格網運算環境於序列型樣探勘之設計與實作

羅宇傑
樹德科技大學資訊管理學系

吳志宏
高雄大學電機工程學系

賴智錦
高雄大學電機工程學系

摘要

本論文提出格網運算環境於序列型樣探勘之設計與實作。本研究實作一Apriori-like演算法的序列型樣探勘於格網運算環境，並加以驗證、分析其探勘效能與結果。Apriori-like演算法相較於相關序列型樣探勘的演算法而言，探勘過程需歷經大量重覆性與遞迴式的資料處理與演算，缺乏高效率的執行效能。但Apriori-like演算法透過修改少量的資料探勘演算程序，即可適用於鬆散耦合的分散式處理，並實行分散任務於格網運算環境。本研究所提出的格網運算環境中，設計了運算格網與資料格網等兩種格網節點型態，所有的格網節點皆以Globus Toolkit實作，每一格網節點安裝與設定本研究所開發的分散探勘程式。格網服務程序為透過使用者或遠端格網節點所觸發之程序，並賦予回應探勘結果至格網主控端，相互合作地完成探勘任務。格網運算環境主要分散於兩個不同的大學校園網路，安裝與設定了16台格網節點，每一格網節點為獨立電腦主機，每台電腦皆配置著不同的硬體元件，藉以呈現真實格網運算的實作環境。最後，經由本研究之實驗結果與效能評估顯示，格網運算環境可提供高度彈性與高效能之運算平台，適用於大容量資料庫的序列型樣探勘。

關鍵字：資料探勘、序列型樣探勘、分散式處理、鬆散耦合處理、格網運算

本研究承蒙行政院國家科學委員會研究經費補助（計畫編號：NSC 94-2213-E-390-00、NSC 95-2221-E-390-023與NSC 96-2221-E-390-034-MY2）特此致謝。

The Design and Implementation of a Grid-Computing Environment for Mining Sequential Patterns

Yu-Chieh Lo
Department of Information Management, Shu-Te University

Chih-Hung Wu
Department of Electrical Engineering, National University of Kaohsiung

Chih-Chin Lai
Department of Electrical Engineering, National University of Kaohsiung

Abstract

This paper presents the design and implementation of a grid-computing environment for mining sequential patterns. An Apriori-like algorithm for mining sequential patterns is deployed in the proposed grid-computing environment. Apriori-like algorithm is not of very high performance in comparison to others but it is more convenient to be realized for distributed processing in a grid computing environment due to its nature of loosely coupled processing. Two types of grids are designed, the computing grid and data grid, in the proposed environment. All grid nodes are installed with full functions implementing the mentioned Apriori-like algorithm for mining sequential patterns, each of which is wrapped by Globus Toolkit. Grid services are invoked by the users or other grids and able to respond to the invoking side for cooperatively completing the mining task. There are 16 computers serving as grid nodes each of which is equipped with different hardware components and is distributed across two WANs. The experimental results show that the proposed grid-computing environment provides a flexible and efficient platform for mining sequential patterns from large datasets.

Key words : Data Mining, Mining Sequential Patterns, Distributed Processing, Loosely Coupled Parallelism, Grid Computing

壹、緒論

近年來，資料探勘(Chen et al. 1996；Frawley et al. 1992)逐漸從學術研究轉移至產業界應用，例如：顧客關係管理、網路行銷、網路入侵行為分析與網路學習等。資料探勘的目的，是從大量資料庫中的資料，找尋隱含於資料中，具有效用的規則、模式與知識。整個探勘過程又稱為資料庫中的知識發掘(Knowledge Discovery in Database，KDD)。進行資料探勘，需不斷地存取資料庫，並進行探勘演算法的運算。在此迭代式處理的過程中，原始資料庫需重覆的經過選擇、清除與歸類等計算，搭配探勘演算法參數的調整，直到滿足設定條件，取得探勘結果。因此，假使資料探勘應用於大容量資料庫，探勘處理的過程將明顯地費時，易引發運算伺服主機極大資源負載量，大量耗費整體探勘任務的執行時間。

根據文獻，有兩種主要提升資料探勘效能的方法。第一種方法為利用新興或不同的資料結構，藉以加速資料探勘的演算程序(Masseglia et al. 1998；Han et al. 2004；Han et al. 2000；Srikant & Agrawal 1996)；另一種方法則利用平行化(Parallel)或分散式(Distributed)的技術，拆解探勘任務為數個子任務，以同步或非同步的方式運作，藉此改善探勘效能(Roughan & Zhang. 2006；Rahman et al. 2005；Natarajan et al. 2004)。前者需不斷地測試新方法的穩定性與延展性；後者則需有效地切割資料或任務，才得以適用平行化或分散式程序。

近年來，格網運算(Grid Computing)已迅速發展於大容量的平行化與分散式運算等領域(Ali et al. 2005；Chervenak et al. 2000；Cao et al. 2002；Cannataro & Comito 2003；Alpdemir et al. 2003；Swany & Wolski 2004；Rahman et al. 2005)。主要的目標為提供成本低廉、整合性基礎建設與協同資源分享等網路服務，以進行分散式處理。

本研究分析相關序列型樣探勘的演算法，發現傳統GSP(Generalized Sequenctial Patterns)演算法的資料處理程序可經由小部份修改後，即可將探勘任務劃分成數個獨立性的子探勘任務，並適用於實行分散式的鬆散耦合處理。在格網運算環境中，可透過建置多個獨立性的運算節點，形成一分散式的鬆散耦合處理架構。因此，格網運算環境將可作為傳統GSP演算法進一步提升探勘效能的基礎。

