結合網頁分類代理人之平行化階層式文件群聚法

曾守正
國立高雄第一科技大學
資訊管理系
imfrank@ccms.nkfust.edu.tw

古惟中
國立高雄第一科技大學
資訊管理系
u9324803@ccms.nkfust.edu.tw

摘　要

面對網際網路上大量的網頁資訊，如何有效組織這些網頁資訊已經成為很重要的課題。本研究結合 WebACE 網頁分類代理人，運用解群技術、平行處理架構，實現「以高頻項目組合為基礎之階層式文件群聚法」(FIHC, Frequent Itemset-based Hierarchical Clustering)，希望能有效提升網頁文件群聚執行效能，以及群聚品質。我們提出將 WebACE 網頁代理人所取得之分類網頁，利用「預先分類資料之解群方法」(Pre-classified De-clustering Method, PCD) 切割成為均等的工作單元，有效達成平行處理節點之間的負載平衡。在我們的平行架構中，各運算節點處理完成後，將運用 FIHC 演算法所產出之群集主題樹 (Topic Tree)，將各節點所回傳之群集結果，利用 XML 檔案合併策略，將多個相同標籤的群集結果進行合併動作。最後，測試結果顯示在平行架構上採用 FIHC 演算法，在群聚品質與執行效能方面都有相當大幅度的提升。

關鍵詞：FIHC 文件群聚、資料解群、平行處理。

1. 簡介

1.1 研究動機與目的

網際網路上大量電子化的文件資訊，已經產生資訊過載的現象。因此，自動化組織文件資訊的需求將會愈來愈重要，Boley *et al.* [2] 的研究提出了WebACE (Web Agent for Document Categorization and Exploration) 網頁代理人，可以針對使用者個人特性 (User Profile) 進行網頁分類的動作。但我們認為透過分類操作的資料並不能完全表現出資料本身的意涵，因此若可以將這些分類資料進行更詳細的分群動作，依照文件的相似特徵進行自動化文件群組的建立，則對於使用者而言，將可以提供更詳盡的文件集展示內容。

因此，本研究針對網頁分類代理人結合文件群聚演算法之加速議題及群集品質加以探討，我們參考文件群聚研究領域常被提及的相關演算法，得知傳統的文件群聚演算法具有下列缺失：

1. 對於高維度特徵向量無法提供好的處理效能，
2. 在處理過程還需要大量儲存空間配合，
3. 針對群集結果無法提供明確的群集描述 (Cluster Description) 等。

對於上述傳統文件群聚演算法可能產生的問題，Fung *et al.*[7] 改良了 HFTC (Hierarchical Frequent Term-based Clustering) 文件群聚演算法 [1]，發展出FIHC (Frequent Itemset-based Hierarchical Clustering) 文件群聚演算法，該演算法經過驗證在執行效能以及群聚品質都有很好的效果。除此之外，FIHC之群聚結果是以主題樹 (Topic Tree) 的方式呈現，並且在樹中的每一個群集都是以高頻項目集 (Frequent Itemsets) 做為群集描述，主題樹的階層關係與群集描述有利於分散式環境下進行結果合併動作處理。

因此，為了加快文件群聚演算法執行速度，減少使用者等待時間，我們希望由 WebACE 網頁代理人取得欲群聚的網頁文件集，接著採用高群聚效果及高效能的FIHC 演算法，配合資料解群 (Data De-Clustering) 概念 [4] 的應用，實現在分散式架構上的快

速處理作業，除了能利用分散式架構加快處理速度以外，還能進一步提升文件群聚品質，所以對於檢索系統使用者來說能獲得更大的幫助。

本研究將以SETI@home研究專案裡，所採用的切割工作單元 (Work Units) 概念為基礎 [10]，建構一個以平行處理架構為基礎之文件群聚系統。而在分散式處理架構中，為了達成參與運算節點間的負載平衡 (Load Balance)，我們會辦法讓每一個節點之FIHC處理程式所回傳之主題樹結構相近，使合併後主題樹與原始單機執行主題樹的群集描述相近，以達到分聚處理之目的。因此，本研究在資料切割方面，以DocId分割方法 [9] 配合PCD解群法，將整個文件集的資料平均分散於切割出來的文件子群中，資料解群方法使文件子群間彼此的相似性高，並且可用以代表文件母群，讓切割出來的工作單元內容相近，確實達到資料平均分配之目的。

最後，我們以WAP (WebACE Project) 測試文件資料集 [2][5] 為文件群聚測試對象，該文件資料集為多個文件群聚技術研究所採用，所以我們認為此測試文件集在文件群聚技術研究上有一定的公信度。我們透過資料切割方法、FIHC演算法、合併策略運用，與平行處理架構等技術之整合應用，經過模擬處理流程，證實我們的方法確實可以有效減少FIHC演算法處理時間，並且在某些參數值條件下多節點執行的群集品質甚至比單節點執行還要好。

1.2 研究貢獻

在過去文件群聚領域的研究中，主要是以提升群聚品質及加快處理速度為目的。在本論文中，我們的方法除了要達成相同目標之外，在結果合併部份還提出了一些不同於過去的新想法。綜合來說，整體的架構與方法有以下特點：

1. 利用簡單的資料配置方法，達到以資料解群方法切割工作單元的效果，這種資料配置方法，簡稱為 PCD (Pre-Classified De-clustering) 解群法，我們將此解群方法應用至文件群聚演算法，使文件群聚演算法適用於平行處理架構，並且經過實驗評估，本方法確實能有助於文件群聚品質之提升。
2. 擴充網頁代理人功能，除了以分類方式向使用者呈現網頁文件集內容之外，本研究所提出的方法，可以加快文件集群聚時間，並且是依照文件內容進行自動化群組的動作，提供使用者更詳盡的網頁文件集內容與呈現方式。
3. 在群集結果合併部份與傳統作法不同，我們不是直接以群與群之間的相似度進行合併，而是提出利用 FIHC 演算法產出群集主題樹的特性作為合併的基礎，省去結果合併過程中計算相似度的比對動作，此種作法將可加快分散式群集合併的動作。

2. 相關研究

2.1 文件群聚技術

文件群聚的目的是希望將所有內容相近的文章集合在一起，然後給予各個群集一個名稱或是摘要，便於使用者瀏覽或查詢。傳統文件群聚演算法是以「階層式群聚法」(Hierarchical Clustering) 與「分割式群聚法」(Partitioned clustering) 為基礎，發展出如：階層式聚合群聚 (Hierarchical Agglomerative Clustering, HAC) 、 K-means 分群法 、 Scatter/Gather文件群集瀏覽系統等。然而這類群聚法在用於文件群聚時，卻無法針對高維度特徵向量以及大量資料提供有效率的群聚效果，而且在完成群聚操作後，無法提供明確的群集描述 [1]。

