Application Platform，則是提供開發者線上撰寫和執行程式的開放平台。

而最後，Amazon 也透過虛擬化的技術將 Amazon EC2 搭配 Amazon S3[15]儲存服務，提供各種不同規格的虛擬主機和儲存空間的 Web Services；使軟體開發者能快速地在上面安裝或執行所需的服務，且只需負擔所使用的時間與資源即可，因此將可以大量降低企業的營業成本。

透過上述的探討，雲端運算將是未來服務供應商最主要的趨勢，由於大量的網路使用者出現，同時也產生了服務回應過慢及服務效率不張等問題。因此，如何選擇合適的服務節點來提昇工作效率，也是我們需要去考量及解決的新興問題。然而，過去選擇服務節點的方法有很多種，像是 First fit、Best fit、LRU (Least Recently Used)、LFU (Least Frequently Used) [10]等方法；First fit 是在所有節點中選擇第一個符合此工作的節點來執行；Best fit 是在所有可執行的節點中，選擇能力最好的節點來執行服務；而 LRU 則是選擇最久未被使用的節點來執行服務，然而，此法不一定會選擇到較佳的服務節點；在 LFU 演算法中，主要是根據節點的使用頻率，來選出使用頻率最小的節點，同樣地，此法也不一定可以選擇到較適合的服務節點。因此為了要在雲端運算的眾多節點中選擇到適當的服務節點將是目前的一大挑戰；而本研究將使用 MDR 選擇機制來通盤考量各種節點因子，像是負載量、本地需求量、CPU 等級等因子來進行篩選。然而，在不同的網路環境當中，所要考慮的環境因子就有所不同，因此本研究針對雲端運算環境所需考量的因子來進行篩選，其因子如 CPU 速度(GMHz)、儲存能力(GB)、目前負載量、傳輸能力(KB/s)、Memory(MB)等，這些因子是符合目前在網路服務伺服器的考慮因子。

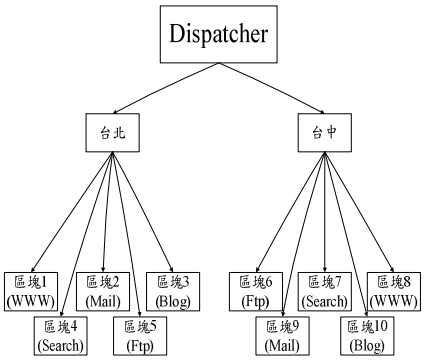
3. 研究方法

3.1 研究架構

由於雲端運算環境是架構在網際網路上以使用者的服務需求為導向的分散式系統應用環境。而為了滿足使用者多元化的需求，在服務配置上是採用事先配置的方式，將服務功能預先分配給服務節點，再將相同的服務節點合併成一個服務區塊。隨後，不同服務類型的區塊將以地區作分群類別，進而提昇服務效能；因此，本篇針對雲端環境之特色並參考雲端運算服務區塊架構圖[1]，來針對服務區塊

的節點選擇進行深入的探討，其架構如圖一所示。

然而，在圖中每個服務區塊在執行服務時，並沒有考慮到服務節點的能力值。此外，由於雲端運算環境包含大量的運算節點，若沒有一個有效率的選擇機制來選擇服務節點，將會造成整體服務效率的降低。因此，本研究基於MDR的選擇機制提出雲端服務節點選擇機制 Cloud Service Node Election Mechanism (CSNEM)，來選出叢集頭節點(Cluster head)及支援節點(Support node)來進行工作分配，不但讓服務區塊的工作能進行快速的分配，也能降低工作完成的回應時間。

