# 移動式網格之分散式資料分群技術

趙景明 chao@cis.scu.edu.tw
陳俊華 bop85035@gmail.com
東吳大學資訊科學系

## 摘　要

在目前的網路使用環境快速發展下，資料量的增加非常迅速，導致傳統資料分群技術執行的效能大幅降低。因此分散式的資料分群技術逐漸興起，不過由於頻寬的限制、分群結果合併誤差等因素的影響，使得執行效能以及分群品質成爲分散式資料分群演算法設計上的重要議題。本研究即是以移動式網格的概念爲基礎，提出一個分散式資料分群的技術，利用網格切割來源資料集後，再以移動式網格決定最終資料群集的方式，企圖在分群品質與執行效能上取得一個較佳的平衡。

關鍵詞：資料探勘、資料分群、分散式環境

# A Distributed Data Clustering Method with Movable Grid

Ching-Ming Chao
Chun-Hua Chen

Department of Computer and Information Science, Soochow University

**Abstract**

**Because of the high demanding and using in present internet applications, the huge amount of data processed has been increasing rapidly and resulting in extreme low efficiency in the traditional data clustering processing. Therefore, the technique of distributed data clustering emerging becomes more and more acceptable. Due to a few concerns such as bandwidth constrains, privacy and security, clustering processing performance and clustering result quality become a very important issue regarding the design of distributed data clustering algorithm. Thus, we propose a novel distributed data clustering algorithm based on the concept of movable grid. The theory behind is to use grids to separate the data source first, and then move them to determine the final clusters. Our experimental evolution shows that movable grid method is a good solution for data clustering in a distributed environment that achieves balance between processing performance and clustering result quality.**

**Keywords: data mining, distributed data clustering, distributed environment**

## 壹、 緒論

近年來由於全球化的趨勢，各企業體爲了維持其在市場中的競爭能力，例如：下降生產成本、開發新產品…等等，不得不將觸角延伸至世界各處。而且，爲了資料處理分析的方便性以及即時性，使得各企業體在各區域競爭市場中紛紛建立起獨立運作的區域資料庫中心。拜網路技術與分散式資料庫技術的日趨成熟與發達所賜，各企業體得以將散居於世界各處的資料庫連接起來整合應用，資料探勘（Data Mining）便是其中的應用之一。但是目前資料探勘所使用的技術或演算法大都是以單一資料庫爲基礎下所發展出來的。若是將其直接套用到現今企業的資料庫架構上，由於資料量的過於龐大、網路頻寬的限制，資料傳輸的品質以及資訊安全…等因素影響，似乎不切實際。因此，如何在這麼一個分散式的資料庫環境下，設計出一個有品質、高效率的資料探勘技術、方法或架構，便是現今資料探勘研究上的一個重要議題。

由於資料探勘技術的種類非常廣泛，有鑑於分群技術（Clustering）在企業實際的應用中十分普遍，因此，本研究將針對在分散式環境下的資料分群技術與方法作爲深入研究探討的對象。包括如何在網路中穩定的傳輸資料，如何以較小的資料結構來描述資料分群後的特徵以及尋找一個最爲理想的資料分群技術與方法。希望藉由本研究的成果，可以作爲相關議題以及企業在建置分散式資料探勘上的一個參考與應用。

## 貳、 文獻探討

### 一、 k-means 演算法

雖然資料探勘的技術很早以前就已經被探討，而且研究分群技術的文獻也有很多。例如在 Han 與 Kamber [7]，Berry 與 Linoff [1]，Jang、Sun 與 Mizutani [10]等人的著作以及 Jain、Murty 與 Flynn [9]，Halkidi、Atistakis 與 Vazirgiannis [6]，Gonzalez [5]等人的論文上，都有很詳細的介紹。而其中 k-means 是最普遍爲人所知道的方法，也是最多人拿來比較的方法之一。而且我們所要提出的分散式資料分群演算法也是以此 k-means 演算法爲基礎改良而來。因此我們有必要對此一演算法做更進一步的研究與探討。

k-means 演算法是屬於分割式（partition）分群的方法。1967 年時由學者 MacQueen 所提出。它是採用歐幾里得（Euclidean）距離來作為對資料分群的基礎。首先，讓我們回顧一下整個 k-means 演算法，並將 k-means 演算法的步驟簡述如下：(1)由使用者定義出所要的群集數 k，並且在資料集空間中任意選取 k 個點做爲起始的資料群集中心。(2)計算資料集中所有的資料點與 k 個資料群集中心點的距離，並且以離中心點距離最短的原則做爲分群條件，將資料集中所有的資料做分

群。(3)以目前的分群結困爲基礎，重新計算各個群集的幾何中心做爲下一次計算時的群集中心。(4)重複執行步驟 2 與步驟 3，直到每一固群集的群中心皆不會再更動爲止。

由檢視上述的 k-means 演算法執行的步驟後，我們可以歸納出幾個特徵：(1)k-means 演算法的資料群集數目是由使用者所定義，因此，若是在原始資料集中，群集的數目大於使用者所設定的數目，則最後分群的結果將會失真，形成一個較爲粗略的分群結果，使得某一些的資料群集會因爲使用者所設定的分群數目不足的緣故而被隱藏消失。(2)k-means 演算法會尋訪資料集中所有的資料，並且將每一個資料皆視爲是某一個資料群集的資料成員。因此，k-means 演算法並沒有辦法過濾所謂的雜訊(noise)資料，同時這些雜訊資料還會影響到所計算出來的資料群集中心正確性，造成計算結果的失真。(3)由 k-means 演算法的步驟得知，假設資料集的資料總筆數爲 n，欲分群的群集數目爲 k，計算群集中心所需的迴圈次數爲 t，那麼整個 k-means 演算法所要花費的時間複雜度爲 O(nkt)，而實際上 n >> k 且 n >> t，因此，k-means 演算法的時間複雜度大約可以近似爲 O(n)，也就是說當資料集中資料量愈大的時候，k-means 演算法計算出結果所需要花的時間就會愈長。而在進行資料探勘時，其所需要的資料量一般來說都是非常龐大的。因此，若是採用 k-means 演算法進行資料群作業的話，將會耗費相當長的一段時間方可完成。