貳、文獻探討

一、序列型樣探勘

序列型樣探勘(Mining Sequential Patterns, MSP) (Agrawal & Srkiant 1995)，屬於資料探勘的重要議題。其目的為經由序列型態資料庫，探索頻繁性序列。最典型的應用範例，是從銷售賣場的大量交易記錄，依所有顧客的購買行為，尋找顧客潛在的購買商品序列。

舉例來說，一個交易序列資料庫，主要以多個顧客交易資料所構成。每一交易包含顧客編號、交易時間與購買項目等欄位。每個顧客在相同的交易時間點，只有一筆交易資訊。每筆交易不考慮購買項目的數量，僅考慮每個項目的購買與否。

項目集合(Itemset)是一群項目(Items)所構成的非空集合，項目集合 I 可表示為$\{I_1, I_2, ..., I_m\}$，其中$I_j(1\leq j\leq m)$是一個項目。一個序列(Sequence)是一群項目集合(Itemsets)的有序串列，序列s可表示為$<s_1, s_2, ..., s_n>$，其中$s_j(1\leq j\leq n)$是一個項目集合。序列長度(Length of Sequence)是指構成序列的項目集合的個數。假如一個序列$s'=<a_1, a_2, ..., a_n>$被包含於另一序列$s=<b_1, b_2, ..., b_m>$，並存在整數$i_1<i_2<\cdots<i_n$，且滿足$a_1\subseteq b_{i1}$、$a_2\subseteq b_{i2}$、…、$a_n\subseteq b_{in}$等條件；則稱序列s'為序列s的子序列(Subsequence)。若序列s不被包含於其它序列，則稱此序列s為最大序列(Maximal Sequence)。

每一顧客的每個交易由一群項目所構成，依交易時間排序後，可形成一顧客序列(Customer Sequence)。顧客序列依交易時間順序，表示為$T_1, T_2, ..., T_n$，$T_i(1\leq i\leq n)$中包含的項目表示為itemset(T_i)。顧客序列則可表示為序列{itemset(T_1), itemset(T_2), ..., itemset(T_n)}。一個序列s的支持度(Support)定義為“包含序列s的顧客序列總數佔全部顧客總數的比例”。假如序列s被包含於一個顧客序列中，則此顧客支持序列s。假定一個顧客序列資料庫，序列型樣探勘的問題為根據使用者定義的最小支持度(Minimum Support)，搜尋所有序列屬於最大序列的結果。每一最大序列即為序列型樣(Sequential Pattern)。滿足最小支持度限制的序列，則稱為大序列(Large Sequence)或頻繁序列(Frequent Sequence)。

Agrawal與Srikant (1995)最早提出序列型樣探勘的概念與AprioriAll演算法，主要任務是透過大量資料庫，找出隨著時間變動且具備特定順序，屬於經常性發生的序列型樣。AprioriAll演算法的作法是一種基於「產生——測試」(generate-and-test)的程序。先從序列型態資料庫找出長度為k-1($k\geq 2$)的頻繁序列(LS_k-1)，再以長度k-1的頻繁序列中的元素以組合(combination)的方式產生長度為k的頻繁序列(LS_k)。接著測試LS_k中各序列型樣是否為頻繁序列？所有通過檢查的頻繁序列再依此過程產生長度為k+1的頻繁序列(LS_k+1)，直到無法再進一步產生頻繁序列為止。Srikant與Agrawal再於1996年提出GSP演算法(Srikant & Agrawal 1996)，以Apriori-like演算法為基準，進一步加入探勘期間(Duration)、事件滑動視窗(Event Sliding Window)及事件間隔時間(Event Interval)三個參數，實行一般化的序列型樣探勘。然而，以基於Apriori-like的GSP演算法進行序列型樣探勘，主要隱含了三大主要缺點：(1)產生大量候選序列；(2)多次掃描資料庫；(3)佔用大量運算與儲存資源。

Masseglia等學者們於1998年提出PSP演算法(Masseglia et al. 1998)。PSP演算法的主要優點，一方面利用Prefix樹狀結構加以儲存資料序列，有效地節省記憶儲存空間。另一方面，亦可刪減非頻繁性序列資料的分支，使其節點不再繼續產生分支，減少候選序列的運算成本。

Han等學者們於2000年提出了FreeSpan(Frequent Pattern-Projected Sequential Pattern Mining)演算法(Han et al. 2000)。FreeSpan與Apriori-like演算法的相較之下，最大優點為將搜尋空間限制在較小範圍的序列資料庫，另外，持續增長與產生的候選序列也隨之減少。但是，FreeSpan演算法依然有如下兩個缺點：(1)重覆的Projected Database；(2)產生過多的候選序列。

Han等學者們再於2004年提出了PrefixSpan(Prefix-Projected Sequential Pattern Mining)演算法(Han et al. 2004)。PrefixSpan演算法相較於GSP演算法與FreeSpan演算法而言，PrefixSpan演算法的探勘效率最佳。主要的優點如下三點所示：(1)不需產生候選序列；(2)Projected Database的資料量持續遞減；(3)主要的成本在於建立Projected Database。