為了解決上述需求，以改善標準文件群

聚演算法可能遭遇的問題，Beil等人提出兩種以高頻項目詞為群聚基礎的文件群聚演算法，分別是FTC (Frequent Term-based Clustering) 與HFTC (Hierarchical Frequent Term-based Clustering) [1]。隨後Fung等人也以HFTC演算法為基礎發展出FIHC方法[7]，其文件群聚效果經過驗證後證實執行效能都優於上述文件群聚演算法。

就分散式處理的適用性而言，傳統的文件群聚演算法需要進行群與群的相似度比對動作，因此要花費較多的運算資源。此外，我們將FIHC文件群聚演算法與同樣是「以高頻項目詞為基礎之文件群聚演算法」的FTC、HFTC演算法相比較，發現FTC演算法的非結構化群集結果，以及HFTC無法自訂群集數量等特性 [1]，並不適用於分散式處理結果的合併操作。因此，我們希望採用FIHC演算法，並修正配合WebACE網頁代理人以實現平行處理架構上的快速文件群聚處理作業。

2.2 FIHC 方法說明

FIHC 利用資料探勘關聯法則 Apriori 演算法計算出文件集的頻繁特徵詞，這個概念就如同購物籃分析 (Market Basket Analysis) 應用，將每份文件當成一個購物籃，而文件集裡的特徵詞資料，則看成是購物籃中的項目。該演算法裡設定最小全域支持度 (Minimum Global Support, GS) 為門檻移除倒置檔 (Inverted File) 中非高頻特徵詞的部份。透過高頻特徵詞的應用，在 FIHC 文件群聚過程中，可以有效降低特徵詞維度，使它適用於大型與複雜的文件集運算。

此外，FIHC 是以群集為主 (Cluster-Centered) 的群聚演算法，可以透過群集支持度 (Cluster Support, CS) 門檻設定，利用公式計算文件對於群集的歸屬程度，最後FIHC是以主題樹的方式呈現群集結果。運用主題樹作為群集結果呈現的好處是，在主題樹裡的群集是以高頻特徵詞當作群集描述，使用者可以很容易的由高頻項目詞所構成的群集樹進行瀏覽。FIHC演算法包含四個重要步驟，如圖 **1**所示。限於篇幅，其詳細說明請讀者們參考 [7] 的說明。

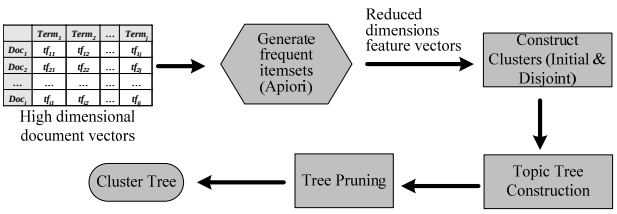


圖 1：FIHC演算法流程 [7]

2.3 文件分配與解群技術

文件經過前置處理、求出關鍵詞，並計算其出現頻率之後，可求得文件倒置檔 (Inverted File)。以倒置檔為主的分割方法主要有兩種：第一種以詞識別符號 (Term Identifier, TermId) 為分割依據，另一種則以文件識別符號 (Document Identifier, DocId) 為分割依據 [9]，DocId分割方法如圖 2所示。

本研究以解群 (De-Clustering) 的原理 [4]，實現DocId分割方法：它會預先將資料切割為內容相近的工作單元，接著分配給各個運算節點，並在各節點的本地端中進行FIHC演算法處理，產生群集主題樹後，回傳遠端發出工作任務之節點。

		$Term_1$	$Term_2$	...	$Term_j$
Node 1	Doc_1	3	3	...	0
	Doc_2	5	2	...	4
	Doc_3	3	7	...	1
Node 2		...	...	...	..
		...	...	...	..
		...	..	...	..
Node 3		...	...	...	..
		...	..	...	..
	Doc_i	3	7	...	1

圖 2：以文件為主的分割 [11]

2.4 WebACE 網頁代理人

WebACE是依照使用者過去所關注的網頁文件資訊建立一個個人化瀏覽資訊。在WebACE架裡使用二種群聚演算法，將使用者瀏覽過的網頁資訊給予群聚處理，產生初始群集。WebACE就可以利用這些群集來產

生新的查詢詞，並且搜尋相似的文件到這些原始群集裡，達到文件分類的效果。限於篇幅，詳細請讀者們參考 [8] 的說明。

2.5 SETI@home 專案

SETI@home主要的研究團隊是美國加州柏克萊大學SETI小組負責，SETI全名為尋找外星智慧計畫(Search for Extraterrestrial Intelligence)。為了要加大運算能力，研究小組發展出SETI@home系統架構 [10]，如圖 3所示，所接收到的電波會被分割成每筆大小為 0.25 MByte的區塊 (亦稱為「工作單元」(Work Unit)，再將工作單元儲存到資料伺服器 (Data Server) 中。資料伺服器利用科學資料庫 (Science Database) 記錄工作單元的處理情況，並利用使用者資料庫 (User Database) 管理全球參與運算的使用者。

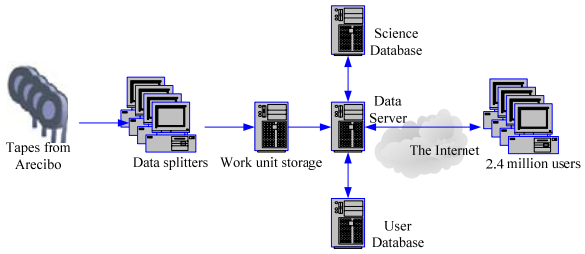


圖 3：SETI@home資料伺服器架構 [10]

3. 系統架構與方法

在本章節我們先將要解決的問題詳細描述，再提出本研究之系統架構、以及執行流程，最後以範例說明完整處理過程。

3.1 問題描述

目前有許多文件群聚演算法被提出，但面對龐大文件資料集，群聚演算法是否能快速得到文件群聚結果，是個有待解決的問題。對於系統加速問題，除了利用超級電腦來執行，採用一般個人電腦所組成的集合也可以達到提升系統效能的效果，本研究在前面章節介紹，可以利用平行處理來分散演算法之處理任務，減少系統執行的等待時間。以下列出幾點是本研究在設計系統架構與方法過程中需要考量到的議題：

(1) 資料分配方面：由於本研究所採用的群聚演算法為 FIHC 演算法，該演算法所產生的群聚結果是以主題樹的方式呈現，如果切割出來的工作單元與工作單元之間資料量或內容差異過大，除了會造成工作負載不均等，而且會讓主題樹節點差異過大，多個子結果集合經過合併動作後，會與單機執行之主題樹結構相似度降低。

(2) 處理時間方面：如果將文件群聚演算法建置於單機系統上，而且要面對大量的文件資料量，從文件前置處理，產生特徵向量，到正式進行文件分群，勢必要花費相當多處理時間，而且過長的等待時間也不符合使用者需求，若能將處理過程中某部份運算分配給其他電腦處理，那可大大降低處理時間。