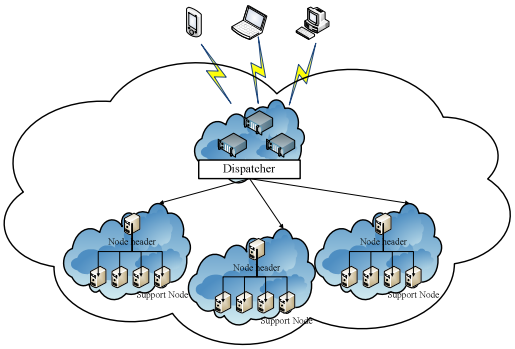


圖一、雲端運算服務區塊架構圖

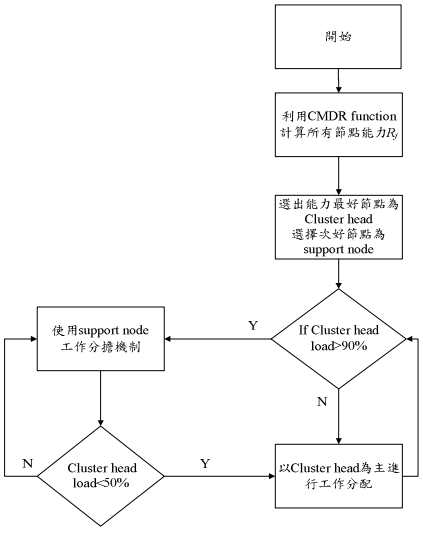
隨後本研究針對雲端運算所提出之階層式架構如圖二所示。其主要架構是為了因應雲端運算的環境之下，每個服務區塊擁有大量的運算節點，而對於如何管理選擇適合節點來執行服務是相當的耗費時間的。所以本研究在利用此架構來因應大量的工作分配。此外，過去有許多學者[5]針對不同的叢集頭選派方式加以探討，但都不是針對雲端運算環境這樣的特性來設計，所以對於在工作分配的效率上沒有效率。因此，本研究在所提出的CSNEM方法中，利用針對雲端運算環境所設計的 CMDR (Cloud Multi-Dimensional Ranking)的節點能力計算方式，進行叢集頭節點及支援節點的選擇，以提昇整體服務效率。隨後，下一節章將介紹所提之叢集頭選擇機制。

3.2 叢集頭選擇機制

本研究所提的CSNEM中，主要是透過選擇每個不同服務區塊的叢集頭節點及備援支援節點來進行工作分配，而為了管理服務區塊中的大量節點及因應大量的工作分配，本研究將透過 CMDR 根據 CPU 速度(GMHz)、儲存能力(GB)、目前負載量、傳輸能力(KB/s)、Memory(MB)等因子來選出在服務區塊中能力最佳的節點作為叢集頭節點。此外，為了避免叢集頭節點在分配工作的時候有負載量過高的情況，進而造成工作無法進行分配而導致服務區塊崩潰的情形，在選擇叢集頭節點時，同時選出能力第二高的節點為支援節點，以便於在叢集頭節點的負載量達到門檻值時，可以啟動支援機制，幫助分配工作。其主要執行流程如圖三所示。



圖二、雲端服務區塊階層式架構圖



圖三、CSNEM 叢集頭選擇機制流程圖

首先，CSNEM 可分為四大步驟，其步驟描述如下所示：

1. 利用 CMDR function 正規化計算所有節點能力 R_j
2. 選出能力最好的節點為叢集頭節點(Cluster head)，次好的為支援節點(Support node)，並根據其能力值 R_j 將工作分配至能力較佳的服務節點。
3. 若該叢集頭節點的負載量大於所設定的門檻值百分比，使用支援節點的工作分擔機制，避免叢集頭節點的負載量超過門檻值，進而影響系統運作。
4. 在工作分擔機制啟動後，叢集頭節點的負載將得到舒緩；而當叢集頭節點的負載量小於所設定的門檻值後，將回復以叢集頭集點為主的工作分配機制。

