

## 快速雲端 $k$ -means 分群方法初探

廖怡欽<sup>1,\*</sup> 羅尹呈<sup>2</sup> 張維倫<sup>2</sup>

<sup>1</sup> 南華大學資訊工程系教授

<sup>2</sup> 南華大學資訊工程系學士

### 摘 要

雲端運算技術可輕易將資料儲存與計算工作分配給多個節點處理，非常適合處理巨量資料分群工作。目前雖有針對雲端運算環境設計的  $k$ -means 分群方法可用，但未善用傳統快速  $k$ -means 分群技術所帶來的好處。為快速瞭解傳統快速  $k$ -means 分群技術在雲端運算環境下的效能表現，本論文整合一個常見的快速  $k$ -means 分群技術至現有的雲端  $k$ -means 分群方法中，提出一套快速雲端  $k$ -means 分群方法，所提方法會產生跟傳統  $k$ -means 分群方法相同的結果。實驗結果顯示，快速雲端  $k$ -means 分群方法確實可有效提升現有雲端  $k$ -means 分群方法的運算效能。

**關鍵詞：**大數據、快速  $k$ -means 分群、雲端運算。

---

\* 通訊作者 E-mail: ycliaw@nhu.edu.tw

DOI: 10.6553/JKTUET.2017.0302.03

## A Preliminary Research on Fast Cloud $k$ -means Clustering

Liaw, Yi-Ching<sup>1,\*</sup> Lo, Yin-Cheng<sup>2</sup> Zhang, Wei-Lun<sup>2</sup>

<sup>1</sup>Professor, Information Technology, Nanhua University

<sup>2</sup>Bachelor, Information Technology, Nanhua University

### ABSTRACT

Cloud computing technology can easily distribute data and computing tasks to multiple nodes and is very suitable to be used to cluster a big data. There are some cloud  $k$ -means clustering methods available. But, the current methods do not leverage the advantage of the traditional fast  $k$ -means clustering techniques. To quickly understand that if the traditional fast  $k$ -means clustering techniques is still useful in the cloud computing environment. This paper integrates a well-known fast  $k$ -means clustering technique with a cloud  $k$ -means clustering method and proposed a fast cloud  $k$ -means clustering method. The proposed method will generate the same clustering result as the traditional  $k$ -means. Experimental result shows that the proposed method can really improve the performance of the cloud  $k$ -means clustering method.

**Key words:** big data, fast  $k$ -means clustering, cloud computing.

---

\* Corresponding author, E-mail: ycliaw@nhu.edu.tw

DOI: 10.6553/JKTUET.2017.0302.03

## 一、前言

隨著網際網路技術的進步以及連網裝置的增加，透過網際網路取得的資料數量、資料種類、以及資料內容的改變速度均呈現大幅度的成長，由於這類資料數量龐大，一般將其稱為巨量資料 (Big Data) [1,2]。巨量資料中隱藏許多重要的資訊，妥善分析、取得、及善用這些資訊，可有效加快人類社會進步的腳步。例如：藉由記錄及分析使用者網頁瀏覽行為，瞭解使用者行為模式，可用來改善網頁設計方式，提高網頁介面的友善度；藉由記錄及分析特定疾病病人的基因組，瞭解導致特定疾病的基因，可用來檢測人們罹患特定疾病的可能性，降低特定疾病發病率及危害程度；藉由記錄及分析製程參數與生產良率的關係，瞭解製程參數對生產良率的影響，可用來找出較佳的製程參數設定方式，提高生產良率；藉由記錄及分析顧客購物行為，瞭解顧客購物行為模式，可依顧客喜好設計產品行銷方式，提高產品行銷效果與銷售成功機率。巨量資料技術應用廣泛，不同領域可搜集各自領域的資料，進行分析，過濾出有用的資訊，改善現有的作業方式。

巨量資料技術應用流程中由搜集資料到取得可用資訊的過程稱為知識探索過程 [3]。一般知識探索過程包括資料搜集、資料淨化、資料分析、與知識萃取等 4 個步驟 [4]。資料搜集步驟負責決定要搜集的資料項目、搜集管道、搜集資料、轉換資料格式、以及儲存資料等工作；資料淨化步驟負責處理有問題與錯誤的資料；資料分析步驟負責分析資料，找出資料的關連性，產生有用的知識，再將相關知識儲存到知識庫。知識萃取步驟則依實際需求由知識庫中提取合適的知識解決問題。在知識探索過程中，資料分析與知識萃取步驟，對於是否能夠有效的由資料中取出有用的資訊，扮演著相對重要的角色。好的資料分析方法可快速過濾出有用的資訊，提高後續知識萃取步驟的處理速度與準確度。