簡單來說 k-means 演算法是將資料由使用者指定分割成 k 個資料集。每一個分割（Partition）代表一個群組（cluster），群組是以最佳化分割標準（partitioning citerion）為目標，其分割標準的目標函數又稱為相似函數（similarity function）。因此，同一群組的資料物件具有最為相似的資料屬性。另外，k-means 演算法對於處理分群資料中有明確集中的情形，有相當不錯的成效。但是若是分群資料中具有雜訊或者是離群值時，所代表資料群組特性的中心值將會受雜訊的影響而有所失真。

## 二、分散式環境下的分群方法

隨著網路技術的成熟與快速發展，企業內所累積的資料含量成指數成長，昔日的分群技術在處理作業上已顯得沒有效率。爲了處理大量資料的問題，Jain、Murty 與 Flynn [9]，Gonzalez [5]等人在他們論文中便提出了一些作法。Bradley、Fayyad 與 Reina [2]針對大的資料集作分群，使用一個 RAM Buffer 的機制來對常用的資料作保留，以減少掃瞄資料的次數，但是這個作法卻需要付出額外的計算以及保留資料集的空間。

在系統架構方面，隨著分散式環境的日益普及，Kargupta、Hamzaoglu 與 Stafford [11]提出了一個在分散式環境中平行處理的探勘軟體代理人（Agent），之後又提出

了一個分散式資料探勘的架構。Du 與 Agrawal [4]提出在 Grid 環境下發展一個分散式資料探勘的方法，利用每一台機器送出離中心點最近的 n 筆資料，然後再把全部的資料結合起來即可找到全部資料中最接近中心點的 k 筆資料。而 Lin 與 Chen [13]則是先在每一台機器中利用 partition 演算法進行資料分群後，把結果傳送給其他機器，接著再透過 hierarchy 的方法將結果進行群組的關聯。其他如 Dhillon 與 Modha [3]就從事有關多處理器平行處理的研究。Oguchi 與 Kitsuregawa [14]則提出透過動態資料複製的方法來提升系統 I/O 的效率。另外還有一項很重要的研究，就是分群的驗證方法，Halkidi [6]等學者在他們所發表的文章，有詳盡的介紹各種驗證及測量的方法。

由以上的探討中可以發現，雖然在分散式資料分群的議題上有一些的討論，但是在方法的概念上不外乎兩種：（1）將整個分散式環境中的所有機器利用某種機制連結起來形成一個虛擬的單一資料庫，然後再利用傳統的資料分群演算法對此一虛擬的單一資料庫進行資料探勘得到結果。這個作法的優點是可以讓分散式的環境中所進行的分群作業得到一個與在實體單一資料庫中所進行的分群作業一致的結果，以及雖然環境已由單一環境轉變成分散式的環境，但是所使用的資料分群演算法並不需要作太大的修改即可以使用。缺點就是必須在網路中傳遞大量的資料，而且必須處理大量的資料才能得到資料分群的結果。因此網路頻寬以及處理器效能將成爲這個方法的瓶頸。（2）在每一部機器中先進行資料的分群，然後將分群的結果透過特徵描述的方式傳送給其他參與者。再透過群集（aggregation）的方式把相類似的群連結起來，最後形成一個階層式的群組關聯圖。此一方法的優點在於克服了第一種方法的缺點，減少了網路傳輸的資料量以及處理器所需處理的資料量，縮短了資料分群的時間。但是缺點就是因爲使用描述特徵的方式來作爲分群的結果，所以這是一個概括的結果。因此當群集完成後將會是一個概略式的分群結果比之單一資料庫分群的結果較爲不準確。也就是說描述特徵的表示式將會成爲這一個方法的瓶頸。

在此，我們把第一類方法稱爲虛擬環境資料分群法，第二類方法稱爲特徵模型資料分群法。以下將針對此兩大類分別列舉一代表性的方法加以說明。

### (一) 虛擬環境資料分群法

因爲資料庫技術的進步以及網路傳輸的發達，使得企業組織的資料在快速成長下變得不僅龐大而且分散。因此，若是可以將所有的資料來源利用一種機制連結起來，在邏輯上可以視爲一個單一資料來源的話，那麼原本傳統的資料分群方法便可以在小幅度的修改調整後，套用到這一個邏輯的單一資料來源，並且得到一個正確的結果。這就是虛擬環境資料分群法的基本概念。根據這樣的基本概念，Dhillon,

Modha [3]在 1999 年提出了一個實際的作法。這個作法是利用平行運算電腦(parallel computing)系統以及訊息傳遞模組(message passing model)將原本分散的資料來源連結起來形成一個單一的虛擬資料來源，之後，再透過 k-means 演算法來對此一資料來源進行資料分群運算，得到結果。其演算法如下所述：

```
1:  P = MPI_Comm_size ();
2:  μ = MPI_Comm_rank ();
3:  MSE = LargeNumber;
4:  if (μ = 0)
5:      Select k initial cluster centroids {m_i}_{i=1}^k;
6:  endif;
7:  MPI_Bcast ({m_i}_{i=1}^k, 0);
8:  do {
9:      OldMSE = MSE;
10:     MSE' = 0;
11:     for j = 1 to k
12:         m_j' = 0; n_j' = 0;
13:     endfor;
14:     for i = μ * (n/P) + 1 to (μ + 1) * (n/P)
15:         for j = 1 to k
16:             compute squared Euclidean
                    distance d^2(X_i, m_j);
17:         endfor;
18:         find the closest centroid m_L to X_i;
19:         m_L' = m_L' + X_i; n_L' = n_L' + 1;
20:         MSE' = MSE' + d^2(X_i, m_L);
21:     endfor;
22:     for j = 1 to k
23:         MPI_Allreduce (n_j', n_j, MPI_SUM);
24:         MPI_Allreduce (m_j', m_j, MPI_SUM);
25:         n_j = max(n_j, 1); m_j = m_j/n_j;
26:     endfor;
27:     MPI_Allreduce (MSE', MSE, MPI_SUM);
28: } while (MSE < OldMSE)
```