隨著序列型態資料庫的大小不斷地增加，上述各種探勘演算法將遇到相當程度的挑戰。為了能夠加速探勘的效能，國外有學者利用多部電腦進行所謂的分散式探勘相繼被提出(Agrawal & Shafer 1996；Zaki 1999；Gualnik & Karypis 2004)。國內則有張昭憲與周定賢等學者們(2005)，為了加速大型資料庫的序列型樣探勘，以PrefixSpan演算法為基礎，提出以分散式架構為基礎的探勘演算法，並據以發展實用的探勘系統。

二、分散式處理

隨著資料庫愈來愈龐大，資料儲存量與運算量也趨於暴增，獨立性主機已不足以處理更龐大的運算程序。分散式系統可依資源共享機制，區分成兩種不同的架構，如緊密耦合處理(Tightly Coupled Processing)與鬆散耦合處理(Loosely Coupled Processing)(Silberschatz & Galvin 2003)。緊密耦合處理的系統架構下，主要將兩顆以上的處理器(Processor，如CPU)集結於同一台主機，共享記憶體資源。多處理器系統可將許多程序或一個程序的許多子程序配置到不同處理器，平行化執行任務。所謂的鬆散耦合處理系統架構，則是以多台主機或運算節點聯合進行計算工作。各個節點間並不共用處理器與記憶體，各自擁有獨立性區域處理器與記憶體，依靠通訊網路的訊息交換機制，互相溝通與執行任務。在鬆散耦合處理的架構下，每個節點的軟硬體元件與資料處理特性皆有所不同。

三、格網運算

（一）格網運算環境與基本架構

格網運算(Grid Computing)是一具備分散式運算、網路服務環境、大容量儲存體與其他運算資源的整合性系統(Grimshaw et al. 1994；Erwin & Snelling 2001；Foster & Kesselmany 1997)，主要目的為建立一虛擬化超級電腦型態的分散式運算環境，提供解決大容量與複雜性運算等問題。一個格網運算環境整合眾多格網節點於一體，以網路服務(Web Service)進行互動式的資料轉移、運算與處理。一格網節點可為個人或組織的資源集合體，此資源集合包含了處理器、記憶體、資料庫或其他軟硬體資源等。每一格網節點屬於獨立基礎架構，提供閒置性軟硬體元件資源，並兼具高度彈性與安全性地存取其他節點資源。

一般說來，建立格網運算環境除了許多計算資源外，還需要管理與安全性的機制，才能有效地的進行協同計算。每一個計算或管理資源的機器，都稱為一個格網節點(Grid Node)，依照其角色分成運算格網(Computing Grid)與資料格網(Data Grid)。前者主要作用於運算服務，後者則提供資料存取服務。目前，格網運算核心使用開放性格網服務架構

(Open Grid Services Architecture，OGSA)標準準則(Foster et al. 2002)。OGSA以跨平台的形式，管理分散式資源，提供高標準的網路傳輸服務品質，亦定義了開放式的介面與標準，結合異質性資源，建立運算、儲存與協同合作等處理機制，強化資料通訊與溝通服務。

（二）格網運算相關文獻

近年來，許多研究學者相繼提出以格網運算解決各類型大量運算與資料處理問題。主要應用包含了實行大量資料探勘、資料與知識格網、分散式服務探索、資源負載管理與分散式代理人等。以下則根據格網運算的相關文獻，加以分析與探討問題解決模式。

Cao et al. (2002)提出代理人(Agent-based)型態的格網運算資源管理系統。此系統利用PACE套件的效能預測技術，將大量執行於區域性的格網資源資訊，以量化資料型態加以表示，評估整體執行任務之成本問題。Cannataro與Talia (2003)提出知識格網的分散式知識探索架構。知識格網主要實作於PDKD(Parallel and Distributed Knowledge Discovery)系統，包含了大量的運算格網，提供可靠性地存取高效能的運算資源。Alpdemir et al. (2003)提出服務(Service-based)型態的分散式格網查詢機制，主要目的為執行運算與資料查詢管理作業。以基於服務型態的格網運算，設計與實作一分散式查詢處理系統，提供資料儲存體與分析資源的服務。Natarajan et al. (2004)提出格網(Grid-based)的企業化資料探勘應用的概念。透過分散式資料庫技術，將大量資料儲存於多個關聯式資料庫系統，結合高效能的運算節點與低成本的網路通訊，加速資料探勘效能。

參、研究設計與方法

一、基於鬆散耦合處理的序列型樣探勘分析

傳統的GSP演算法是屬於迭代式的處理，不斷地重複地產生與測試候選序列，進而尋找頻繁序列。本文的MSP則是GSP的擴充，會求得最大頻繁循序型樣，可將其過程分成排序、大項目集合、轉換、序列與最大序列等5個階段。每一階段執行的複雜度不同，甚至具有遞迴性的任務。從各階段的資料相依性來分析，可分成兩類：

1. 處理程序內的相依性(intra-process dependency)：指某個處理程序在該階段對資料集進行處理時，如果可以將輸入資料集分割成數個小集合，輸出結果為所有小集合輸出的聯集，資料庫的分割方式可以各自獨立。
2. 處理程序間的相依性(inter-process dependency)：指某一處理程序在接收前一階段輸入的資料後，便可獨立進行處理，執行過程中並不需要其他的資料輸入。

GSP演算法的五個獨立性任務中，任務與任務之間存在著順序性的處理原則，屬於處理程序間的相依性。而每個任務之中，則可隨著劃分資料庫與任務的處理原則，形成處理程序內的相依性。因此，GSP演算法依據這兩種資料處理相依性、搭配修改少量的探勘演算核心，即可轉換成鬆散耦合型態的分散式處理，並適用於格網運算環境實行序列型樣探勘。圖1為基於鬆散耦合處理的GSP流程。