(3) 在結果合併方面：若是採用一般群與群之間相似度進行合併，在原始群集數量為 m，回收結果的群集量為n的情況下，群集比對次數將會是$m * n$次，群集數量愈多就可能造成比對次數也跟著增加。除了運用群與群之間相似度比對的方法，Deb *et al.* [3] 將 RACHET (Recursive Agglomeration of Clustering Hierarchies by Encircling Tactic) 技術應用在分散式階層文件群集合併策略方面，在合併過程中，要先計算群與群之間的歐幾里德距離 (Euclidean distance)，建立群與群之間的距離矩陣，所以在合併過程中勢必要花費不少時間，因此我們認為結果合併問題，是影響分散式文件群聚技術處理效能上很重要的關鍵。

3.2 系統架構

本研究以平行處理架構為主如圖 4所示，圖中左半部屬於Server端部份，右半部屬於Client端運算節點，我們規劃系統工作執行可分為三個階段：

- **Server 端切割文件集成為被下載的工作單元**:(a)主要是由 WebACE 代理人取得經過簡單分類的網頁文件集、(b) 文件前置處理，取出關鍵詞建立關鍵詞-文件矩陣，以 DocId 方法配合 PCD 方法，將文件集切割為工作單元、(c) 選擇未完的工作單元，將相關資料存放於 Server 端之資料伺服器 (Data Server, DS)。

- **Server 端透過 Data Server 分派工作單元**：(c) Client 端取得 DS 所分派之工作單元、(d)工作單元存放於 Client 的磁碟進行 FIHC 文件群聚、(e) 利用 FIHC 文件群聚演算法產出群集主題樹、(f) 每一個節點將結果交由資料伺服器回傳給 Server 端。

- **Server 端接收回傳給果**：(g) 將主題樹合併，結果顯示至 Server 端之使用者。

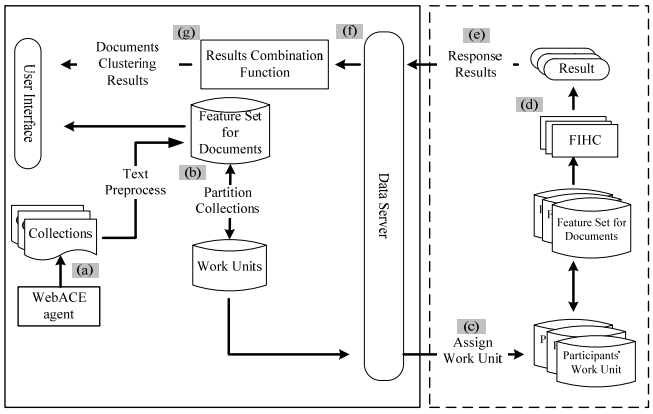


圖 4：系統架構圖

在此，我們假設表 1範例文件集為WebACE代理人所取得之分類文件集，在以下章節中我們將該文件集作為文件群聚目標，該文件集主要分為cisi、cran、med這三個類別，並且已經過前置處理，取出關鍵特徵詞，最後以文件倒置檔的方式呈現。我們利用該文件集展示：工作單元切割法、FIHC處理結果，以及最後結果合併方法。

表 1：範例文件集 [7]

Doc. Name	Feature vector					
	flow,	form,	layer,	patient,	result,	treatment
cisi.1	(0	1	0	0	0	0)
cran.1	(1	1	1	0	0	0)
cran.2	(2	0	1	0	0	0)
cran.3	(2	1	2	0	3	0)
cran.4	(2	0	3	0	0	0)
cran.5	(1	0	2	0	0	0)
med.1	(0	0	0	8	1	2)
med.2	(0	1	0	4	3	1)
med.3	(0	0	0	3	0	2)
med.4	(0	0	0	6	3	3)
med.5	(0	1	0	4	0	0)
med.6	(0	0	0	9	1	1)

3.2.1 工作單元切割法

以下介紹我們所採用的工作單元切割方法：PCD 資料解群法，以及範例執行結果。

3.2.1.1 預先分類資料之解群方法 (PCD)

(1) 方法說明

對於經過網頁代理人 (Web Agent) 分類之網頁資料 [2]，本研究採用簡單的資料分配方法達到切割工作單元的效果，提出解群後的子群集文件是平均由每一個類別之文件集取出的資料所組成，為了方便後續說明，我們將這個方法稱之為「預先分類資料之解群方法」(Pre-classified De-clustering Method, PCD)。在此我們所面對的母群資料是已事先經過分類，即母體文件集$D = \{D_1, D_2, \ldots, D_j\}$，而且母體文件集是由多個類別$C = \{C_1, C_2, \ldots, C_n\}$ 所組成，每一個類別的文件數量$N = \{N_1, N_2, \ldots, N_n\}$，則每一個類別的文件組成為$C_i = \{D_{i,1}, D_{i,2}, \ldots, D_{i,N}\}$，其中$i$是文件集中第$i$個類別，如果要切割出$q$個工作單元，則每一個類別$C_i$要切割出$q$個均等的區塊$P = \{P_{i1}, P_{i2}, \ldots, P_{iq}\}$，類別裡每一個區塊符合 $P_{iq} = \{D_j \mid D_j \in R_i, |D_j| \cong \frac{N_i}{q}\}$，其中$R_i$是$C_i$未被選擇為工作單元的文件子集，而每一個工作單元分別是由各個類別中的區塊中的文件所組成，而且每個區塊之文件數量要相近。

不過這個資料分配方法會因為每一個類別裡的資料量不同，經過資料區塊分割後每個區塊資料量也不一定相同，所以會有負載不平衡的情況產生。例如：有一個擁有四個類別資料集，這四個類別分別為 $A\{a_1\}$、$B\{b_1, b_2, b_3\}$、$C\{c_1, c_2, c_3\}$、$D\{d_1, d_2, d_3, d_4, d_5\}$，將這些資料依類別切割為二個工作單元，則每

個類別會被切割為二個區塊分別是，A_{p1}\{a1\}、B_{p1}\{b_1, b_2\}、B_{p2}\{b_3\}、C_{p1}\{c_1, c_2\}、C_{p2}\{c_3\}、D_{p1}\{d_1, d_2, d_3\}、D_{p2}\{d_4, d_5\}，在這種情況下，類別裡的資料數量若無法被 2 整除，就會造成第二個區塊比第一個區塊資料量還少。同理，在類別 C_i 裡的資料量 N_i 無法被工作單元 q 整除的情況下($N_i \bmod q \neq 0$)，最後一個區塊的數量都是最少的。