圖 1 Parallel k-means 演算法

| | |
|---|---|
| MPI_Comm_size() | returns the number of processes |
| MPI_Comm_rank() | returns the process identifier for the calling process |
| MPI_Bcast(message, root) | broadcasts "message" from a process with identifier "root" to all of the processes |
| MPI_Allreduce(A, B, MPI_SUM) | sums all the local copies of "A" in all the processes (reduction operation) and places the result in "B" on all of the processes (broadcast operation) |
| MPI_Wtime() | returns the number of seconds since some fixed, arbitrary point of time in the past |

圖 2 訊息傳遞函數介面

圖 1 是整個分散式演算法的描述，而圖 2 是訊息傳遞模型所使用的介面函數(Message Passing Interface: MPI)。在圖 1 中，我們可以將整個演算法主要分成四個大區段。第一個區段：稱爲初始化區段(Initialization)，從行 1 到行 7。在這一個區段，設定了所需要的分群數目、群集中心點，以及透過介面函數得知所有處理器(processor)的數量以及每一個處理器的識別編號(identifier)。第二個區段：稱爲距離計算區段(distance calculation)，從行 14 到行 21。在這一個區段，每一個處理器將分配到一部分的資料集，之後，每一個處理器將同時並行計算所分配到的資料集中每一筆資料記錄到群集中心的歐幾里得距離(Euclidean distance)，並且依據最短距離原則，把資料記錄指定給距離最近的資料群集中心。第三個區段：稱爲群集中心重新設定區段(centroid recalculation)，從行 22 到行 26。在這一個區段，每一個處理器利用訊息傳遞模組，透過訊息交換的方式得知整個資料集的分群結果，並藉由此一整體的分群結果，重新計算所有的資料群集中心。第四個區段：稱爲停止條件區段(convergence condition)，行 28。判斷整個資料分群計算是否已經達到停止條件，若是已經符合停止條件，則停止整個資料分群的計算，否則將重複執行行 8 到行 27，直到符合停止條件爲止。

簡單的說，整個演算法就是利用平行計算的策略，將全部的資料切割分配給所

有的處理器，然後再透過共享記憶體(shared memory)以及訊息的傳遞的機制完成整個資料分群的計算。

### (二) 特徵模型資料分群法

雖然使用虛擬環境資料分群方法可以減少傳統資料分群演算法修改的幅度，使得傳統的資料分群演算法幾乎皆可以於分散式的環境下再使用。但是由於過程中需要大量的訊息傳遞、交換，因此，網路的頻寬便成爲整個虛擬環境資料分群方法的瓶頸。所以，如果可以在每一個資料源處先把各自的資料分群計算出來，再將這些分群結果集中起來，再進行分析、合併。之後，如果能夠得到一個可以接受的整體性資料分群結果的話，由於各自計算出來的資料分群結果其資料量必定遠小於虛擬環境資料分群法計算過程中所需傳遞的資料量，因此將可以克服因爲網路頻寬因素對整個資料分群計算所造成的限制。這個就是特徵模型資料分群法被產生出來的概念。

由 Kriegel, Kroger, Pryakhin, Schubert [12] 在 2005 年所提出的一個 DMBC(Distributed Model-Based Clustering)方法便是依據特徵模型資料分群法的概念所產生出來的一種方法。DMBC 是由 EM(Expectation Maximization)分群演算法所衍生而來，並且利用高斯分配所使用的統計參數來描述群集的特性($\mu_c, \Sigma_c$)。其中 $\mu_c$ 代表每一個群集的群集中心，$\Sigma_c$ 代表高斯分配的共係數矩陣(covariance matrix)。也就是說，利用高斯分配($\mu_c, \Sigma_c$)不僅可以知道群集的中心點之外，還可以了解整個群集的分佈情形。所以針對模型特性具有相當良好的描述能力。而且利用高斯分配資料結構傳遞個別資料源的分群結果比傳遞整個資料源來說，資料量相對的小，因此，也避免了頻寬不足的問題。以下是 DMBC 資料分群方法的處理步驟。

1. 在每一個資料來源處，各自利用 EM 資料分群演算法進行資料分群處理，然後再將分群的結果使用高斯分配資料結構來描述每一個資料群集的特性。如圖 3 所示。其中 $k_{MAX}$ 代表機率密度函數的最大值。$K_{MAX}$ 愈大代表分群的群組數愈多，群組數愈多則與網路傳輸頻寬或品質的相依性就愈高。

```
localEM(Database D, Integer k_max)
  maxcover = 0;
  bestClustering = 0;
  for k := 1 to k_max do
    M := EM(D, k);
    if cover(M) = |D| then
      return M;
    end if
    if cover(M) > maxcover then
      maxcover = cover;
      bestClustering = M;
    end if
  end for
  return bestClustering;
```

圖 3 資料端之分群演算法

2. 將每一處資料來源的分群結果透過網路傳遞集中到同一個地方。
3. 檢視由每一個資料源所收集過來的全部資料群集結果，並且針對每一個資料群集的特性(群集中心、分佈情形)進行比較，最後將特性相近的資料群集合併，完成整體性的資料分群結果。

**4.** 將最後的整體性資料分群結果回傳給所有的參與者。

由上得知，雖然特徵模型資料分群法可以利用傳送描述群集特徵的資料結構，來解決網路頻寬不足的問題。並且使用這些特徵模型資料來產生最後的整體性資料分群結果。但也因爲如此，特徵模型設計的優劣將直接嚴重影響到最後整體性資料分群結果的正確性。