資料分群 (data clustering) 技術 [5,6] 是一種簡單有效的資料分析技術，廣泛應用在許多領域（資料探勘、樣型識別、人工智慧、商業行為分析、顧客行為分析……等）中，可將資料依資料內容的相似程度分成多個不同的群組。要將資料分群並不容易，求取資料分群的最佳解是 NP-Hard 問題 [5]，為了在有限時間內產生可接受的分群結果，目前已有許多使用經驗法則 (heuristic) 的資料分群方法可用 [5,6]。其中，最常見的資料分群方法為  $k$ -means [7] 分群方法。要將  $n$  筆資料（每筆資料可視為資料

表示空間中的一個資料點）分成  $k$  個群組， $k$ -means 分群方法的作法是一開始先產生  $k$  個群組，並為每個群組選取一個初始中心點（常見的作法是由資料集中隨機選取一筆資料作為一個群組的中心點）；接著，計算所有資料點與所有群組中心點的距離，將所有資料點分配到距離最近的群組內；資料點分配完畢，再使用相同群組內所有資料點的平均值作為該群組的新中心點；重覆執行資料點分配以及群組中心點計算步驟，直到群組內的資料點不再變動或達到特定終止條件（重覆次數超過指定次數或變動情況少於指定數值），即完成資料分群工作。

使用  $k$ -means 資料分群方法，在每個重覆過程中，必須計算所有資料點到所有群組中心點的距離，以及計算新的群組中心點。每個重覆過程的時間複雜度為  $O(nk)$  個距離計算，若重覆次數為  $t$ ，則整個分群過程需要  $O(tnk)$  個距離計算。 $k$ -means 分群方法雖然可在多項式時間內完成資料分群工作，但若資料量龐大 ( $n$  很大)，計算複雜度仍然相當高。為了降低資料分群方法的計算複雜度，目前已有許多快速  $k$ -means 分群方法出現 [8-16]，快速  $k$ -means 分群方法主要是利用加快群組中心點的搜尋過程或依據群組中心點的變動情形，減少資料點與群組中心點距離的計算量，達到效能提升的效果。由於所減少掉的距離計算不會影響最近群組中心點的搜尋結果，因此這些快速  $k$ -means 分群方法可以產生跟原本  $k$ -means 完全相同的分群結果。

在巨量資料的應用環境中，資料數量通常很大，大到難以使用傳統資料分群方法來處理巨量資料的分群工作。雲端運算環境 [17,18] 具有平臺開放性、高擴充性，以及低成本等優勢，可將資料儲存與計算工作分散到多個雲端節點上，非常適合用來處理巨量資料。目前的雲端運算環境中，已有  $k$ -means 分群工具 [19] 及相關演算法 [20-22] 可供使用。現有的雲端  $k$ -means 分群方法雖可利用雲端環境解決巨量資料分群的問題，但未善用傳統快速  $k$ -means 分群技術 [8-16] 所帶來的好處。為了初步瞭解傳統快速  $k$ -means 分群技術應用至雲端運算環境下的效能表現，本論文將一個常見的快速  $k$ -means 分群技術整合至現有的雲端  $k$ -means 分群方法中，開發出一個快速雲端  $k$ -means 分群方法。藉由分析所提方法與現有雲端  $k$ -means 分群方法的效能差異，作為後續開發高效能快速雲端  $k$ -means 分群方法的有力參考依據。

本論文其餘章節安排如下，第二章介紹現有的  $k$ -means 分群方法、雲端運算平臺 Hadoop、以及雲

端  $k$ -means 分群方法；第三章描述本論文所提的快速雲端  $k$ -means 分群方法；第四章介紹實驗結果；第五章則提出結論與未來研究方向。

## 二、背景知識

以下各節分別介紹  $k$ -means 分群方法 [7]、雲端運算平臺 Hadoop [18]、以及 Hadoop 雲端運算平臺上的雲端  $k$ -means 分群工具 Mahout [19] 等背景知識。