如果是以一般Row Order的方式配置類別區塊到各個工作單元裡，如圖 5(a) Raw Order所示，則WU_1 = {A_{p1}, B_{p1}, C_{p1}, D_{p1}}、WU_2 = {B_{p2}, C_{p2}, D_{p2}}，也就是WU_1 = {a_1, b_1, b_2, c_1, c_2, d_1, d_2, d_3}、WU_2 = {b_3, c_3, d_4, d_5}，會造成工作單元資料量有差異，所以我們利用Row-prime Order的方式配置資料 [6]，以減少資料負載差距過大的情況發生。如圖 5 (b) Row-prime Order所示，則WU_1 = {A_{p1}, B_{p2}, C_{p1}, D_{p2}}、WU_2 = {B_{p1}, C_{p2}, D_{p1}}，也就是WU_1 = {a_1, b_3, c_1, c_2, d_4, d_5}、WU_2 = {b_1, b_2, c_3, d_1, d_2, d_3}。

關於每個工作單元之資料量方面，PCD解群法經過Row-prime order資料配置方法的處理，除了第一個或最後一個工作單元的資料數量可能比較少外，每個工作單元的資料數量都是相等的，所以彼此間的差距並不大。

(a) Row Order

Work Unit	A	B	C	D
WU_1	A_{p1}	B_{p1}	C_{p1}	D_{p1}
WU_2		B_{p2}	C_{p2}	D_{p2}

(b) Row-prime Order

Work Unit	A	B	C	D
WU_1	A_{p1}	B_{p2}	C_{p1}	D_{p2}
WU_2		B_{p1}	C_{p2}	D_{p1}

圖 5：資料配置方式比較 [6]

根據上述PCD解群法的說明，我們可以利用圖 6演算法圖示加以說明其切割工作單元的過程，若由網頁代理人取得之原始資料分為四類Class 1~4，欲將原始資料集切割為工作單元Work Unit 1~3，透過簡單的Row-prime order配置方法，即可得到切割結果。另外，我們可以利用圖 7的函式來展式處理方法：輸入參數為文件陣列$D[n]$、欲切割之工作單元數量q，該演算法中以$D[i][j]$代表在文件集裡第i項類別的文件j，文件集共有c個類別，每個類別切割出p個區塊，以Row-prime order將資料平均配置q個工作單元中。

(2) 範例說明

我們以表 1作為範例文件集，展示PCD演算法處理過程，此範例文件集主要分為cisi、cran、med這三個類別，每一個類別所包含的文件集如表 2所示，如果將這個文件集切割為二個工作單元，首先，每個類別要切割出二個文件數量相近的區塊，類別cisi僅有一份文件，所以只有區塊$cisi_{p1}$，類別cran可以切割出二個區塊$cran_{p1}$、$cran_{p2}$，類別med可以切割出二個區塊med_{p1}、med_{p2}，如表 3所示。

接下來，我們利用Row-prime order資料配置方法進行區塊分配的動作，如表 4所示，第一個工作單元WU_1 由文件{cisi.1, cran.4, cran.5, med.1, med.2, med.3}組成，第二個工作單元WU_2 由文件{cran.1, cran.2, cran.3, med.4, med.5, med.6}組成。

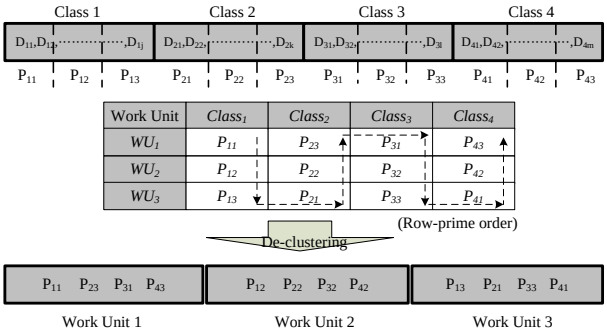


圖 6：預先分類文件之解群演算法圖示

```
/*Input：D[n] –An array of documents.
We use D[i][j] to denote the document j of class i in collections.
    q – Number of Work Unit.
Output：The work units in array Work_Unit[q]. */
Function PCD (D[n] , q){
   Int i , j , q , c , m , x , temp; Int k = 1 , r = 1;
   Int Num_Of_Doc[c] ; String Work_Unit[q];
   x = q;
   for i = 1 to c {
      m = Roundup(Num_Of_Doc[i]/q);
      for p = r to x{
         for j = k to m{
            add D[i][j] to Work_Unit[p];
         }
         k = m + 1;     // The partition (start)
         m = m + m;     // The partition (end)
      }
      temp = r;      // For Row-prime order
      r = x;
      x = temp;
   }
return Work_Unit[q];
}
```

圖 7：PCD 演算法

表 2：預先分類文件集範例

類別名稱	文件集
cisi	cisi.1
cran	cran.1 , cran.2 , cran.3 , cran.4 , cran.5
med	med.1 , med.2 , med.3 , med.4 , med.5 , med.6

表 3：類別區塊切割

類別名稱	Partition 1	Partition 2
cisi	$cisi_{p1}${cisi.1}	
cran	$cran_{p1}${cran.1, cran.2, cran.3}	$cran_{p2}${cran.4, cran.5}
med	med_{p1}{med.1, med.2, med.3}	med_{p2} {med.4, med.5, med.6}

表 4：Row-prime order 資料配置方法

Work Unit	cisi	cran	med
WU_1	$cisi_{p1}$	$cran_{p2}$	med_{p1}
WU_2		$cran_{p1}$	med_{p2}

3.2.2 FIHC

文件集經過解群成為多個工作單元，可以將工作單元分配給平行處理節點，本研究規劃在每一個運算節點上建置FIHC處理程式 [5]，當運算節點取得由部份倒置檔所組成的工作單元後，以這些工作單元作為FIHC處理程式的輸入資料，經過運算最後得到群集主題樹。

主題樹最上層根節點屬於樹的*Level* 0 部份，群集描述標籤為 “null”，所有沒有群集歸屬的文件都是屬於此群集C(*null*)，根節點以下的各層級*Level k*的群集描述標籤是*k-itemset*，*k* >= 1。假設，我們利用網路上的Peer 1 與Peer 2 兩個節點處理之前由PCD解群法所切割完成的工作單元，在GS = 0.3、CS = 0.7、群集數量為 2 的條件下，文件群聚結果如圖 8所示，我們所應用的切割工作單元方法，經過群聚處理所得到 2 棵主題樹，最後一個步驟則要進行樹合併的動作。

主題樹是群集結果概念上的呈現方式，實際上FIHC處理程式是以XML檔案作為群集結果輸出格式，所以平行處理架構上的參與運算節點所回傳的結果實際上是一個XML檔案，如圖 9所示，圖中根元素標籤(Tag) “root” 對應到主題樹中C(*null*) 節點，root是沒有群集歸屬之文件的儲存點，在根元素底下包含了一些子元素，子元素標籤是“cluster”，而子元素的屬性值 “label” 就是群集名稱，主題樹階層的概念可以藉由XML文件表達出來，所以我們可以利用FIHC處理程式輸出的XML檔案做為主題樹合併的基礎。