## 參、 研究方法

### 一、 演算法模型

Kriegel, Kroger, Pryakhin, Schubert [12]在他們的文章中提到，一個良好的分散式資料分群演算法至少必須要具備兩個基本的條件，一個是效能，一個是資料分群的正確性。一個演算法無論再好，若是在執行效能上不能夠有令人可以滿意的表現的話，那麼這個演算法勢必無法在現實的環境中實現。因爲時效性就是在實用上的一個重要因素。也就是說，即使利用這樣一個演算法作出來的分析結果再好，倘若錯過了應用的時機，喪失了時效的話，那麼所分析出來的結果將失去原來應該具有的價值。同理，若是所使用的演算法其所作出來的結果過於粗糙，甚至不正確的話，一樣無法在現實的環境中實現。因爲，如果使用者無法從這麼一個演算法所作出來的結果中獲取所需要的資訊，甚至因這樣的結果而做出錯誤的判斷的話，即使演算法的速度再快、效能再好，亦無法被實際的應用。

然而，在分散式資料分群演算法的設計上，效能與正確性卻是一個取捨兩難的問題。若是要講求效能、追求速度的話，那麼就必須在資料正確性上作某種程度上的妥協。若是十分在意資料正確性的話，那麼勢必在執行上就必須要耗費比較長的時間方能達成。在檢視研究一些虛擬環境資料分群演算法後，發現傳遞交換大量的資料要突破頻寬限制的障礙將是一項不容易的挑戰，但是若可以設計一個良好的資料結構用來描述分群結果與群集特性的話，不僅可以擁有較佳的執行效能，而且也能夠具有不錯的資料分群結果。所以特徵模型資料分群法便是本篇論文採用的分群方法原則。Kriegel, Kroger, Pryakhin, Schubert [12]，Tasoulis and Vrahatis [15]皆是採用特徵模型資料分群法的原則所發展出來的演算法。經過相關的文獻研究探討後發現，大多數應用特徵模型資料分群原則的演算法在資料源進行資料分群的時候，都會先限定資料源資料分群後資料集群的數量，在各自進行分群作業後，才會把這些群集資料集中一處進行分析、合併。所以群集的數目愈多，最後結果的正確性愈高。群集的數目愈少，最後結果的正確性愈低。因此個別資料源的群集數目將是決定最後結果正確性的重要關鍵因素。根據此項結果，本論文提出一個假設。由於個別資料源的群集數目是影響最後整

體性分群結果的關鍵因素，若是能在不限定資料群集數目的情況下，設計一個方法可以自動找出每一個資料源的最佳資料群集數目，然後再將這些群集資料集中一起進行分析處理的話，由於此刻集中在一起的群集皆是各個資料源中最佳的分群結果，因此勢必可以獲得到一個較爲理想的整體性資料分群結果，大幅降低了預先設定群集數目所帶來的影響。也就是說，藉由自動尋找個別資料源中最佳分群數目的方式，使得特徵模型資料分群演算法可以在執行效能以及資料正確性中找到一個可以令人滿意的平衡點。

基於上述原因，我們於是構思了一個改良式的 k-means 資料分群演算法，在每一台機器上作資料分群的時候，利用群組重疊程度的測量方式來決定最爲適當的區域群組數，再將這一個資料分群的結果傳遞給其它的參與者，由於分群的結果較爲詳細，因此當所有的群組作群集之後，將會得到更爲接近單一資料庫資料分群後的品質，改善了原本特徵模型資料分群演算法的缺點。

另外有別於計算群組幾何中心做爲群集中心的傳統 k-means 演算法，我們將採用以表示資料集中程度的密度概念，做爲我們判斷資料群集品質的方式。假定資料庫中的每一筆資料對應到空間中的一點，而每一筆資料的屬性代表這一筆資料的空間維度。接著再將這一個多維度空間切割成一個一個的網格，每一個交集的地方稱爲網點。在每一個網點的地方，定義一個密度函數 p(x)，密度函數的值是透過把每一筆資料權值化（Weighting）後加總計算所求得。

$$\rho(x_p) = \frac{1}{H}\sum_{i=1}^{N} W(x_i - x_p) \qquad (1)$$

其中 N 代表資料點的數目，H 代表網格的大小，Xi 代表第 i 個資料點位置，Xp 代表網點的位置。而計算資料點權值的權值函數如下

$$W(x - x_p) = \begin{cases} 1 - \frac{|x - x_p|}{H} & \text{if } |x - x_p| \le H \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

簡單來說密度函數就是在測量空間中的資料點於每一個網點附近的集中程度，而權值函數就是在計算資料點與網點間的距離。所以當資料點距離網點愈近，則權值愈高，資料點在網點附近愈集中，則密度函數值愈大。因此，一旦計算完所有網點的密度值後，經由密度值的大小即可判斷所包圍的區域內資料點的群集品質(集中程度)。若群集品質過低，則表示此群集在此一區域內非常分散，並不足以構成一個群集，我們就可以將其忽略不計。藉此剔除掉品質不佳的群集。

再者，我們在研究探討相關的分散式資料分群演算法文獻時可以發現，在虛擬環境資料分群演算法這一類的技術方面，其分群作業在執行時的每一個階段，每一台參與分群作業的機器都必須要提供及分