### 2.1 $k$ -means 分群方法

給定一個內含  $n$  個資料點的資料集  $S = \{x_i \mid 1 \leq i \leq n, x_i \in R^d\}$ ，其中  $x_i = [x_{i,1}, x_{i,2}, \dots, x_{i,d}]^T$  代表一個具  $d$  個維度的資料點。 $k$ -means 分群方法的目的是將  $S$  分成  $k$  個群組，產生一個群組集合  $P = \{C_i \mid 1 \leq i \leq k\}$ ，其中， $C_i$  代表第  $i$  群資料點所成的集合，對所有符合  $1 \leq i, j \leq k$  且  $i \neq j$  的  $\langle i, j \rangle$  對而言， $C_i \cap C_j = \phi$ ，且所有群組的聯集等於  $S$ ，也就是：

$$S = \bigcup_{i=1}^k C_i \quad (1)$$

$k$ -means 分群方法的作法是先為每個群組  $C_i$  產生各自的中心點  $c_i$ ，其中， $c_i$  的初始值可由  $S$  中隨機選取產生。然後重覆執行以下兩個步驟，直到所有群組內的點不再變動或達到特定的終止條件（重覆次數超過指定次數或中心點變動量少於指定數值）為止。

#### 1. 群組指派步驟：

幫  $S$  內的每個資料點  $x_i$  指派一個群組  $C_j$ ，指派群組的作法是由所有的群組中心點中找到一個與  $x_i$  最相似的中心點。令  $c_j$  是與  $x_i$  最相似的中心點，也就是  $D(x_i, c_j) \leq D(x_i, c_l)$ ，其中  $1 \leq j, l \leq k$  且  $j \neq l$ ， $D(.,.)$  是距離函式。距離的計算通常使用歐基里德距離，任意兩點  $a = [a_1, a_2, \dots, a_d]^T$ ， $b = [b_1, b_2, \dots, b_d]^T$  的歐基里德距離計算公式定義如下：

$$D(a, b) = \sqrt{\sum_{i=1}^d (a_i - b_i)^2} \quad (2)$$

#### 2. 群組中心點更新步驟：

為每個群組計算新的中心點，令  $C_j = \{x^{c_j}_i \mid 1 \leq i \leq n_{c_j}\}$ ，其中  $x^{c_j}_i$  代表群組  $C_j$  中第  $i$  個資料點， $n_{c_j}$

代表群組  $C_j$  的資料點個數，新的群組中心點  $c_j$  計算公式如下：

$$c_j = \frac{\sum_{i=1}^{n_{c_j}} x^{c_j}_i}{n_{c_j}} \quad (3)$$

$k$ -means 分群方法的完整運作流程如圖 1 所示。

### 2.2 Hadoop 雲端運算平臺

Hadoop 是一個開放原始碼的雲端運算平臺 [18]，此平臺參考 Google 於 2003 年與 2004 年發表的〈The Google File System〉[23] 和〈MapReduce: Simplified Data Processing on Large Clusters〉[24] 兩篇論文，實作出 Hadoop 的分散式檔案系統（HDFS）以及 MapReduce 資料處理技術，HDFS 負責資料儲存工作，MapReduce 則負責工作分配與執行。

Hadoop 雲端運算平臺使用叢集架構來儲存與處理資料，叢集架構由多個節點組成，每個節點是一個可獨立運作的儲存及運算單元。一個叢集內含一個名稱節點（name node）與多個資料節點（data node），名稱節點負責管理及監控資料，資料節點則負責儲存資料。資料儲存在 HDFS 內，會被切割成多個區塊並分散儲存在不同的資料節點上，為了避免資料損壞所造成的資料遺失，每個區塊會被複製多份儲存在多個資料節點上。當有資料損壞時，名稱節點可取用位於其他節點上的資料複本，並隨時保持資料在系統中維持在一定份數的狀態。由於