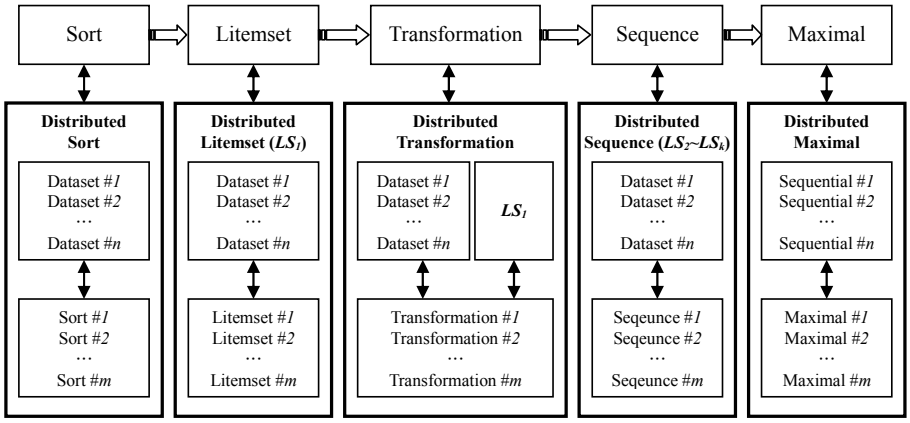


圖1：基於鬆散耦合處理的GSP流程

本研究依據分割式(Partition-based)為基礎，將序列資料庫及探勘任務劃分成數個等分，並透過格網節點部署的機制，實行分散處理的序列型樣探勘，概念如下三點所示：

1. 任務分割(Task-Partition)：依據分散式處理的原則，將傳統GSP演算法中的工作，分成如下五個任務：
   (1)分散式排序T_{ST}：以顧客編號為主要鍵值，升冪排序，再以交易時間為次要鍵值，升冪排序，形成排序後之顧客交易資料庫。
   (2)分散式大項目集合T_{LS}：尋找所有長度1的大項目集合，分別指定一整數值代表。
   (3)分散式轉換T_{TM}：依據所有顧客序列資料為基礎，刪除非大項目集合的資料，將序列資料轉換成整數值之轉換後顧客序列資料庫。
   (4)分散式序列T_{SQ}：依轉換後顧客序列資料庫，及所有長度1的大序列為基礎，依序找出所有LS_2～LS_k的大序列。
   (5)分散式最大序列T_{MX}：以所有LS_2～LS_k的大序列為基礎，檢驗與刪除所有具備子序列特性的序列資料，找出所有LS_2～LS_k的最大序列，重新對應至原始大項目集合。
2. 資料分割(Data-Partition)：輸入與輸出資料會隨著格網節點不同的規畫而不同。基本上，可以將每一資料集$D=\{d_1, d_2, ..., d_n\}$，依需求分割成數個子集合，其中每一等分的大小為使用者指派或等分切割成相同大小，例如：等分切割成α個子集合，$D=D_o \cup D_1 \cup \cdots \cup D_{\alpha-1}$，其中為每一子集合元素個數的下限。

$$D_i = \begin{cases} \{d_{i\times m+1}, d_{i\times m+2}, \ldots, d_{(i+1)\times m}\}, & 0 \le i \le \alpha-2 \\ D - \bigcup_{0\le j\le \alpha-2} D_j, & i = \alpha-1 \end{cases}$$

$m = \left\lfloor \frac{|D|}{\alpha} \right\rfloor$ 為每一子集合元素個數的下限。

3. 格網節點部署：格網節點(Grid Node，*GN*)可分成運算格網節點(Computing Grid Node，*CG*)與資料格網節點(Data Grid Node，*DG*)兩大類。*CG*主要負責執行大量計算工作，*DG*主要負責儲存大量資料。每一個格網節點*GN*具下列的識別資料：

$GN=<ID, IP, CPU_{Type}, CPU_{Speed}, RAM_{Size}, HD_{Capacity}, LAN_{Bandwidth}, Type>$

其中*ID*表示*GN*的編號，*IP*表示*GN*的網路位址，CPU_{Type}表示*GN*的處理器等級，CPU_{Speed}表示*GN*的處理器核心運算速率，RAM_{Size}表示*GN*的記憶體容量大小，$HD_{Capacity}$表示*GN*的硬碟容量大小，$LAN_{Bandwidth}$表示*GN*的網路卡傳輸速率，*Type*表示*GN*的主機型態與網路應用服務。每一*GN*可能扮演*CG*或*DG*，依據其規格由使用者指定，有可能同時擔任*CG*與*DG*。在執行工作方面，由使用者指派其執行的任務。

二、問題探討與設計

為了達成前述分散式序列型樣探勘的目的，本研究根據格網運算的任務切割，劃分資料庫與探勘任務的內容，並區分下列五種型態的模組設計：

（一）運算工作規劃(Job Configuration)

其目的主要是依據序列型樣探勘的資料輸入量與目前上線的格網運算，規劃可使用的*CG*與*DG*資源。格網主控端(Grid Console)依據序列型樣探勘中的工作量，將任務*T*與資料*D*切割為合適的小單元，並傳送至格網運算環境中，執行探勘任務的運算。

（二）工作分派器(Job Dispatcher)