在 Step (1)中，每個節點需先進行正規化以進行計算。其節點 *i* 的 *cpu* 值 C_i 的計算方法如公式(1)所示，其中 $cpu_{\max}$ 為該服務區塊中的 *cpu* 量最大值。隨後，在根據不同的服務類型去作各個服務區塊選擇叢集頭節點之權重調整，同時利用 CMDR 計算每個節點之能力 R_j，以作為叢集頭節點之依據。在此函式中，除了考量節點的 *cpu* 值之外，將節點的儲存能力(GB)、目前負載量、傳輸能力(KB/s)、Memory(MB)等相關因子作為此函式之決策因子，而各決策因子之值還可利用公式或數值對應表推算出其相對應之值。以便選出最適合之叢集頭節點。因此，提出之 CMDR 如公式(2)所示，其參數說明如表一所示。

$$C_i = \frac{cpu_i}{cpu_{\max}}, 0 < C_i \le 1 \quad \text{……..公式(1),}$$

$$R_j = \sum_i^n a_i X_{ij} = a_1 \times X_{1j} + a_2 \times X_{2j} + ... + a_n \times X_{nj}, 0 \le a_i X_{ij} \le 1,$$

$$\sum_i^m a_i = a_1 + a_2 + ... + a_m = 1$$

……公式(2)

表一、CMDR 參數說明

R_j	為 node *j* 所計算出的能力
X_{ij}	為 node *j* 的第 *i* 個決策因子
i	為函數中第 *i* 個決策因子,共有 *n* 個決策因子
a_i	為各個決策因子的權重值

隨後，本研究在 step (2)將依上述的公式(2)來計算每一個節點的能力值，同時進行能力值之比較，將能力值最好的節點設定為叢集頭節點，次好的節點為支援節點。

在 step(3)中，會判斷叢集頭節點的負載是否大於所設定的門檻值，若是大於門檻值的話會啟動支援節點進行工作分擔，目的是為了不要讓叢集頭節點因負載過高造成故障或損毀的情況，導致系統無法運行。

最後在 step(4)中，系統判斷叢集頭節點的負載量達到舒緩過後，即回覆由叢集頭節點為主的工作分配。

透過選出叢集頭節點來幫助服務的工作分配，但由於在雲端環境之下，擁有大量的服務節點，因此叢集頭節點可能成為系統的瓶頸，本研究選出支援節點來幫助舒緩過多的負載量，可讓工作的分配更加有效率。

4. 實例說明

本節將以兩個實例來說明本研究所提出的雲端服務節點選擇機制 CSNEM 與其他的方法的比較，而由於 CSNEM 根據每個服務節點的能力去分配工作，不但可以降低執行時間，且能夠使得整個服務效率得到提昇。由於在雲端的環境中有很多服務節點，因此本研究假設工作的數量會小於等於服務節點的數

量，所以在本例中會比較(a)工作數量(T)小於服務節點(N)以及(b)任務數量等於服務節點這兩個例子。

首先在每個服務區塊中利用 CMDR 來選擇叢集頭節點及支援節點，而當使用者的需求進來之後，再依據每個節點的能力值(CPU 速度(GMHz)、儲存能力(GB)、目前負載量、傳輸能力(KB/s)、Memory(MB)等)將工作分配給對應的節點。

以圖三為例，我們假設目前有十個工作需要被執行。分別是 T1、T2、T3、T4、T5、T6、T7、T8、T9、T10，其預測時間如表二所示，隨後並依據(a)任務數量小於服務節點(T<N)及(b)任務數量等於服務節點來分別來計算不同選擇方法之下的工作執行時間。其範例及假設條件如下：

1. 每個節點皆可以服務每個工作。
2. 每個工作的執行時間可被預測。
3. 服務節點個數大於工作總數，讓每個工作都可分配到一個節點中執行。

範例(a)：任務數量小於服務節點(T<N)