享各個階段時期的詳細狀態資料與結果。因此，當所需處理的資料量或所參與的機器數量愈來愈多時，網路上傳輸的負載負擔也將會愈來愈重。也就是說，虛擬環境資料分群演算法在執行資料分群作業的時候需要有足夠大的網路頻寬支援，方能使分群作業能夠平順進行。簡單的說，虛擬環境資料分群演算法對於網路頻寬的相依性非常的高。這也造成了這一類演算法技術在先天上所形成的限制。另一方面，以特徵模型資料分群演算法這一類的技術來說，其主要的策略在於讓各別的參與者獨自進行資料的分群作業，等到結果完成後，再把這些分群的結果集中在一起，進行最後的整理、分析、合併，完成最終的整體性資料分群結果。特徵模型資料分群演算法的關鍵在於是否可以建立一個適當的模型用來描述各自參與者所產生的資料分群結果。因爲若是參與者各自的資料分群結果可以被描述的愈詳細、愈完整的話，那麼最後分析合併後的整體性資料分群結果將會愈正確。但是，也就是因爲想要建立這麼一個良好模型的緣故，在建立模型時往往需要耗費很多的時間用來計算產生模型所需要的各項參數，以及如何設計一個合適的分析、合併方法可以正確的利用這些模型參數，藉以計算產生最後分散式資料分群演算法的最終整體性資料分群結果。

基於對 k-means 演算法以及相關的分散式資料分群技術文獻的研究與探討之後得知，若是我們想要以 k-means 演算法爲基礎開發設計出一個完整的、理想的分散式資料分群演算法的話，有幾個方面就必須要被考慮以及克服。

1. 如何定義出最適當的 k-means 演算法起始參數，使得分群的結果最爲理想。
2. 如何辨識出資料集裡面的雜訊資料。
3. 如何改善 k-means 演算法面對處理大量資料時，其執行效能不佳的問題。
4. 如何設計一個良好的資料集模型，使得它可以完整的描述出其所代表的資料群集特徵。
5. 如何設計出一個適合的資料結構，使其能以最小的儲存結構記錄最多的資料訊息，以其降低資訊在網路上傳輸時的頻寬負擔。

## 二、分群演算法

藉由上節的整理歸納得，我們所要設計出的分散式資料分群演算法必須要能解四個主要的問題。所以這個新演算法它的特徵以及目的如下：

1. 在決定如何設定適當的 k-means 演算法起始參數，以獲得理想的分群結果方面，我們改變了 k-means 演算法必須在一開始時就必須由使用者給定所需群集數作爲參數的作法。我們設計了一個新的作法。方法是不事先限定分群數，而是讓演算法逕自進行分群作業，等到作業完作後，再由演算法本身根據我們所設定的規則，自行決

定最爲適當的群集數目。如此一來，便可以改善因爲所設定的分群數不適當而造成結果失真的問題。

**2.** 在 k-means 演算法無法辨識資料集裡面的雜訊資料方面，我們設定了一個資料群集所含成員資料數目的門檻值。當某一群集所含的資料量小於這一個門檻值的時候，我們將會把此一群集視爲雜訊。用以過濾資料集裡面的雜訊資料。

**3.** 在解決如何處理大量資料方面，我們改變了 k-means 演算法針對每一個可能的資料群集中心必須計算與資料集裡頭所有資料點相隔距離的方式，改採切割區域的概念，將整個資料集空間切割成一個一個的網格(grid)，之後，在每一個被切割的子區域內根據其範圍內所包含的資料點之間的相隔距離計算求出每一個子區域的幾何中心位置，接著再套用我們所設計出的演算法求得資料集的最終分群結果。由於採用切割區域的概念，使得在分群作業的時候不需要計算每一個候選的群集中心與資料集中全部所有資料點的相隔距離，藉此加快了整個資料分群的速度。

**4.** 在選擇適當的資料模型方面，我們選擇使用幾何中心位置、群集所包含的範圍、群集所含的資料量以及群集的密度做爲模型的描述參數。以簡單的數學運算降低計算模型參數時的複雜度。

在敘述了我們所要設計的分散式資料分群演算法的特徵以及目的之後，我們將其命名爲 GridScan 演算法，並將整個演算法的處理步驟詳細敘述如下：

**1.** 首先將整個資料空間，針對每一個維度依據使用者定義的規則等比例切割成一個一個的子區域。並且移除不包含任何資料點的子區域。在決定子區域的範圍是採用系統自動設定或是由使用者來定義所要切割的規則方面，我們參考 Hinneburg and Keim [8]在其文獻中所提出的密度函數概念，不同的密度函數設定將影響資料集裡面所有資料的能見度。以及 Hierarchical Clustering 的想法，選取不同的分群階層將會影響最後的分群結果。因此我們原本傾向由使用者自行定義資料集子區域的切割距離與規則。但是因爲使用者對於資料在資料空間中分佈狀況的了解程度並不相同，所以若是將區域的切割規則由使用者自行決定的話，將無法保證分群結果的品質。於是本研究最後採用統計學領域在資料分組上常用的 Sturge Rule 法則(k = 1 + 3.322logN) 作爲資料區域切割的規則。其中 k 爲分割後的子區域數，N 爲空間內的資料總數。例如假設空間有 524200 筆的資料，則子區域總數 k = 1 + 3.322log(524200) = 20。即資料空間將被切割成 20 個子區域，空間分割

結果如圖 4 所示。

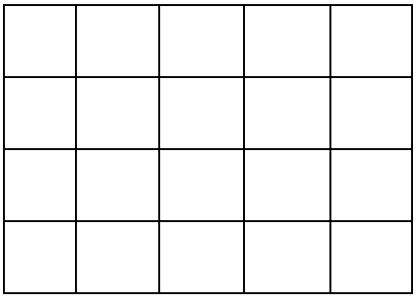

圖 4 資料空間分割

**2.** 在每一個子區域內根據所包含的資料點位置各個維度計算出每一個子區段內資料點密度最高的幾何位置以及算術平均位置。假如子區域內密度最高的位置只有一個，則此一位置即是下一次子區域移動後的中心位置。否則指定子區域的算術平均位置爲下一次子區域移動後的中心位置。其演算法簡述如下，演算法結果如圖 5 所示。

```
for subArea = 1 to n
   pn = maxDensityN(subArea)
   if pn == 1
      newP = maxDensityP(subArea)
   else
      newP = avgCenter(subArea)
```