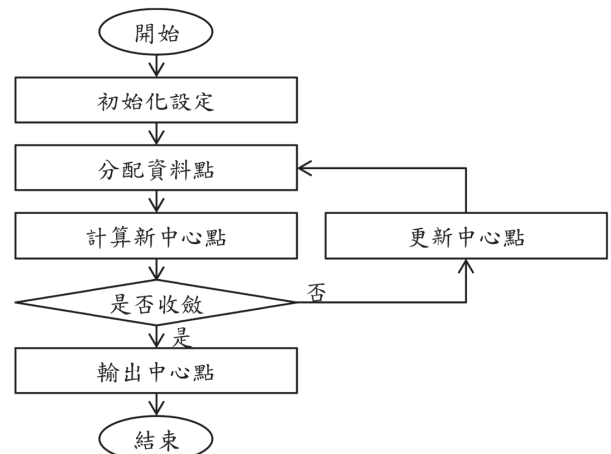


圖1  $k$ -means分群方法運作流程

同一份資料會有多份存放在不同的資料節點內，資料處理工作進行時，也可減少部分資料傳輸時間。

MapReduce 的工作分配與執行則分別由工作分配節點 (job tracker) 與工作節點 (task tracker) 負責。在 MapReduce 架構下，每個運算工作包括 Map 與 Reduce 兩個過程。在 MapReduce 運算過程中，工作分配節點會將 Map 與 Reduce 工作分配給多個工作節點。分配到 Map 工作的工作節點負責為所分配到的資料進行鍵值匹配工作，產生中間結果。中間結果包含多個鍵值(key)與數值串列(value)對，每個鍵值與數值串列對內含一個鍵值以及一個數值串列，該數值串列是在 Map 工作中對映到該鍵值的所有資料的集合。Map 工作節點所產生的中間結果經過洗牌與排序 (shuffle and sort) 過程後，會將整理過的中間資料傳送給適當的 Reduce 工作節點，對所接收到的中間結果進行合併動作，產生最後的結果。MapReduce 運作流程如圖 2 所示。

### 2.3 雲端 $k$ -means 分群方法

Mahout [19] 是一套建構在 Hadoop 運算環境上的機器學習工具庫，該工具庫提供許多機器學習相關的函式庫，其中也包括  $k$ -means 分群工具，可供用來為巨量資料分群。

Mahout 的  $k$ -means 分群工具運作流程如圖 3 所示。首先進行初始化設定，初始化設定過程會進行程式的組態設定以及讀取待處理的資料集與所有群組的初始中心點。然後，執行 Map 運算為每個

資料點找出與其距離最近的群組中心點並產生初步的分群結果。初步分群結果中，鍵值為中心點，數值串列則為子資料集中離此中心點最近的所有資料點。接著，執行 Reduce 運算為每個群組計算新的中心點。其中，Map 與 Reduce 運算會分配給多個工作節點執行，每個工作節點執行一部分工作。為所有群組產生新中心點後，再計算新舊中心點的差異並依據該差異值與指定的收斂參數判斷是否收斂。如果收斂，輸出新中心點並結束程式；否則使用新的群組中心點取代舊有的群組中心點，然後再重覆執行 Map 與 Reduce 運算。Map 與 Reduce 運算過程會重覆執行許多次，直到達到收斂條件為止。

除了 Mahout 所提供的  $k$ -means 工具外，近年也有一些用來改善雲端  $k$ -means 運算效能的演算法出現 [20-22]，這些分群方法主要是利用改變群組初始中心點的產生方式來改變分群收斂時間以及分群效果，資料分群運作流程基本上跟 Mahout 的  $k$ -means 分群方法運作流程類似。

### 三、快速雲端 $k$ -means 分群方法

現有的雲端  $k$ -means 資料分群方法的運作流程 [19-22] 如第二章所述，主要計算工作落在 Map 運算上。Map 運算負責為每個資料點找出最相似的群組中心點。要為一個資料點找出與其最相似的群組中心點，必須計算該資料點與所有群組中心點的距離（距離計算方法如式 (2) 所示）。此部分的計算