依據序列型樣探勘的任務與資料分割，分派到不同的格網節點上，如下三種模式：

1. 單一任務搭配完整資料庫於單一格網節點(Single Task on Single Grid-Node with Complete Dataset，*STSNCD*)：以單一運算格網為基礎，讓特定性運算格網依據完整資料庫的資料格網，執行特定性探勘任務。可適用於T_{ST}、T_{LS}、T_{TM}、T_{SQ}與T_{MX}的探勘任務。
2. 多個任務搭配完整資料庫於多個格網節點(Multiple Tasks on Multiple Grid-Nodes with Complete Dataset，*MTMNCD*)：以多個運算格網為基礎，將特定性探勘任務劃分成數個子探勘任務，並依據完整資料庫的多個資料格網，分散地執行於多個運算格網。可適用於T_{LS}、T_{SQ}與T_{MX}的探勘任務。
3. 多個任務搭配部份資料庫於多個格網節點(Multiple Tasks on Multiple Grid-Nodes with Partitioned Dataset，*MTMNPD*)：以多個運算格網為基礎，搭配多個不同部份資料庫的多個資料格網，執行多個探勘任務。可適用於T_{ST}、T_{TM}與T_{SQ}的探勘任務。

（三）工作進度監視器(Job Progress Monitor)

本研究將透過格網系統介面，持續追蹤分散式序列型樣探勘T_{ST}、T_{LS}、T_{TM}、T_{SQ}與T_{MX}等五個任務的執行進度與狀態，並擷取探勘任務的結果。

（四）任務溝通語言(Task Communication Language)

在格網運算環境中，格網主控端主要透過任務溝通語言，分派運算任務與資料集至被選取的所有格網節點，並於運算任務完成之時，回應適當的探勘任務結果。

（五）資料傳輸(Data Transmission)

在格網運算環境中，格網主控端透過任務溝通語言，與所有的格網節點進行資料傳輸。資料傳輸模式屬於遠端程序呼叫(Remote Procedure Call，RPC)。本研究以Java程式語言撰寫格網服務的JAX-RPC程序(JSR-000101: Java APIs for XML based RPC)，並裝載JAX-RPC程序於所有的遠端格網節點。格網主控端與遠端格網節點之間，則透過參數傳遞的形式，進行資料接收、運算作業與回應探勘任務結果。

肆、系統設計與實作

為了驗證研究設計與方法的可行性，本研究將透過格網環境的系統設計、中介軟體的安裝及系統流程的規劃，實作適用於序列型樣探勘的格網運算系統。

一、系統設計

根據資料探勘結合格網運算環境的特性，本研究實作了資料格網與運算格網兩種不同型態的格網節點，作為序列型樣探勘的資料儲存與運算的基礎。兩種型態的格網節點如下說明：

1. 資料格網(*DG*)：具大容量記憶體與硬碟之資料儲存能力的節點。主要提供分散式儲存空間的機制，藉以儲存大容量的序列資料。本研究建置16個資料格網可供使用。
2. 運算格網(*CG*)：具高速計算能力的節點。主要提供分散式處理與運算機制，有利於序列探勘的過程中，進行大規模及高速運算的資料探勘。本研究建置16個運算格網可供使用。

每個格網節點，同時提供了資料格網與運算格網兩種功能，及安裝完整分散式GSP演算法的所有函式庫，即排序、大項目集合、轉換、序列與最大序列等五個步驟的分散式演算函式。所有的分散式演算函式，可透過中介軟體(Globus Toolkit 3, GT3)封裝成不同探勘任務內容的格網服務。格網服務可經由使用者或其他的格網節點進行啟動並予以回應。中介軟體將依每個格網節點的運算能力及資料儲存的能力，認定該節點為運算格網或資料格網。格網服務存取點(Access Points of Grid Services)則依據使用者的需求，進行格網任務的運算服務，並適時地回應運算結果。格網節點的管理機制包括了節點名稱、資源、資料與任務等功能，如圖2所示。

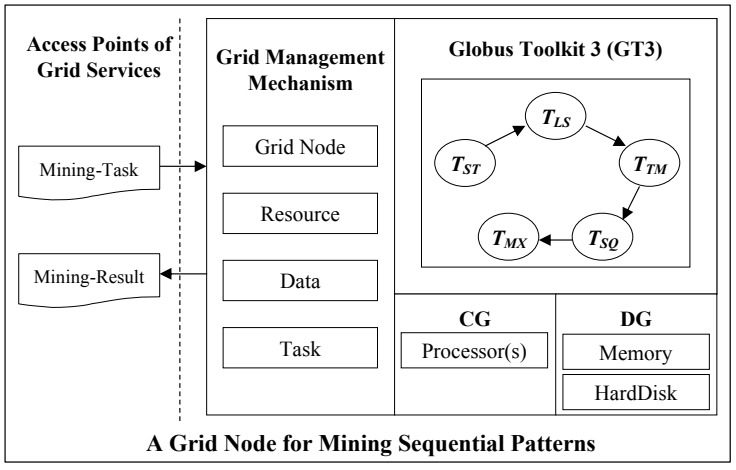


圖2：格網節點的設計與實作

二、中介軟體

本研究使用Globus Toolkit 3(GT3)作為格網的中介軟體。GT3屬於一個開放源碼軟體(Open Source Software)，由Globus 聯盟所發展(Globus Alliance)，並結合了部份學術界機關與企業組織共同推動。其目標為建置符合“Grid(格網)”的核心基礎技術，並支援開發相關格網應用的服務及函式庫。GT3的架構包括四個主要組成元件：(1)格網安全基礎設施；(2)資源管理；(3)資料管理；(4)資訊服務 (Ferreira et al. 2003)。