Step (1)：在此例中由於任務數量小於服務節點，以 T1-T5 為例來執行。調度節點(Dispatcher)目前收集到需要被執行的工作分別是 T1、T2、T3、T4 及 T5，並將這些工作交由叢集頭需求的執行時間，在本例中把工作分配給服務節點來進行分配。隨後，叢集頭節點會依照工作進入的次序來決定服務的先後。在此時服務區塊會將所有節點相關資訊(CPU 速度(GMHz)、儲存能力(GB)、目前負載量、傳輸能力(KB/s)、Memory(MB))收集並提供給叢集頭節點進行工作分配前的計算。其各服務節點資訊如表三所示。

Step (2)：由於各個節點因子的計量單位不盡相同，為求方便計算，將其節點相關資訊根據公式(1)作正規化處理，如表四所示。

Step (3)：隨後，根據表四中的服務節點資訊代入 CMDR 函式後，即可將服務節點的綜合能力值 R_j 依序排列出來。而為了縮短使用者需求的執行時間，在本例中把工作分配給服務能力最好的五個節點去執行。其五個工作預測被執行的節點如表四所示。隨後，根據不同叢集頭選擇機制所選出的服務節點列表如表五所示。

Step (4)：由於工作的執行時間是根據 CPU 能力來決定，不同的 CPU 能力會影響其執行時間，其計算方法如公式(3)。

$$\text{執行時間} = \frac{\text{預測執行時間}}{\text{CPU能力}} \quad \text{公式(3)}$$

透過公式(3)計算出每個工作在不同的 CPU 執行的工作時間，例如工作 T1 在 N4 以及 N7 的執行時間分別為 0.88s 及 1.73s。根據 CPU 能力所變更的工作執行時間如表六所示：

表二、工作預測執行時間

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
預測執行時間（s）	3	5	10	8	2	7	11	4	5	4

表三、服務區塊中各服務節點之資訊

服務節點	CPU 速度(GMHz)	儲存能力(GB)	目前負載量	傳輸能力(KB/s)	Memory (MB)
N1	3.13	2750	0.82	2983	233.4
N2	2.28	1800	0.87	8221	160.22
N3	1.24	1000	0.69	3969	34.07
N4	3.42	500	0.02	3585	104.02
N5	1.74	350	0.34	8712	207.82
N6	3.51	100	0.84	724	121.06
N7	1.73	250	0.36	11051	319.57
N8	3.35	5000	0.34	7692	142.81
N9	3.29	1350	0.24	2336	540.08
N10	2.78	2500	0.42	4796	406.06

表四、節點資訊正規化

服務節點	CPU 速度(GMHz)	儲存能力(GB)	目前負載量	傳輸能力(KB/s)	Memory(MB)
N1	0.891737892	0.55	0.942528736	0.269930323	0.432158199
N2	0.64957265	0.36	1	0.743914578	0.296659754
N3	0.353276353	0.2	0.793103448	0.359153018	0.063083247
N4	0.974358974	0.1	0.022988506	0.324405031	0.192601096
N5	0.495726496	0.07	0.390804598	0.788344946	0.384794845
N6	1	0.02	0.965517241	0.065514433	0.224151977
N7	0.492877493	0.05	0.413793103	1	0.591708636
N8	0.954415954	1	0.390804598	0.696045607	0.264423789
N9	0.937321937	0.27	0.275862069	0.211383585	1
N10	0.792022792	0.5	0.482758621	0.433987874	0.751851578

表五、各方法選出的執行節點(T<N)

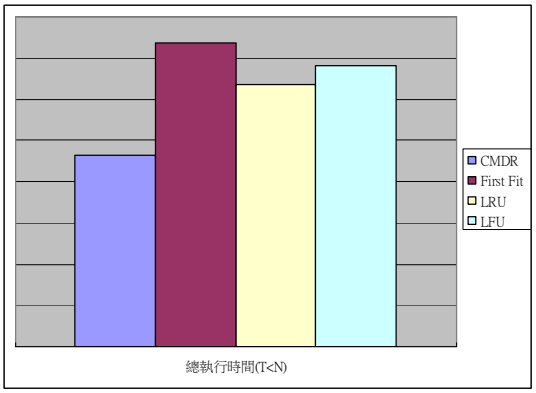
Job/選出節點	CMDR	First Fit	LRU	LFU
T1	N8	N1	N4	N7
T2	N1	N2	N6	N10
T3	N9	N3	N7	N9
T4	N10	N4	N1	N3
T5	N2	N5	N9	N8