//subArea：子區域空間。
//maxDensityN()：計算密度最高處的個數。
//maxDensityP()：取得密度最高處的位置。
//avgCenter()：取得算術平均位置。

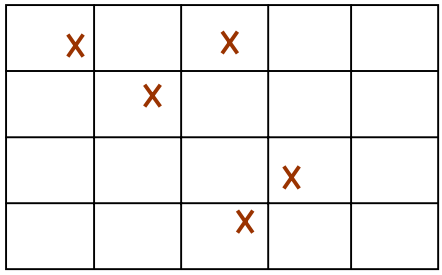

圖 5 計算出子區域內密度最高的位置

**3.** 移動子區域直到步驟(2)計算出的中心位置位於子區域的中央爲止。如圖 6 所示。

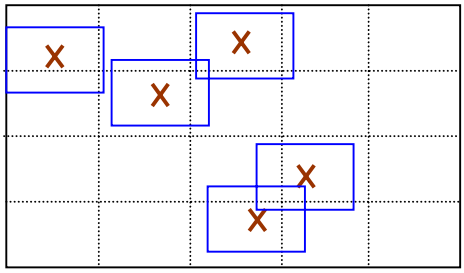

圖 6 移動網格

**4.** 重複步驟(2)、(3)直到每一個子區域的中心點不再移動爲止。在移動過程中，或許有可能會有子區域發生重疊的現象，雖然這個時候進行子區域合併的動作會加快整個演算法計算過程的速度，但是此刻進行合併的話卻有可能提早將群集品質不佳的資料群吸收進來，嚴重影響最後計算的結果。因此合併的動作將會留到所有子區域的候選中心點不再變動爲止。

**5.** 檢查是否有子區域重疊現象發生。由於子區域的中心位置皆是該區域中密度最高的地方，而距離中心位置愈遠，也就表示其資料點的集中程度愈差。因此我們可以合理的把每一個子區域都視爲是一個常態(鐘形)的分配情況。而根據契比雪夫定理以及經驗法則定理得知，當由中心位置涵蓋範圍到達前後各一個標準差的時候，其將包含大約 68%的資料量。因此根據定理，我們設定若兩個區域重疊時，假如重疊範圍內的資料含量超過其中某一個子區域所有資料量的 32%上的時候，則我們就把這兩個子區域合併成爲一個子區域。演算法如下：

```
overlap[] = checkOverlap(subArea[])
for i = 1 to n
   dataN = countNumber(overlap[i])
   dataRate = minRate(dataN,
      overlapArea[])
   if dataRate >= 0.32
      mergeArea(overlapArea[])
```

//overlap[]：重疊空間集合。
//subArea[]：子區域空間集合。
//overlapArea[]：發生重疊的子區域空間。
//checkOverlap()：找出重疊的空間。
//countNumber()：計算重疊空間內資料總數。
//minRate()：計算重疊空間與相關子區域空間資料量比例的最小值。
//mergeArea()：合併子區域空間。

**6.** 重複步驟(5)直到沒有子區域需要再合併爲止。

**7.** 重新計算每一個子區域的中心位置以及子區域每一個維度的全距(每一維度的最小值與最大值)。

**8.** 由每一個維度依序向外擴大，根據契比雪夫定理以及經驗法則定理得知，當由中心位置涵蓋到前後各 3 個標準差時，其範圍將包含了大約 95%以上的資料量。因此我們假設每一個子區域皆爲近似包含中心位置前後各 3 個標準差的距離。所以當子區域擴大爲包含前後各增加 1 個標準差的範圍時，若是所增加範圍的資料量大於原先子區域內所有資料量的 5%以上時，我們便可以合理的決定擴大這個子區域。

**9.** 重複執行步驟(8)，直到所有的子區域皆不再需要擴大爲止。

**10.** 藉由計算每一個群集的密度，即可得知資料集裡面所有資料群集的品質，然後過濾掉密度小於使用者所設定的門檻值者。剩下的群集數目即是此資料集的代表群集。

**11.** 將每一個資料集的代表群集透過網路傳輸集中到一處。內容包括每一個資料群集的幾何中心、群集密度、維

度全距以及群集所包含的資料量。

**12.** 執行步驟(6)、(7)即可得到我們所要的最終的分散式資料資料分群結果。

## 三、 系統架構

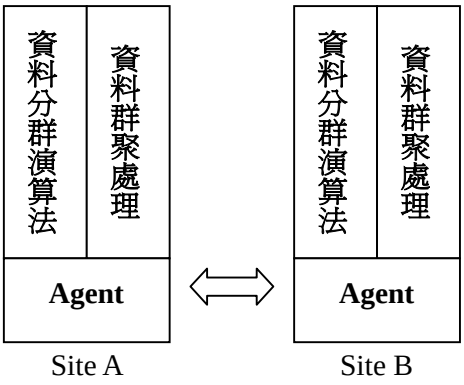


圖 7 系統架構

如圖 7 所示，藉由結合代理人(Agent)以及 GridScan 資料分群演算法可以得到如圖八的系統架構。每一台機器先行在自己內部利用 GridScan 資料分群演算法得到屬於自己最佳的分群資料，然後再透過代理人機制將分群資料傳送給網路中的其他參與者。由於是最佳化的分群資料，所以當資料整合後將不會造成太大的失真，也由於網路資訊傳遞的效率提升，加快了分散式資料分群的速度與節省了大量的處理時間。

## 肆、 實驗與分析

在實驗與分析階段，我們將比較自行設計的GridScan演算法、Dhillon, Modha [3]所提出的虛擬環境資料分群演算法(分散式 k-means)，以及 Kriegel, Kroger, Pryakhin, Schubert [12]所提出的特徵模型資料分群演算法(DMBC)。