三、系統實作

（一）格網運算環境之需求

本研究建置格網運算環境之實驗需求，作業系統平台使用Linux、格網運算平台採用Globus Toolkit、程式語言平台則使用Java Virtual Machine。所有的資料探勘程式皆以JAVA程式語言進行開發與設計，並透過GT3的封裝技術，將探勘程式轉換成格網化的探勘任務，安裝與執行於所有的格網節點。

（二）格網節點之分佈狀態

依據格網節點的實作概念，本研究建立一具備分散處理型態的格網運算環境。以16台運算主機實作成16個格網節點，每一格網節點裝置了不同的硬體元件。這些格網節點主要分佈於兩個不同區域校園的網路環境，我們以Campus-1及Campus-2稱之，分別以兩個不同型號的路由交換器，Foundry FastIron Edge Switch 2402 與 Cisco 6509 Router，互相連結於廣域網路的網路環境。完整的格網節點之分佈狀態，如圖3所示。

（三）格網運算環境之基本資訊

表1為本研究實作格網運算環境之基本資訊。格網節點編號可辨別所屬的格網節點。處理器等級與其核心運算速率可實作成不同等級的運算格網。記憶體與硬碟容量可實作

成不同等級的資料格網。格網節點的運算主機型態，例如：個人電腦、相關系列工作站與伺服器等，及主機附屬的網路應用服務，例如：檔案傳輸伺服器與郵件伺服器等。基於安全性與隱私權考量，各格網節點所屬的網路位址將不予顯示。

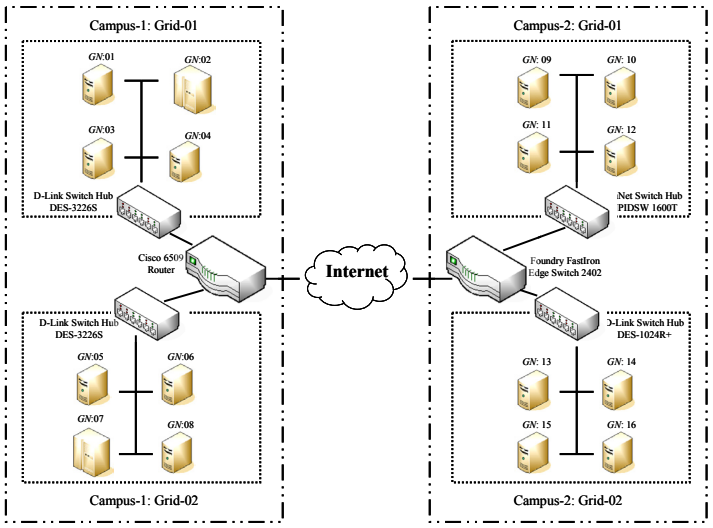


圖3：格網節點之分佈狀態

表1：格網運算環境之基本資訊

節點編號	處理器 & 運算速率	記憶體	硬碟容量	區域網路	主機型態
01	P4-3.2GHz	1024 MB	IDE-15 GB	RealTek 8139 10/100 Mbps	PC
02	P4-3.0GHz	1024 MB	IDE-10 GB	VIA VT86c100A 10/100 Mbps	HP Workstation xw4200
03	P3-733MHz	512 MB	IDE-9 GB	RealTek 8139 10/100 Mbps	PC
04	P3-800MHz*2	512 MB	IDE-35 GB	National Semiconductor DP83815 10/100 Mbps	PC (Mail Server)
05	P3-800MHz	512 MB	SCSI-12 GB	RealTek 8139 10/100 Mbps	PC (FTP Server)
06	P4-3.4GHz	512 MB	IDE-10 GB	RealTek 8139 10/100 Mbps	PC
07	P4-2.8GHz*2	2048 MB	IDE-40 GB	Broadcom BCM 5751 10/100/1000 Mbps	HP Workstation xw6200
08	P4-1.6GHz	512 MB	IDE-9 GB	RealTek 8029 10/100 Mbps	PC
09	P4-2.0GHz	1024 MB	SCSI-34 GB	Broadcom BCM 5702 10/100/1000 Mbps	IBM xSeries 205 (Mail Server)
10	P4-2.0GHz	768 MB	SCSI-34 GB	Broadcom BCM 5702 10/100/1000 Mbps	IBM xSeries 205
11	P4-2.0GHz	768 MB	SCSI-34 GB	Broadcom BCM 5702 10/100/1000 Mbps	IBM xSeries 205
12	P4-2.0GHz	512 MB	SCSI-34 GB	Broadcom BCM 5702 10/100/1000 Mbps	IBM xSeries 205
13	P3-550MHz*2	256 MB	SCSI-54 GB	Intel EtherExpress 10/100 Mbps	IBM Netfinity 3500
14	P4-2.0GHz	512 MB	SCSI-34 GB	Broadcom BCM 5702 10/100/1000 Mbps	IBM xSeries 205
15	P4-2.8GHz	512 MB	IDE-80 GB	RealTek 8029 10/100 Mbps	PC
16	P4-2.0GHz	512 MB	SCSI-34 GB	Broadcom BCM 5702 10/100/1000 Mbps	IBM xSeries 205