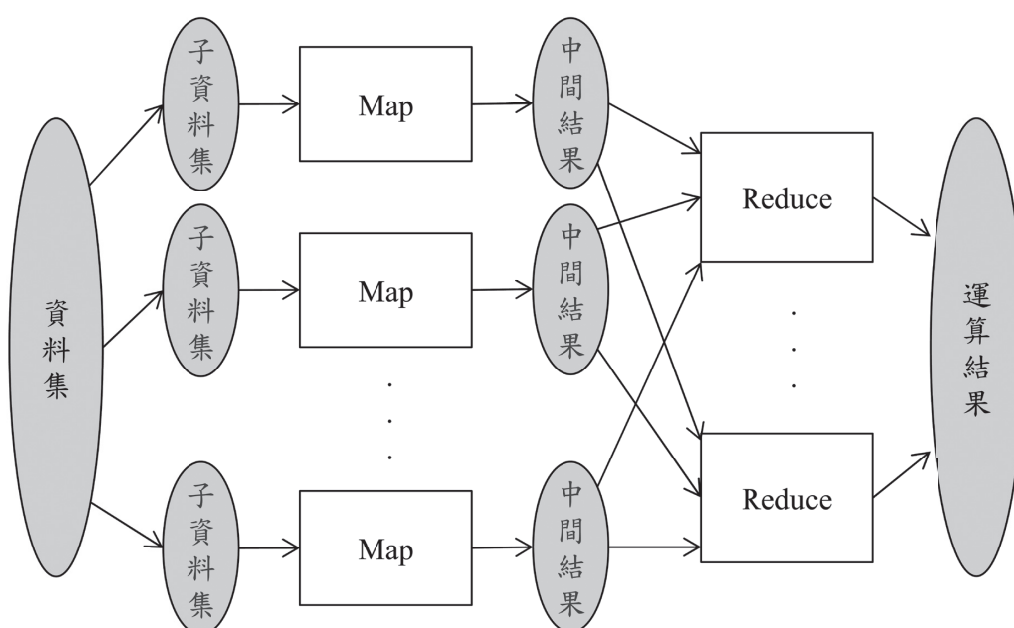


圖2 MapReduce運作流程

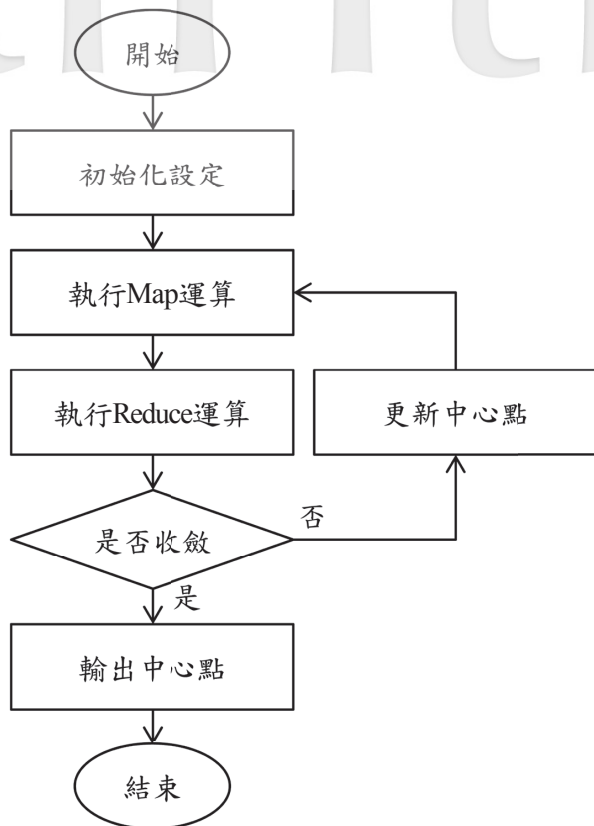


圖3 Mahout的k-means分群方法運作流程

量與群組中心點的數目  $n$  以及資料維度  $d$  成正比，當群組中心點的數量或資料維度大時，運算量會變得很大。

為了降低運算量，目前已有許多快速資料分群方法 [8-16] 可用來降低這部分運算的複雜度。這些方法主要運作原理是取出群組中心點的特徵，在計算過程中使用這些特徵以及簡單的特徵過濾公式過濾掉特徵差異太大的群組中心點（被過濾掉的群組中心點不可能為距離最近的群組中心點，所以被過濾掉也不會影響分群結果），節省許多不必要的距離計算。

現有快速資料分群方法雖可有效提升單機版資料分群法的運算效能，但套用到雲端運算環境下未必同樣有效。主要原因是快速資料分群方法，必須先對群組中心點進行前置處理，再為資料點與群組中心點進行配對動作。單機執行時，所有資料點的配對工作都在同一臺機器執行，群組中心點的前置作業時間對比所有資料點的配對時間相對較小。在雲端運算環境下，資料點與群組中心點的配對工作會分散給多個工作節點處理，群組中心點前置處理時間對比整體運算時間相對較大。對需要花費較

多時間進行前置作業的方法來說，不見得可以在雲端環境下得到好處。再者，雲端環境下的資料與工作分配行為，也會影響實際運算時間。

為降低前處理時間對所開發方法的影響，本論文選擇使用具有較低前置作業處理時間的快速搜尋方法 [12] 來加速 Map 運算中的資料配對工作。所使用的方法利用資料點的平均值（式 (4)）作為資料點的特徵，加速群組中心點的搜尋工作。式 (4) 中， $m_x$  表資料點  $x$  的平均值， $d$  為資料點的維度：

$$m_x = \frac{\sum_{i=1}^d x_i}{d} \quad (4)$$

使用資料點的平均值作為特徵值，在實際進行資料點與群組中心點配對工作前，必須先計算所有群組中心點的平均值並將群組中心點進行排序。實際配對時，再計算資料點的平均值；然後找出平均值與資料點平均值最相似的群組中心點；接著，計算該群組中心點與資料點的距離；再依平均值差異由近而遠依次拜訪群組中心點。如果該群組中心點通過過濾條件的檢查，表示此群組中心點有可能是最相似的群組中心點，則必須計算此群組中心點與資料點的距離並更新最近距離以及最近群組中心點的資訊；否則表示此群組中心點不可能是最相似的群組中心點，因此不用計算距離。使用資料點平均值與群組中心點平均值的差異值過濾不可能的群組中心點，可節省掉大量的距離計算。