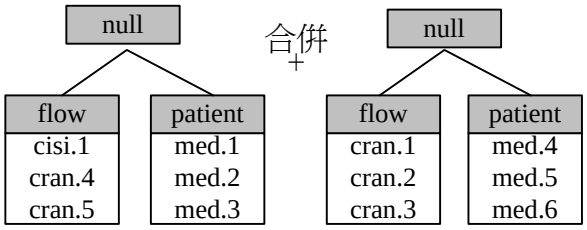


圖 8：產出之主題樹

```
-<root num_clusters="2" label="null"
num_children="2" num_docs="2">
   <documents num_docs="0" />
-<cluster label="Apple" num_children="0"
num_docs="1">
- <documents num_docs="1">
   <document>Doc.2</document>
   </documents>
   </cluster>
-<cluster label="IBM" num_children="0"
num_docs="1">
- <documents num_docs="1">
   <document>Doc.4</document>
   </documents>
   </cluster>
   </root>
```

圖 9：FIHC 輸出結果範例

3.2.3 結果合併策略

(1) 方法說明

若是以 FIHC 演算法所產生的主題樹為合併基礎，合併動作是把在主題樹中同一階層而且是同一個主題名稱的內容進行合併，而不需要將每一個群集進行一對一的比對動作，如此不用處理過於繁雜的比對動作，達到節省系統執行資源的目的。

FIHC 演算法所產生之主題樹的樹高是依照群聚標籤詞的個數決定，例如根節點Level 0，所有沒有群集歸屬的文件都是屬於Level 0 的群集，而在 Level 1 的標籤是一個字詞組成，在 Level 2 是二個詞所組成，Level k 是由 k 個詞所組成，我們可以利用這樣的特性將同一層級裡相同標籤名稱的群集直接進行合併。因此，主題樹進行合併時可能會有三種情況，如下所示：

Condition 1：負責結果合併的運算節點不存在「合併後主題樹」，則新加入的主題樹

直接成為合併後主題樹。

Condition 2：兩棵主題樹的節點，位於相同層級而且擁有相同主題標籤，兩節點所代表的群集可以直接進行合併。

Condition 3：兩棵主題樹的節點，位於相同層級但其中有某些節點之標籤不在合併後主題樹裡，則以新增的方式將節點加入合併後主題樹。

圖 10為主題樹合併範例，假設目前已存在合併後主題樹(a)，希望加入的回收後主題樹(b)，圖中灰色圓形圖是要增加之主題樹與原始合併後主題樹不同的樹節點，實施合併動作時，從Level 0 至Level 3 由上往下掃描，配合先前所提到的三種情況進行節點合併，經過結果合併步驟後，合併後主題樹如圖 11所示。

- Level 0：兩樹之 Node(Null)都是未規類文件群集，屬於 Condition 1 故直接合併。
- Level 1：Node(A)、Node(B)屬於 Condition 1 故直接合併，主題樹(b)之 Node(D)屬於 Condition 2 故以新增的方式將節點加入合併後主題樹。
- Level 2：Node(A,B)、Node(A,D) 屬於 Condition 1 故直接合併，主題樹(b)之 Node(B,D) 、Node(D,G)、 Node(D,H)屬於 Condition 2 故以新增的方式將節點加入合併後主題樹。
- Level 3：主題樹(b)之 Node(A,B,I) 屬於 Condition 2 故以新增的方式將節點加入合併後主題樹。

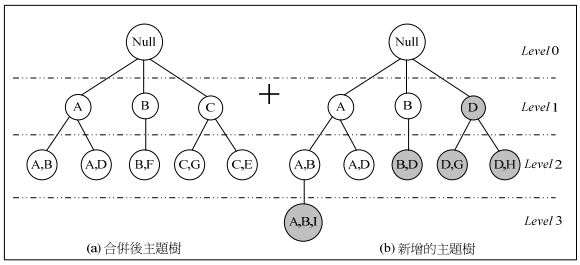


圖 10：主題樹合併

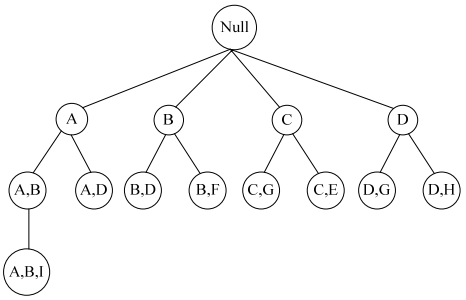
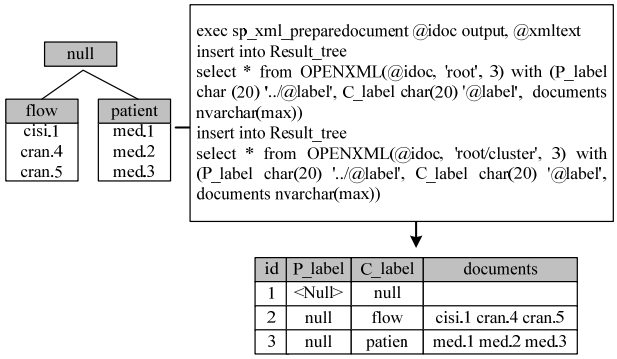


圖 11：合併後主題樹



id	P_label	C_label	documents
1	<Null>	null	
2	null	flow	cisi.1 cran.4 cran.5
3	null	patien	med.1 med.2 med.3

圖 12：利用 OPENXML 函數取出節點元素

以上說明是概念上主題樹合併方法，我們若以FIHC處理程式實際產出之XML檔案作為合併基礎，可以使用Microsoft SQL Server 2005 的SQL Query Analyzer執行OPENXML函數 [13]，剖析XML文件，取出根元素、子元素以及所屬的文件名稱，將XML文件的資料以關聯表的形式存入資料庫，以利後續合併處理相關動作。利用OPENXML函數擷取節點元性之處理範例如圖 12所示。

(2) 範例說明

在此，我們以概念上的主題樹來說明合併結果，如圖 13所示，在本範例中單機執行結果與平行處理架構執行結果相同，不過經過我們的實證評估，多機執行結果與單機執行結果並不一定完全相同，雖然群集結果不同，但在某些參數值設定下，以多機執行之群集品質是較單機群集品質好，關於群集品質評估，我們在下一章節會加以說明。

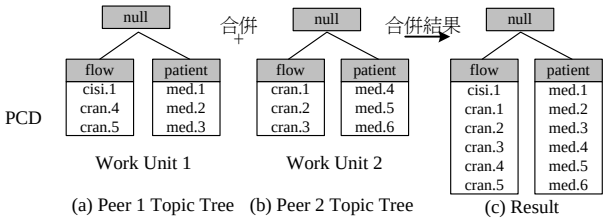


圖 13：範例群集結果比較

4. 實驗結果與討論

4.1 測試資料集

本研究採用的測試文件集為WAP測試文件資料集，該文件資料集是WebACE研究專案 [2] 所產出的分類文件資料，運用此資料集也符合我們在之前章節所提出的PCD解群概念，PCD可以結合網頁代理人之分類功能，將資料進行工作單元切割。