表六、工作的執行時間(T<N)

Job/執行時間	CMDR	First Fit	LRU	LFU
T1	0.895522	0.958466	0.877193	1.734104

T2	1.597444	2.192982	2.873563	1.798561
T3	3.039514	8.064516	5.780347	3.039514
T4	2.877698	2.339181	2.555911	6.451613
T5	0.877193	1.149425	0.607903	0.597015

因此，在範例(a)中，由於 First Fit 的選擇機制只選擇第一個能夠服務的節點，並不一定會選擇到較佳的節點進行服務。LRU 與 LFU 的選擇機制分別是選擇之前最久未使用過的以及使用頻率最小的節點，這樣的選擇方式也不一定會選出較佳的服務節點。然後，透過圖四可以清楚的知道，CSNEM 可以選擇出較佳的節點進行服務，因此獲得較低的總執行時間。



圖四、總執行時間 (T<N)

範例(b)：任務數量等於服務節點(T=N)

Step (1)：以 T1-T10 為例來執行。分配者將這些工作交由叢集頭節點來進行分配，依照工作進入的次序來決定服務的先後。

Step (2)：同範例(a)。

Step (3)：使用 CSNEM 及其他節點選擇方法所選擇出來的節點如表七所示。

Step (4)：同例子(a)，根據 CPU 能力所變更的工作執行時間如表八所示：

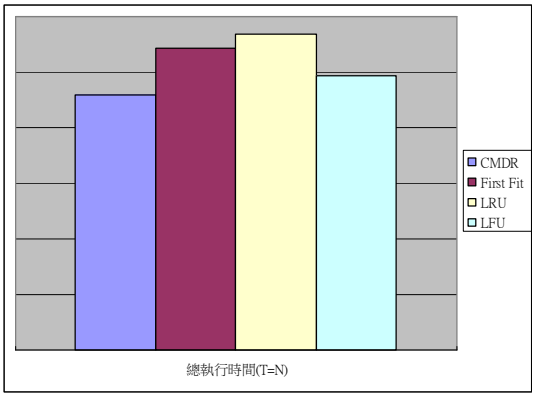
在圖五的範例(b)中，我們所提出的方法與其他方法相較之下，總執行時間較是有較好的效果。主要是本研究的方法有考慮服務節點的能力值，可以選出較佳的節點來進行服務。

表七、各方法選出的執行節點(T=N)

Job/選出節點	CMDR	First Fit	LRU	LFU
T1	N8	N1	N4	N7
T2	N1	N2	N6	N10
T3	N9	N3	N7	N9
T4	N10	N4	N1	N3
T5	N2	N5	N9	N8
T6	N6	N6	N3	N6
T7	N4	N7	N2	N1
T8	N7	N8	N5	N5
T9	N5	N9	N10	N4
T10	N3	N10	N8	N2

表八、工作的執行時間(T=N)

Job/執行時間	CMDR	First Fit	LRU	LFU
T1	0.895522	0.958466	0.877193	1.734104
T2	1.597444	2.192982	2.873563	1.798561
T3	3.039514	8.064516	5.780347	3.039514
T4	2.877698	2.339181	2.555911	6.451613
T5	0.877193	1.149425	0.607903	0.597015
T6	1.994302	1.994302	5.645161	1.994302
T7	3.216374	6.358382	4.824561	3.514377
T8	2.312139	1.19403	2.298851	2.298851
T9	2.873563	1.519757	1.798561	1.461988
T10	3.225806	1.438849	1.19403	1.754386



圖五、總執行時間 (T＝N)