在分散式 k-means 演算法方面，由其演算法步驟與系統架構得知，分散式 k-means 演算法的效能與資料量的多寡成反比以及與參與計算的處理器的數量多寡成正比。也就是說當資料量愈多時，分散式 k-means 演算法的效能愈低。而當參與計算的處理器愈多時，分散式 k-means 演算法的效能就愈好。不過，在實際上，所需分析的資料量往往遠大於參與計算的處理器數量，而資料量增長的速度也遠大於處理器增加的速度。因此可以說，分散式 k-means 演算法的效能與資料量多寡的相依性最爲明顯，當資料量愈多時，分散式 k-means 演算法的效能就會愈低。在 DBMC 演算法方面，由於演算法的目的著重在群集特徵描述的正確性以及分群結果的品質上，所以在區域伺服器(Local Site)以及中央伺服器(Center Site)上皆採用了 EM 演算法做爲資料分群的核心技術。也因此在整個計算的過程中使用了大量複雜的數學運算，例如根號運算、積分運算等等。所以十分耗費處理器的運算資源。雖然避免掉了網路頻寬限制與網路傳輸品質影響的問題，但是卻增加了區域伺服器運算的時間，造成區域伺服器的負擔。在 GridScan 演算法方面，因爲資料集預先做了分割，所以每一個候選的群集中心只需要與所分配子區域內的資料點作運算，而不用涉及

資料集內所有的資料點，同時整個計算過程只有用到一般的算術運算，因此加快了資料分群運算的速度。因此在演算法效能上預估比虛擬環境資料分群演算法以及特徵模型資料分群演算法爲好。

我們將設計兩組實驗環境：兩台機器以及四台機器，使用相同的資料，但會依實驗環境的不同而會分割成 2 份以及 4 份。同時會針對資料的分散程度進行模擬設定。大致上可以區分成三種情況。第一種情況是假定某一個整體性資料群集的成員是特別集中在一個資料來源或是極少數個資料來源，第二種情況是假定某一個整體性資料群集的成員是平均分散在所有的資料來源或是極大多數的資料來源，第三種情況是假定在所有的資料來源中被視爲雜訊的資料記錄，是否有可能會成爲整體性資料群集中的一個群集。測量的向度將分爲資料處理時間與資料分群品質。預期在資料處理時間上可以比虛擬環境資料分群演算法以及特徵模型資料分群演算法爲快，而資料分群結果品質可以優於虛擬環境資料分群演算法，而接近於特徵模型資料分群演算法。

在實驗環境方面，我們使用安裝 windows 2000 作業系統以及 ms sql 2000 的三部伺服器電腦。並且依據上述之實驗模擬情境隨機產生 20000 筆資料、40000 筆資料、60000 筆資料、80000 筆資料以及 100000 筆資料作爲本實驗之分析資料。圖 8 爲分散式 k-means、DBMC 以及 GridSacn 在執行結果上的效能比較。由實驗結果可以知道當資料量愈大時，無論是分散式 k-means 或是 DBMC 其資料處理時間都會隨著資料量的增加而有明顯的增加，但是 GridScan 在資料處時間的增加上卻不明顯。這是由於本研究的方法在資料量增加時其所需處理的網格數量小於總資料量的情形將會愈明顯，因此整體資料分群效能會比其他兩種方法爲好。

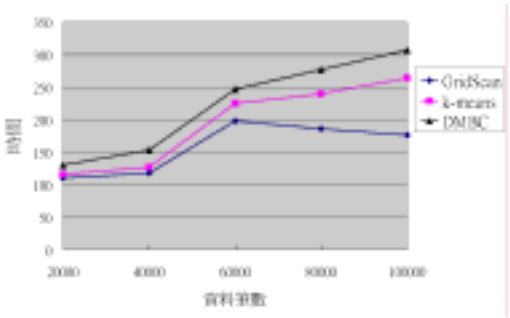


圖 8 分群方法效能比較

圖 9、圖 10 圖 11 分別爲 20000 萬實驗資料時 GridScan、分散式 k-means 以及 DMBC 三種方法資料分群的結果，其中圖 9 內深藍色菱形的點所代表的是雜訊資料。而表 1 則是三種方法之資料分群各個群組成員到群組中心的距離平均值資料。由此三種方法的資料分群結果圖示以及表 1 比較可以得知，GridScan 資料分群方法由於可以過濾掉資料來源中的雜訊資料，因此，使得資料分群的結果可以不受到雜訊資料影響，最後獲得一個較佳的資料分群品質。