WAP 所包含的文件總數是 1560 份，將全部文件分為 20 類，每個類別的文件數量分別為 5～341 份不等。

4.2 實驗介紹

本研究所提出之系統架構是為了將WebACE代理人所產出之分類網頁資料，利用PCD解群分配法將文件群聚的資料與任務平均分散至平行處理架構裡的各工作站中，以加快FIHC文件群聚演算法之處理速度，並且依照網頁文件內容進行更詳細之文件群聚動作，提升群聚品質。我們將模擬單一主機執行，以及多個運算節點同時執行的條件下，FIHC處理程式所花費的文件群聚時間，如此可以瞭解平行處理加速效果。此外，我們亦探討多節點運算之群集品質，以及工作單元切割方法對於群集結果的影響。對於上述各項評估衡量準則，我們可以觀察FIHC平行處理架構之執行效率 (Efficiency) 與群聚準確性 (Accuracy)，而整體實驗評估項目及內容，如圖 14所示。最後，我們對於各項數據提出一些評估與比較。

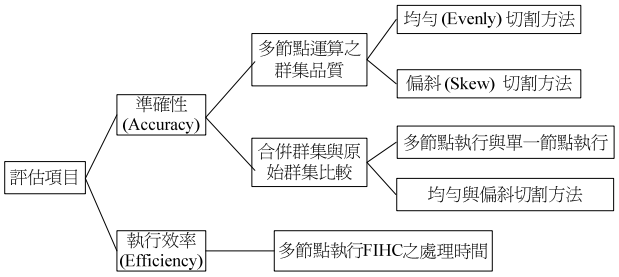


圖 14：實驗評估項目

4.3 數據分析及效能評估

4.3.1 多節點執行群聚準確性評估

(1) 以解群方法切割工作單元之群聚準確性評估

本研究以F-measure評估法 [7][12] 作為文件群聚結果準確性評量之依據，我們比較平行處理架構上與單機執行之FIHC演算法，在各項參數值設定條件下的文件群聚準確性，將每個節點所回傳之群集結果的F-measure值予以平均計算，再與單一節點執行結果進行比較。

首先，我們以GS值做為檢測對象，GS參數值主要是會影響在群聚過程裡高頻項目集合的產生，以及群集解體 (Clusters Disjoint) 公式計算 [7]。在此我們將CS設為 0.65，然後以GS為變數，將GS值設定為 0.15 至 0.4，NC設為 10，在此條件下，測試不同的GS值在平行處理架構上執行FIHC演算法與單機執行FIHC演算法對於群集正確性之影響，如圖 15所示，各種GS值在平行處理架構執行FIHC演算法的情況下，大多數得到的群集準確性是比單機執行還好。

接著，我們以同樣的計算方法，測試不同的CS值在平行處理架構上執行FIHC演算法與單機執行FIHC演算法對於群集正確性之影響，將GS設為 0.25，NC設為 10，在此條件下測試多種CS值 0.3 至 0.7，測試結果如圖 16所示，我們發現單機執行與在平行處理架構執行，群集準確性差別不大。

最後，我們比較不同的群集數量，對於平行處理架構執行與單機執行的影響，不同的群集數量參數設定主要是會影響到FIHC演算法最後主題樹裁切 (Tree Pruning) 步驟

[7]。我們將GS設為 0.25，CS設為 0.35，在此條件下，以較少的群集數量作為輸入參數，在平行處理架構執行之群聚正確性比單機執行要好，如圖 17所示。

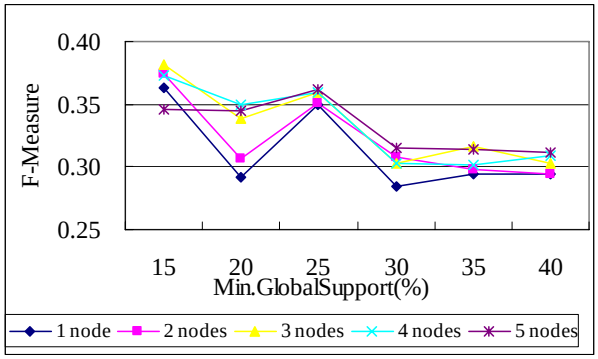


圖 15：F-measure for various GS value by PCD

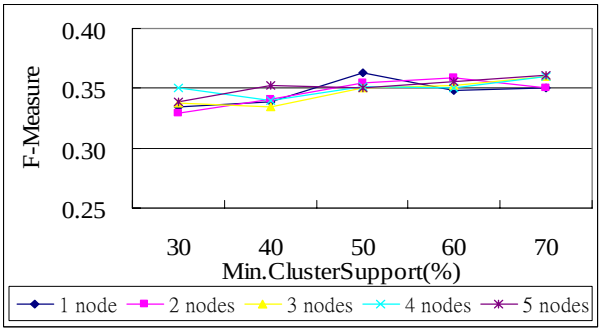


圖 16：F-measure for various CS value by PCD

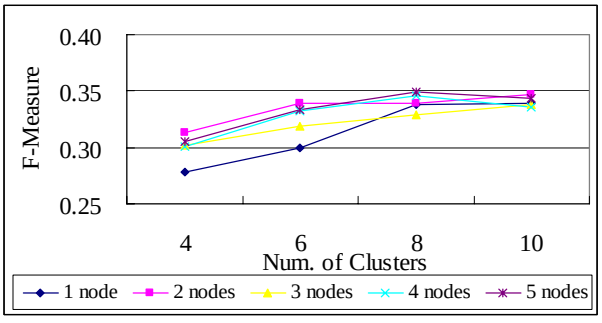


圖 17：F-measure for various NC value by PCD

(2) 以偏斜方法切割工作單元之群聚準確性評估

我們將測試資料集以偏斜方法切割工作單元，測試在多機執行的情況下所獲得之 F-measure 值，測試過程所使用的參數條件，與前述評估解群方法之群聚準確性所設定的條件相同，我們使用各種 GS 值、CS 值、NC 值作為評估準則。我們可以發現以偏斜方法將資料切割為多個工作單元的執行結果，所計算而得的 F-measure 值是優於單一主機執行結果，甚至還優於利用解群方法切割工作單元的結果，主要是因為我們所採用的偏斜切割方法是在已資料分類的情況下進行，是一種較極端的偏斜資料配置方法，主要是用來與均勻的解群方法進行比較，所以在工作單元內的資料本身相似度就比較高，於是經過分散式群聚演算後得到這樣的結果。

但是偏斜資料所組成的工作單元，其每個工作單元所產生的群聚主題樹之間相似度很低，合併群集與原始群集之主題樹組成結構差異很大。在此，我們以 WAP 測試文件集裡的 15 份 Art 類別文件與 33 份 Cable 類別文件為例，說明文件在群集中配置的結果，在 GS=0.25、CS=0.35、NC=6 的條件下執行二個工作單元，並且將群集結果合併，把合併群集與原始群集進行比較，我們可以得知合併群集相似於原始群集的比例僅有 14% (由於篇幅限制，在此將不列出詳細文件在群集中配置之結果)。