令  $m_{c_i}$  與  $m_{x_j}$  分別是群組中心點  $c_i$  與資料點  $x_j$  的平均值， $D_{min}$  為資料點  $x_j$  與其在計算過程中所找到之最接近群組中心點的距離。則過濾條件定義如下：

$$\sqrt{d} |m_{c_i} - m_{x_j}| \leq D_{min} \quad (5)$$

結合前處理步驟以及以上過濾條件，本論文所提之快速雲端 k-means 分群方法的運作流程如圖 4 所示：

圖 4 加入群組中心點前處理功能以及修改原 Map 運算的作法。群組中心點前處理的工作主要是計算群組中心點的平均值以及將群組中心點依平均值由小到大排序。令由小到大排序後的中心點資料為  $\{c'_i : 1 \leq i \leq k\}$ ，而群組中心點平均值的資料為  $\{m_{c'_i} : 1 \leq i \leq k\}$ 。快速 Map 運算的主要工作是使用快速群組中心點搜尋演算法，幫每個資料點找出

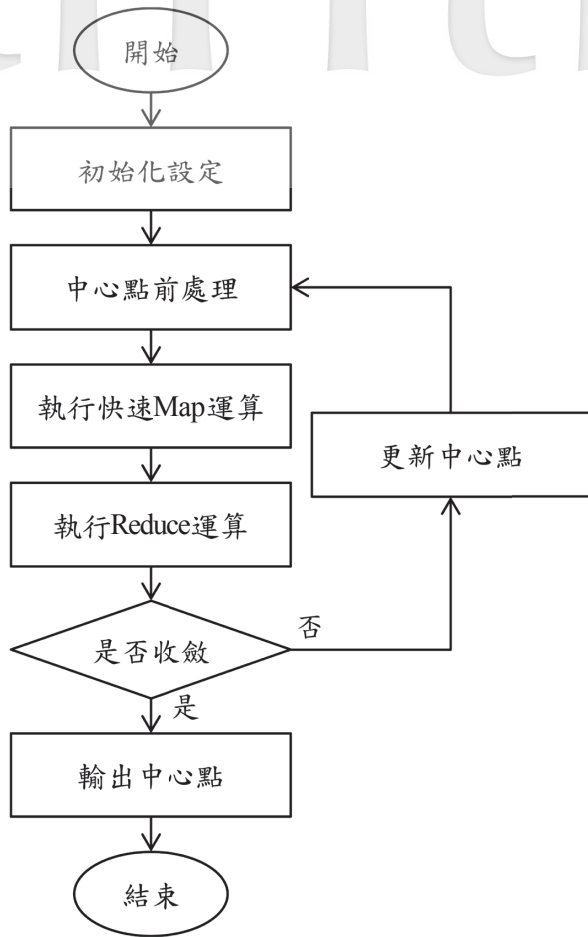


圖4 快速雲端k-means分群方法運作流程

最相似的群組中心點。令  $x$  是資料點，資料點  $x$  的快速群組中心點搜尋演算法如下：

1. 計算  $x$  的平均值  $m_x$  以及使用二元搜尋法找出平均值與  $m_x$  最接近的中心點  $c'_f$ 。
2. 計算  $D_{min} = D(x, c'_f)$  以及設定  $near = f$ 。
3. 令  $p = 1, p_{min} = 1, p_{max} = m$ 。
4. 假如  $(f + p) > p_{max}$ ，跳到步驟 5。
- (1) 檢查式 (5) 是否符合，如不符合，設定  $p_{max} = f + p$  後跳到步驟 5。
- (2) 計算  $dist = D(x, c'_{f+p})$ ，若  $dist < D_{min}$ ，設定  $D_{min} = dist$  及  $near = f + p$ 。
5. 假如  $(f - p) < p_{min}$ ，跳到步驟 6。
- (1) 檢查式 (5) 是否符合，如不符合，設定  $p_{min} = f - p$  後跳到步驟 6。
- (2) 計算  $dist = D(x, c'_{f-p})$ ，若  $dist < D_{min}$ ，設定  $D_{min} = dist$  及  $near = f - p$ 。