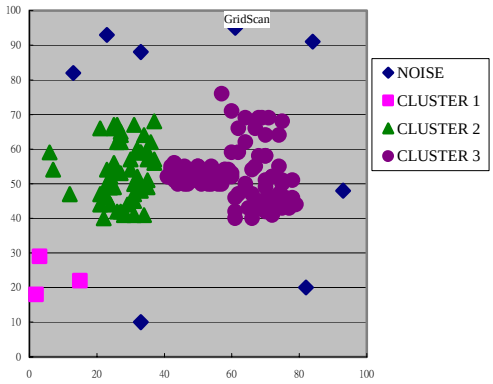


圖 9 GridScan 資料分群結果

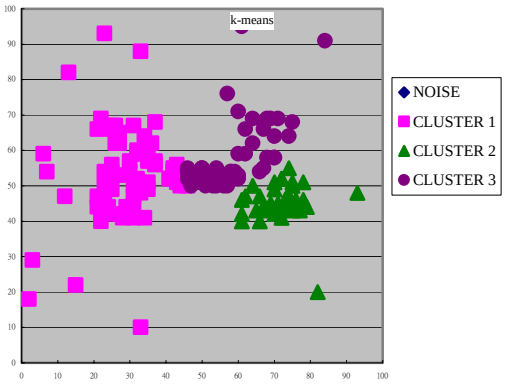


圖 10 分散式 k-means 資料分群結果

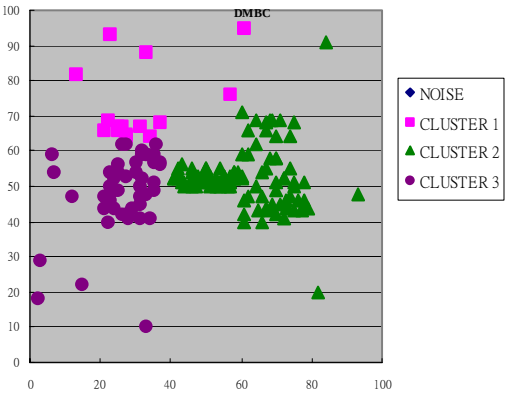


圖 11 DMBC 資料分群結果

| | 群組 1 | 群組 2 | 群組 3 | 內平均距離 |
|---|---|---|---|---|
| GridScan | 7.421 | 9.632 | 11.548 | 9.534 |
| 分散式 k-means | 13.130 | 7.026 | 10.346 | 10.167 |
| DBMC | 13.500 | 12.313 | 10.942 | 12.252 |

表 1 群組內平均距離表

## 伍、結論

將資料儲存於單一儲存體透過 DBMS 來管理與存取，並且提供給傳統 Data Mining 演算法做爲分析來源的方式，由於資料庫被大量的使用以及網路技術的發達，企業內的資料量迅速成長的緣故，使得傳統 Data Mining 分析的效能愈來愈不理想，因此將資料分散儲存統一管理的概念愈來愈普及。分散式資料分群演算法的技術也開始被提出來討論。

而到目前爲止，分散式資料分群演算法的技術大致上可以將其區分爲兩大類。第一大類稱爲虛擬環境資料分群演算法。其主要概念是希望將分散式的環境透過邏輯的結構整合成一個虛擬的單一環境，並且利用所參與計算作業的處理器共同合作的方式來提升分群作業的整體效能。但是實際上由於資料量成長的速度遠超過參與計算的處理器數量，因此，分群作業效能的提升仍然有限。第二大類稱爲特徵模型資料分群演算法。其主要概念是希望雖然

分群結果是由資料存放的來源處各自計算，但是在整合所有的資料分群結果後，仍然可以保有良好的資料分群品質。不過，由於使用複雜數學運算來建立特徵模型的緣故，雖然保有了分群的品質，卻使得整體分群作業的效能不理想。在本研究中我們提出了一個稱爲 GridScan 的分散式資料分群演算法，利用分割區域的概念以及簡單的統計模型量數，企圖在計算效能與分群品質上取得一個可以接受的平衡點。

在實驗驗證方面，我們選擇分散式 k-means[4]以及 DMBC[12]作爲比較的對象。由實驗證明(如圖 8 所示)GridScan 與其他分群演算法效能比較的曲線結果可以得知，GridScan 分割區域的概念確實可以克服在大量資料的情況下，分群演算法效能下降的問題。不僅如此，由圖 8 的曲線變化可以看出，GridScan 的資料分群效能當資料量愈來愈大時，其效能優異性愈是明顯。另外在分群品質的比較實驗(如圖 9、圖 10、圖 11)中可以得知，GridScan 具有雜訊過濾的能力，可以汰除掉關聯性很低的雜訊資料，提高分群結果的品質。而由表一群組內平均距離的實驗結果更可以證明 GridScan 確實較其他分群方法有更佳的分群品質。

## 參考文獻


1. Berry, Linoff, “mastering data mining”, wiley, 2000.
2. Bradley, P. S., U. Fayyad, C. Reina, "Scaling Clustering Algorithms to Large Databases", Proc. 4th Intl. Conf. on Knowledge Discovery and Data Mining (KDD98), AAAI Press, 1998.
3. Dhillon, I. S., D. S. Modha, “A Data-Clustering Algorithm on Distributed Memory Multiprocessors”, Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD, 1999.
4. Du, W., G.. Agrawal, “Developing Distributed Data Mining Implementations for a Grid Environment”, CCGRID, 2002.
5. Gonzalez, R., A. Kamrani, “Overview of data mining: A survey of methodologies and technologies for data mining and intelligent data discovery”, Data mining for design and manufacturing**,** 2001.
6. Halkidi, M., Y. Batistakis, M. Vazirgiannis, “Clustering validity checking methods: part II”, ACM SIGMOD Record, v.31 n.3, Sept, 2002.
7. Han, Kamber, “Data mining. Concepts and Techniques”, Morgan Kauffman, 2001.

8. Hinneburg, A., D. A. Keim, “An Efficient Approach to Clustering in Large Multimedia Databases with Noise”, Proc. 4rd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, 1998.
9. Jain, A. K., M. N. Murty, P. J. Flynn, “Data clustering: a review, ACM Computing Surveys (CSUR)”, v.31 n.3, p.264-323, Sept. 1999.
10. Jang, Sun, Mizutani, “Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence Companion software available”, Prentice Hall, 1997.
11. Kargupta, H., I. Hamzaoglu, B. Stafford, “Scalable, Distributed Data Mining”, An agent Architecture. KDD, 1997.
12. Kriegel, H. P., P. Kroger, A. Pryakhin, Schubert M., “Effective and Efficient Distributed Model-based Clustering”, Fifth IEEE International Conference on Data Mining, 2005.
13. Lin, C. R., M. S. Chen, “Combining Partitional and Hierarchical Algorithms for Robust and Efficient Data Clustering with Cohesion Self-Merging”, IEEE Trans. Knowl. Data Eng. 17(2): 145-159, 2005.
14. Oguchi, M., M. Kitsuregawa, “Dynamic remote memory acquisition for parallel data mining on ATM-connected PC cluster”, Proceedings of the 13th international conference on Supercomputing, p.246-252, June 20-25, 1999.
15. Tasoulis, D. K., M. N. Vrahatis, “Unsupervised Distributed Clustering”, Parallel and Distributed Computing and Networks, 2004.