因此，若考量合併群集與單機執行的原始群集結果之間相似度問題，偏斜資料分配方法是不適合的，關於本研究驗證主題樹結構相似度的方法，在下一章節合併後群集描述評估會加以說明。

4.3.2 合併後群集描述評估

我們先將這些主題樹之群集描述進行聯集合併的動作，然後將合併後之群集描述集合與單一主機執行之群集描述集合進行比較，確認樹狀結構的相似程度，以下我們進行兩項評估動作：

- 多機與單一主機執行之群集結果比較。
- 解群方法與偏斜配置方法各別取得的主題樹進行合併動作，同樣以 Level 1 之群集描述作為評估依據，比較這兩種資料分配方法對於群集結果的影響。

(1) 多機執行與單一主機執行比較

實證過程中我們發現，在平行處理架構上之FIHC文件群集結果雖然不一定與單一主機執行時相同，但群集品質確有可能會更佳，以表 5為例，在GS=0.25、CS=0.35、NC=6的條件下，利用多節點執行二個工作單元，比在單一主機執行未切割工作單元之文件集多了 3 個群集描述，而且二種群集結果的文件配置也可能不同。

在此，與4.3.1節相同，我們利用WAP測試文件集裡的15份Art類別文件與33份Cable類別文件，加以說明文件在群集中配置的結果，此範例顯示，在相同群集描述標籤的條件下，比較每一個群集內文件配置的情況，得知合併群集相似於原始群集的比例約 54% (由於篇幅限制，在此將不列出詳細文件在群集中配置之結果)，明顯優於偏斜切割工作單元的結果。

表 5：群集描述詞比較

1 node	2 nodes
carolyn, compani, elit, galaxi, gateway, worldwid	carolyn, compani, elit, galaxi, gateway, worldwid, **claud**, **hideo**, **strategist**

無論如何，在本範例中雖然原始群集與合併群集並不完全一樣，不過我們可由圖 17比較該參數條件下之群集F-measure值，發現切割為二個工作單元的群集結果之F-measure值是優於單機執行的群集結果。所以我們可以歸納出下列觀察現象：

- 群集數量及標籤變多

透過工作單元的切割，降低每次群聚處理的資料量，如此一來文件不會因為整個原始文件集裡某些文件所產出的高頻項目組影響，而失去了文件本身應該有的意義。而利用部份文件集作為群聚對象的好處是，除了能加快處理速度，還可以產出更切合某些文件的群集描述標籤。因此，在本研究裡，在子結果合併動作後，所產出之合併群集會有更多的群集標籤。

並且就整體群聚水準來說，這種部份群集結果不同的現象，是我們可以接受的，不過在未來的研究中我們仍希望可以達到分散式處理的目的，就是合併群集與原始群集完全相同的目標。

- 標籤數量控制

但是我們發現切割工作單元的方法，在結果合併後可能會產出太多群集描述標籤，雖然這樣可以用更詳細的方法來組織群集，不過這也有可能會造成群集數量過多的現象，而我們採用的解群切割方法，剛好可以讓合併後主題樹不會過於寬闊，組成結構近似於原始主題樹。因此，本研究所使用的方法，會有折衷的效果，既可以達到精確組織文件群集的目的，又不會讓合併群集數量過多。總之，把合併群集與原始群集進行比較，解群切割方法達到的效果就是，合併群集的標籤與文件配置部份與原始結果相同，部份獨立出來文件群集用更合適的標籤來建構這些新的群集。

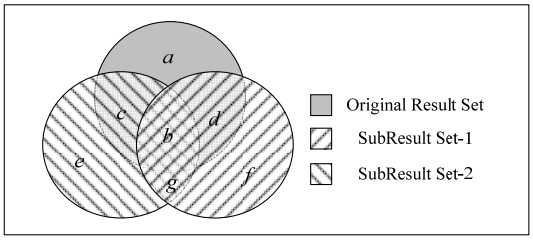


圖 18：群集描述集合關係圖

所以關於單機執行與多節點執行FIHC演算法之群集結果，兩者之間的群集關係可以由圖 18群集描述集合關係圖來說明，單一主機執行之群集描述集合，我們以OriginalResult代表，多節點執行回傳之群集描述集合我們以SubResult代表，兩種群集集合的關係如下：

- *a*：利用多節點執行之群集描述集合，與單一主機執行之原始群集描述集合比較，所缺少的群集描述。
- *b*、*c*、*d*：兩個 SubResult 之群集描述合併後，與單一主機執行之原始群集描述集合比較，相同的群集描述。
- *e*、*f*、*g*：利用多節點執行之群集描述集合，與單一主機執行之原始群集描述集合比較，額外多出來的群集描述。

(2) 解群方法與偏斜配置方法比較

我們可以從實驗結果得知，在相同參數條件下，均勻的解群方法與偏斜方法所產生之合併群集對於原始群集的相似比例，很明

顯的解群切割方法是優於偏斜的切割方法，並且透過主題樹結果呈現，我們發現若合併群集之群集描述與原始群集之群集描述相同的部份愈多，合併群集的文件配置的情況與原始群集相同的機會愈高，所以在本研究裡，我們可以利用群集描述來估算合併群集與原始群集相似度。

一般常用的正確率估算方法有二種，一種是正確率 (Precision)、另一種是查全率(Recall)，如圖 19所示，在此，Precision值愈高代表合併群集與原始群集描述相同的數量愈多，Recall值愈高代表合併群集與原始群集主題樹結構愈相近，計算公式如下所示：

$$precision = \frac{A}{A+B} \quad , \quad recall = \frac{A}{A+C}$$

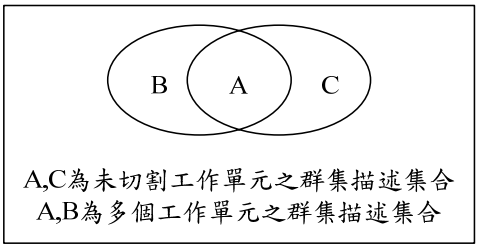


圖 19：評估合併群集與原將群集相似度之方法

在此，我們分別以直方圖代表Precision測試，折線圖代表Recall測試，兩種工作單元切割方法比較，如圖 20所示，採用解群方法之群集描述Precision都有達到 60%以上，而採用偏斜方法切割工作單元群集描述Precison最高僅到達 40%。另外，以解群方法切割工作單元之群集描述Recall可以高達95%以上，而採用偏斜方法切割工作單元之群集描述查全率皆在 85%以下，所以無論是Precision或Recall，在合併後主題樹群集描述與原始群集結果之間比較，以解群方法切割工作單元都是優於偏斜的資料切割方法。因此，我們可以瞭解以解群方法切割工作單元可以有效達到與原始群集結果相近的目的。