（四）系統架構

本研究的系統架構如圖4所示。區分成格網主控端、運算工作規劃、工作分派器、工作進度監視器、格網資源資料庫與格網運算環境等主要系統模組，如下說明所示：

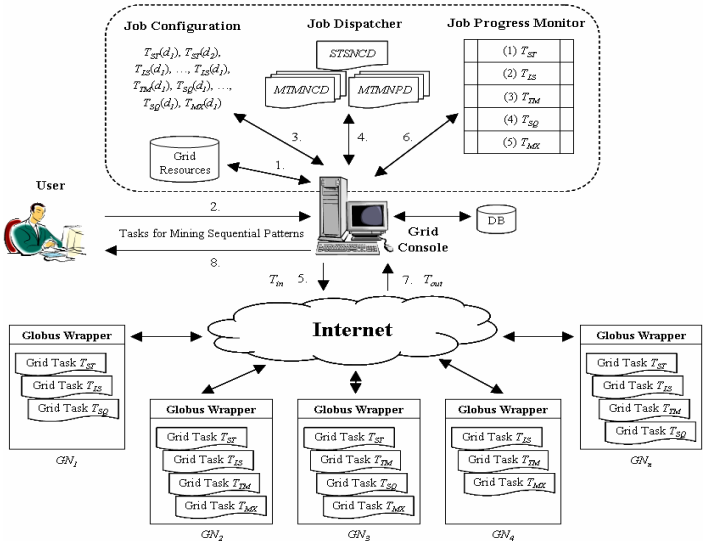


圖4：格網運算環境於序列型樣探勘之系統架構圖

格網主控端：負責監控六個模組之間的資料、資源、工作與任務等訊息傳遞。屬於本系統的主要溝通橋樑。格網主控端兼具了運算工作規劃、工作分派器與工作進度監視器的角色。以輸入與輸出任務之溝通語言，與格網運算環境的格網節點進行資料與任務的傳遞，資料傳輸機制為JAX-RPC。運算工作規劃：負責劃分資料庫與任務，讓探勘任務得以形成獨立性探勘子任務。工作分派器：負責提供*STSNCD*、*MTMNCD*與*MTMNPD*等三種不同的格網運算策略。工作進度監視器：負責監視探勘任務的執行狀態。格網資源資料庫：負責記錄所有格網節點的儲存與計算資源。格網運算環境：負責與格網主控端進行互動式的溝通。首先，接收格網任務，選擇可用性的格網資源，開始執行格網任務。最後，回應格網任務之結果。至於各工作的時間計算，將包含資料動態分派至各節點的時間。

在系統架構圖中所顯示的八個流程，其意義如下：

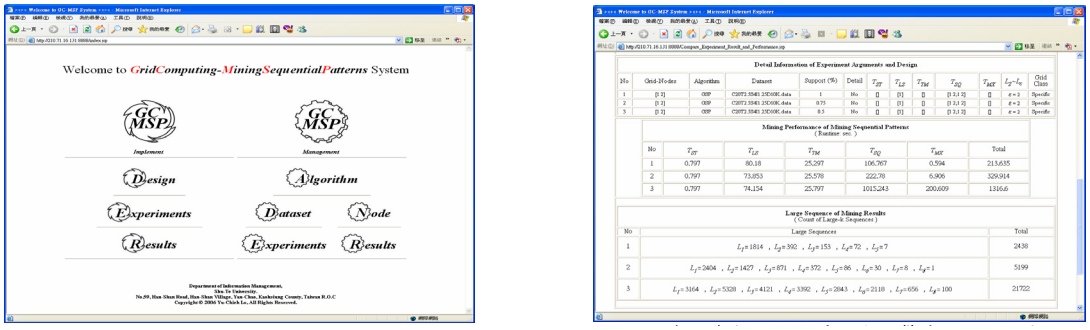
1. 格網主控端與格網運算環境進行更新格網資源資料庫的程序；
2. 使用者提出序列型樣探勘之任務需求；
3. 格網主控端依據運算工作規劃的功能，進行資料庫與任務的劃分，讓序列型樣探勘的每一探勘任務形成數個獨立性探勘子任務；
4. 格網主控端依據工作分派器的功能，為分散式探勘任務選擇合適的格網運算策略；
5. 格網主控端將分散式探勘任務的輸入程序，以JAX-RPC資料傳輸機制，傳送資料

庫與探勘任務至遠端格網節點。透過資料格網與運算格網節點，執行探勘任務的資料儲存與運算作業；

6. 格網主控端依據工作進度監視器的功能，開始監視每一探勘任務的執行狀態；
7. 每一探勘任務成功地完成，格網主控端可透過輸出程序取得探勘結果；直至完成五大探勘任務，否則，重覆步驟(3)～(7)；
8. 使用者獲得序列型樣探勘結果的相關資訊。

（五）系統介面設計

圖5為本研究所開發之JSP-based使用者介面，簡稱GC-MSP系統，主要負責管理、監督與實作格網化序列型樣探勘的分散式處理，檢視、比較探勘效能與結果，及後端系統管理作業。



(a) GC-MSP主系統介面　　(b) 實驗參數、探勘時間與頻繁序列統計個數

圖5：格網運算環境於序列型樣探勘之JSP-based系統介面

伍、實驗設計與分析

一、實驗需求與環境

本研究使用IBM所提供的模擬資料庫產生器(IBM Quest Data Mining Project 1996)，產生3個不同參數型態的序列資料庫。在這些序列資料庫中均含有10000筆的顧客交易記錄，5000筆的資料項目，25000筆的潛在最大項目集合與5000筆潛在最大序列等資料。表2為產生序列資料庫的參數設定，其中$|C|$是每個顧客的平均交易數量，$|T|$是每個交易的平均項目個數，$|S|$表示潛在最大序列型樣的平均長度，及$|I|$表示潛在最大項目集合的平均內含項目數量。