6. 假如  $(f - p) \leq p_{min}$  而且  $(f + p) \geq p_{max}$ ，跳到步驟 7，否則設定  $p = p + 1$  後跳到步驟 4。

7. 結束程式並輸出  $c'_{near}$ 。

## 四、實驗結果

為了評估現有雲端  $k$ -means 分群方法以及本論文所提的快速雲端  $k$ -means 分群方法的效能表現，本論文進行了兩組實驗，第一組實驗固定分群數目然後調整資料集中資料點的數量，看看資料點的數量對運算時間的影響。第二組實驗則固定資料點的數量，但調整分群的數目，看不同的分群數目對執行時間的影響。由於  $k$ -means 執行效能受初始群組中心點的影響很大，為減少實驗結果受初始群組中心點的影響，實驗前先隨機產生 5 組不同的初始群組中心點，然後兩組實驗均使用這 5 組初始群組中心點進行實驗，然後計算平均執行結果。

實驗所使用的雲端運算平臺使用 Hadoop 以及 5 臺個人電腦架設成一套小型的 Hadoop 雲端運算環境。其中一臺作為名稱節點以及工作分配節點，其餘 4 臺作為資料節點以及工作節點。每臺電腦硬體配備有 Intel Core 2 Duo E8300 2.83 GHz CPU、2 GB 記憶體、250 GB 硬碟、以及 100 Mbps 乙太網路，所使用的軟體配備為 Ubuntu 14.04 LTS Desktop 作業系統、JDK 1.7、以及 Hadoop 2.6.0。所有實驗所需的資料點與初始群組中心點均使用均勻亂數產生器產生數值範圍介於 0 ~ 1 之間的亂數，所有資料點的維度均為 1，所有實驗的收斂參數均設定為 0.0001。

### 4.1 針對不同資料量的實驗

為瞭解資料點數量對分群效能的影響，本實驗固定分群大小為 1,024 群並分別產生資料量為 40,000、160,000、與 640,000 等 3 種不同數量的資料集。表 1 所列為現有雲端  $k$ -means 分群方法與快速雲端  $k$ -means 分群方法的執行時間與所節省的時間比例。

由表 1 可見，本論文所提方法確實可改善雲端  $k$ -means 分群的運算效能，效能改善比例與資料量成正比。效能改善比例與資料量成正比的主要原因是資料量愈大，單一工作節點所要處理的資料點愈多，雲端運算固定成本所占的比重愈低，所提方法的改善效果愈明顯。其中資料點為 40,000 時，由於

表1 不同分群方法針對不同資料量的處理時間  
(單位:秒)

方法 \ 資料量	40,000	160,000	640,000
現有雲端 k-means 分群法	489.6	281.2	562.1
快速雲端 k-means 分群法	486.3	276.3	550.0
節省計算量 (%)	0.7	1.7	2.2

Hadoop 將工作分配給較少的工作節點執行，所以執行時間反而較長。

4.2 針對不同分群數量的實驗

為瞭解分群數量對分群效能的影響，本實驗將每個資料集的資料量固定為 160,000 並分別為該資料集產生 256、1,024、與 4,096 等 3 種不同分群數目的執行結果。表 2 列出使用現有雲端 k-means 分群方法與快速雲端 k-means 分群方法針對不同分群數目的執行時間。

由表 2 可見本論文所提方法確實可改善雲端資料分群的運算效能，且分群數目愈多效能改善愈大。主要原因是所提方法可加速群組中心點的搜尋速度，群組中心點愈多加速效果愈明顯。其中執行時間隨著分群數目增加而遞減的原因是受到收斂參數與雲端運算環境影響所致。

五、結論與未來研究方向

資料分群是一種簡單有用的資料處理技術，使用雲端運算技術可降低巨量資料的分群時間，但必須使用相當多的運算資源。為降低雲端資料分群的運算資源使用量，本論文嘗試整合現有快速 k-means 分群技術與雲端 k-means 分群方法，開發出一套快速雲端 k-means 資料分群方法，藉以瞭解現有快速 k-means 分群技術是否適合用來改善雲端 k-means 分群方法的運算效能。實驗結果顯示，

表2 不同分群方法針對不同分群數目的處理時間  
(單位:秒)