4.3.3 執行效能評估

我們將FIHC演算法建置於硬體環境，並設定演算法參數為GS=0.05、CS=0.25、NC=5，分別以單機執行與平行處理架構 2~5個節點同時執行，測試結果如圖 21所示，將工作單元切割為 2~5 個，以 2~5 個運算節點同時執行的方式執行FIHC演算法。就執行效率而言，多運算節點架構之FIHC演算法是優於單機執行之FIHC演算法，就整體來說，平行處理架構之FIHC演算法，隨著工作單元數量增加，參與運算的節點增加，執行效率明顯提升。

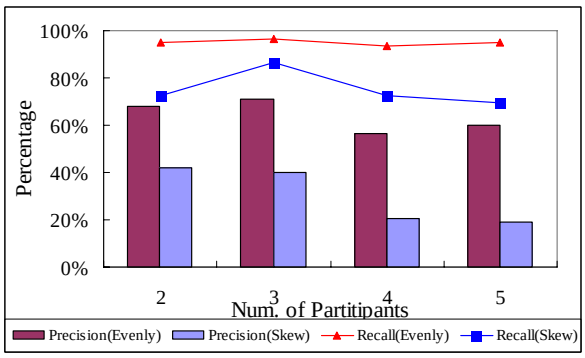


圖 20：解群方法與偏斜分配方法之比較

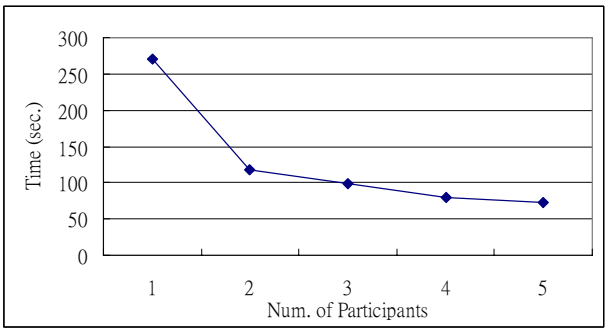


圖 21：以平行處理架構執行 FIHC 效能比較

5. 結論與建議

本研究結合 WebACE 網頁代理人所產生之預先分類文件資料，我們提出將解群方法應用至 FIHC 文件群聚演算法，藉由 PCD 解群法的應用實現切割工作單元的概念，使文件群聚技術適用於平行處理架構。並且在分散式運算環境中，子結果群集合併動作是影響整體處理效能的關鍵，所以本研究提出以群集描述為基礎的結果合併策略。最後我們將本研究所提出之系統架構與方法予以模擬測試，由實證評估結果得知，本研究所提出之系統架構與方法，在執行效率方面，多運算節點架構之 FIHC 演算法明顯優於單機作業，而在各種常用的參數值設定下，多運算節點群聚結果比單機執行 F-measure 值高，

也就是多機執行之群集品質較單機執行為佳，若進一步瞭解 FIHC 參數設定對於多節點執行 F-measure 值的影響，不同的「最小全域支持度 (Minimum Global Support)」以及「群集數量」這兩種參數設定對於多節點執行影響較大。除了上述 F-measure 值與執行效能的測試，本研究再將屬於平均分配 (Evenly) 的解群法與偏斜 (Skew) 分配方法進行比較，以資訊檢索領域常用的評估準則 Precision、Recall 加以測試，我們可以驗證解群方法確實可以加強合併群集與單機執行的原始群集結果之間的相似度。

利用分散處理的目的，就是把一個工作分配到多個主機執行，讓多機執行的結果取代原始結果，不過在實驗過程中，我們發現分散式架構之 FIHC 演算法所產出的群集結果與單一主機執行的合併群集結果，群集內文件配置情況並不一定完全相同，就我們在論文中所提出的理論來說，我們可以大膽假設，若文件資料量愈龐大，每個工作單元的特徵會更接近母體，兩種群集結果應該會愈相近，不過值得注意的是，本研究所提出之方法是一個創新的嘗試，關於群集結果差異的部份，我們認為這是在未來可以深入研究，還有許多值得改進的地方。

參考文獻

[1] Beil, F., Ester, M., and Xu, X., "Frequent Term-Based Text Clustering," *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 436 – 442.

[2] Boley, D., Gini, M., Gross, R., Han, E. H.(Sam), Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J., "Document Categorization and Query Generation on the World Wide Web Using WebACE," *Journal of Artificial Intelligence Review*, Vol. 13, No. 5-6, 1999, pp. 365-391.

[3] Deb, D., Fuad, M. M., and Angryk R. A., "Distributed Hierarchical Document Clustering", *IASTED Conference on Advances in Computer Science and Technology*, Puerto Vallarta, Mexico, January 23-25, 2006, pp.328-333.

[4] Fang, M. T., Lee, R. C. T., and Chang, C. C., "The Idea of De-Clustering and its Applications", *Proceedings of 12th International Conference on Very Large Data Bases —VLDB'86*, (Kyoto, Japan). Aug. 1986, pp.181-188.

[5] FIHC Toolkit. http://www.cs.sfu.ca/~ddm/dmsoft/Clustering/fihc_index.html.

[6] Frank S.C. Tseng and Pey-Yin Chen, "Parallel Association Rule Mining by Data De-Clustering to Support Grid Computing," *The 9th Pacific Asia Conference on Information Systems: PACIS 2005*, Bangkok, Thailand, July 7-10, 2005, pp. 1071-1084.

[7] Fung, B. C. M., Wang, K., and Ester, M., "Hierarchical Document Clustering Using Frequent Itemsets," *Proceedings of the SIAM International Conference on Data Mining (SDM '03)*, 2003, pp. 59-70.

[8] Han E.-H., Boly D., Gini M., Gross R., Hastings K., Karypis, "WebACE: A Web Agent for Document Cateorization and Exploration," *Proceedings of Agents 98*, 1998.

[9] Jeong, B. S., and Omiecinski, E., "Inverted file partitioning schemes in multiple disk systems," *IEEE Transactions On Parallel And Distributed Systems*, Vol. 6, No. 2, February 1995, pp.142-153.

[10] Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Leboisky, M., "SETI@home-Massively Distributed Computing for SETI," *IEEE Computational Science and Engineering*, Vol. 3, No. 1, Jan-Feb, 2001, pp.78-83.

[11] Macfarlane, A., "Distributed Inverted Files And Performance : A Study Of Parallelism And Data Distribution Method In IR," Department of Information Science, City University, London, A thesis for the degree of Doctor of Philosophy, August 2000.

[12] Steinbach, M., Karypis, G., and Kumar, V., "A Comparison of Document Clustering Techniques," *KDD Workshop on Text Mining*, (Boston, MA, USA). August 20-23, 2000.

[13] 曾守正、周韻寰，「資料庫系統進階實務」台北：華泰書局，2003。