本研究使用 CMDR 演算法選取出能力值較佳的節點來執行，同時也在選取節點時考慮了節點相關資訊(CPU 速度，儲存能力、負載量、傳輸能力、Memory 等)，因此，叢集頭節點在分配工作時可以讓工作分配到能力值較佳的節點，進而讓工作的執行時間能夠縮短如圖四、圖五所示。然而在本例中，若叢集頭節點的負載量過高時，為了避免造成服務區塊的崩潰，將會使用支援節點來取代叢集頭節點，同時透過上列的步驟再次將工作分配給能力較佳的服務節點，以維持系統的可靠度。因此，透過本研究的節點選擇機制將可以有效的提升整體的服務效率。

5. 結論與未來工作

由於網際網路的蓬勃發展及使用者對於網路服務需求的增加，因此，以分散式系統為基礎架構的雲端運算概念將被廣泛的應用。然而，雲端運算是由大量的服務節點所組成，如何利用這些節點來達到較佳的服務效率是很重要的議題。

因此，本研究提出一個叢集式節點的服務區塊架構以及雲端服務節點選擇機制 CSNEM，來選出適合分配工作的叢集頭節點。此外，透過使用 CMDR 讓工作分配到最佳的服務節點進行服務，除了可以降低工作總執行時間，更提升了整個雲端環境的服務效率。

然而，在雲端運算的服務架構中，可能出現服務需求量突然暴增的情況，這部分需要去考慮到工作排程的問題，因此，未來將搭配排程演算法來提昇雲端運算服務的效率。

誌謝

這篇論文是國科會計畫(NSC99-2221-324-041-MY3，NSC99-2632-E-324-001-MY3，NSC99-2221-E-018-018)研究成果的一部份，我們在此感謝國科會經費支持這個計畫的研究。

參考文獻

[1] 江茂綸、陳彥儒，“以服務區塊提昇雲端運算架構之效能”，The 15th Mobile Computing Workshop，2010。

[2] Weiss, A. “Computing in The Clouds,” *netWorker*, Vol. 11, No. 4, pp.16-25, 2007.

[3] Vouk, M. A. “Cloud Computing- Issues, Research and Implementations,” Information Technology Interfaces, pp. 31-40, 2008.

[4] Kopetz, H. and Ochsenreiter, W., “Clock Synchronization in Distributed Real-Time Systems,” *IEEE Transactions on Computers*, Vol. C-36, No. 8, pp. 933-940, 1987.

[5] Shafiq U. H., Hussein T. M., Nicolas D. G., “Achieving Reliability over Cluster-Based Wireless Sensor Networks using Backup Cluster Heads,” IEEE Global Telecommunications Conference, pp.1149-1153, 2007.

[6] Dias de Assuncao, M., Alexandre di Costanzo, Buyya, R., “Evaluating the Cost-Benefit of Using Cloud Computing to Extend the Capacity of Clusters,” The 18th ACM international symposium on High performance distributed computing, pp. 141-150, 2009.

[7] Maggiani, R, “Cloud computing is changing how we communicate”, Professional Communication Conference, pp.1-4, 2009.

[8] Naseera, S. “Agent Based Replica Placement in a Data Grid” ,Computational Intelligence ,Communication Systems and Networks, pp.426-430, 2009.

[9] X.b. Wang, Y.M. Teo, and J.N. Cao, “Message and Time Efficient Consensus Protocols for Synchronous Distributed Systems,” *Journal of Parallel and Distributed Computing*, Vol. 68, No. 5, pp. 641-654, 2008.

[10] Per Brinch Hansen, Operating system principles. California Institute of Technology.

[11] http://aws.amazon.com/ec2/

[12] http://www.microsoft.com/windowsazure/

[13] http://www.google.com.tw/

[14] http://hadoop.apache.org/

[15] http://aws.amazon.com/s3/

[16] http://eblog.cisanet.org.tw/post/Cloud-Computing.aspx