分群數量方法	256	1,024	4,096
現有雲端 k-means 分群法	1,287.5	279.9	98
快速雲端 k-means 分群法	1,281.2	276.1	94
節省計算量 (%)	0.5	1.3	4.1

所提方法確實可有效改善雲端資料分群方法的執行效能，也證明快速 k-means 分群技術確實可用來改善雲端 k-means 分群方法的運算效能。

未來，我們將應用更多可用的快速 k-means 分群技術到雲端運算環境，設計更全面的實驗，分析各種快速 k-means 分群技術在雲端運算環境下，不同資料集、不同資料量大小、不同分群數目、不同節點數、不同資料維度下、以及不同網路連線速度下的行為表現，進而開發出具高效能表現的快速雲端 k-means 分群方法。

誌謝

本研究感謝南華大學校內專題研究計劃（計劃編號：Y103000953，計劃名稱：低資源需求之巨量資料處理技術開發）的支持。

參考文獻

1. Zikopoulos, P., & Eaton, C. (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. New York: McGraw-Hill.
2. Mayer-Schönberger, V., & Cukier, K. (2013). *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Boston, MA: Houghton Mifflin.
3. Begoli, E., & Horey, J. (2012). Design principles for effective knowledge discovery from big data, *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, Helsinki, Finland.
4. Ahuja, S. P., & Moore, B. (2013). State of big data analysis in the cloud. *Network and Communication Technologies*, 2(1), 62-68.
5. Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3), 645-678.
6. Aggarwal, C. C., & Reddy, C. K. (2013). *Data Clustering: Algorithms and Applications*. London: CRC Press.
7. Jain, K. A. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31, 651-666.
8. Equitz, W. H. (1989). A new vector quantization

- clustering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(10), 1568-1575.
9. Lai, J. Z. C., & Liaw, Y.-C. (2008). Improvement of the  $k$ -means clustering filtering algorithm. *Pattern Recognition*, 41(12), 3677-3681.
  10. Lai, J. Z. C., Huang, T.-J., & Liaw, Y.-C. (2009). A fast  $k$ -means clustering algorithm using cluster center displacement. *Pattern Recognition*, 42(11), 2551-2556.
  11. Bei, C.-D., & Gray, R. (1985). An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communications*, COM-33(10), 1132-1133.
  12. Ra, S.-W., & Kim, J.-K. (1993). Fast mean-distance-ordered partial codebook search algorithm for image vector quantization. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(9), 576-579.
  13. Baek, S.-J., Jeon, B., & Sung, K.-M. (1997). A fast encoding algorithm for vector quantization. *IEEE Signal Processing Letter*, 4(12), 325-327.
  14. Pan, J.-S., Lu, Z.-M., & Sun, S.-H. (2003). An efficient encoding algorithm for vector quantization based on subvector technique. *IEEE Transactions on Image Processing*, 12(3), 265-270.
  15. Lai, J. Z. C., & Liaw, Y. C. (2004). Fast searching algorithm for vector quantization using projection and triangular inequality. *IEEE Transactions on Image Processing*, 13(2), 1554-1558.
  16. Chen, Y.-S., Hung, Y.-P., Yen, T.-F., & Fuh, C.-S. (2007). Fast and versatile algorithm for nearest neighbor search based on a lower bound tree. *Pattern Recognition*, 40(2), 360-375.
  17. Vouk, M. A. (2008). Cloud computing—Issues, research and implementations. *Journal of Computing and Information Technology*, 16(4), 235-246.
  18. The Apache Software Foundation. (2014a). *Apache Hadoop* [Software]. Retrieved <https://hadoop.apache.org>
  19. The Apache Software Foundation. (2014b). *Apache Mahout: Scalable machine learning and data mining* [Software]. Retrieved <http://mahout.apache.org>
  20. Akthar, N., Ahamad, M. V., & Ahmad, S. (2016). MapReduce model of improved K-means clustering algorithm using Hadoop MapReduce, *2nd International Conference on Computational Intelligence & Communication Technology*, Ghaziabad, India.
  21. Wu, K., Zeng, W., Wu, T., & An, Y. (2015). Research and improve on K-means algorithm based on Hadoop, *6th IEEE International Conference on Software Engineering and Service Science*, Beijing, China.
  22. Lin, K., Li, X., Zhang, Z., & Chen, J. (2014). A K-means clustering with optimized initial center based on Hadoop platform, *9th International Conference on Computer Science & Education*, Vancouver, Canada.
  23. Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google file system, *19th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY.
  24. Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters, *6th Conference on Symposium on Operating Systems Design & Implementation*, San Francisco, CA.