表2：產生序列資料庫的參數設定

序列資料庫	$\|C\|$	$\|T\|$	$\|S\|$	$\|I\|$	檔案大小 (MB)
*C*10-*T*2.5-*S*4-*I*1.25	10	2.5	4	1.25	1.95
*C*10-*T*2.5-*S*8-*I*1.25	10	2.5	8	1.25	1.86
*C*20-*T*2.5-*S*4-*I*1.25	20	2.5	4	1.25	3.78

本研究的序列實驗主要以Java程式語言實作序列型樣探勘，並安裝於一台個人電腦之中，此電腦裝配了Pentium-4 2.0GHz中央處理器，及512MB記憶體，作業系統為Windows 2000 Server。透過前述的3個序列資料庫，及最小支持度設定值為1%、0.75%、0.5%或0.33%，測試序列實驗結果，並將探勘效能作為比較格網實驗結果的基礎，如表3所示。表3的GSP探勘效能及搜尋序列型樣顯示，當最小支持度設定值愈小或序列資料庫容量愈大時，將可以搜尋到大量且長度較長的序列型樣，但探勘效能卻愈來愈不佳。然而，以*LS*(Litemset)與*SQ*(Sequence)任務之運算效能的影響最大。為了有效地解決探勘效能不佳的問題，本研究將*STSNCD*、*MTMNCD*與*MTMNPD*等三種分散式MSP之工作分派策略實行於格網運算環境。分散式MSP執行於不同參數型態的序列資料庫時，可搜尋到不同長度的頻繁性序列，但也造成每個任務的探勘時間將有所不同。每一分散式任務將各自選擇合適的工作分派策略，在*LS*的任務中，我們選擇使用*STSNCD*的分派策略；在*SQ*的任務中，依據所需探勘時間的不同，於LS_2採行*MTMNCD*的分派策略，而$LS_{k>2}$採行*MTMNPD*的分派策略；上述的分派策略中，因*LS*的執行時間並不長，故不採取完全分散的處理方式，以減低資料切割與合併的額外負擔(overhead)；而*SQ*的動作主要是產生項目集合的組合，並測試其支持度。其中，LS_2是整個*SQ*過程最耗時間的，且LS_2完成後才能提供後續LS_k的進行，*MTMNCD*策略以分散之工作與節點加速項目集合的產生，但以完整資料集減低項目集在不同節點中進行重複出現的檢查次數。$LS_{2>2}$採行*MTMNPD*策略，與LS_2的理由相近，但因為序列長度越長，其出現重複機會越低，檢查重複出現的動作不若在LS_2中般頻繁，故採用分散式的資料集。

表3：GSP的序列實驗結果

序列資料庫	Support (%)	Max. Seq.	ST (sec.)	LS (sec.)	TM (sec.)	SQ (sec.)	MX (sec.)	Total Time (sec.)
C10-T2.5-S4-I1.25	1	1	0.657	18.718	5.266	10.344	0	34.985
	0.75	3	0.438	16.703	5.469	20.312	0	42.922
	0.5	5	0.422	16.719	8.797	115.781	1.484	143.203
	0.33	8	0.531	16.391	5.875	70761.766	14.265	70798.828
C10-T2.5-S8-I1.25	1	1	0.688	10.984	3.750	7.516	0.016	22.954
	0.75	1	0.422	10.922	3.828	16.688	0	31.860
	0.5	2	0.422	10.906	4.125	33.406	0	48.859
	0.33	5	0.422	10.938	4.265	33490.984	0.532	33507.141
C20-T2.5-S4-I1.25	1	5	1.187	109.703	35.485	212.438	0.609	359.422
	0.75	8	2.735	109.687	25.953	117882.718	7.297	118028.390
	0.5	8	2.703	128.344	28.859	454223.860	184.187	454567.953

二、實驗設計

以下設計了兩個不同格網運算環境的實驗，加以驗證本研究之方法與設計。並針對格網運算環境與單一主機執行完整GSP的探勘效能，相互評估與比較加速效能的效果。

探勘加速效能之計算公式如下所示：

$$\text{探勘加速效能（}Speed\ Up\text{）}=\frac{\text{單一運算主機執行完整}GSP\text{的探勘時間}}{\text{格網運算環境執行完整}GSP\text{的探勘時間}}$$

（一）實驗一

實驗目的：在格網運算環境中，主要以規劃格網節點個數為2、4、6與8個等資源使用量，搭配S1、S2、S3與S4等四種分散式MSP之格網規劃，如表4所示，檢驗與比較加速探勘效能的結果，且得以產生遞增加速效能的效果。使用兩個不同的單一區域或混合區域校園的格網節點，對於加速效能也可以有具體影響。格網運算環境的16個格網節點，可劃分成兩個不同的區域校園，即Campus-1與Campus-2。實驗一的*C*10-*T*2.5-*S*4-*I*1.25與*C*10-*T*2.5-*S*8-*I*1.25的實驗內容，主要以四種格網規劃為基礎，及1%、0.75%、0.5%與0.33%等四個不同的最小支持度，測試與收集探勘效能的結果分別如表5與表6所示。實驗一的*C*20-*T*2.5-*S*4-*I*1.25實驗內容，主要以四種格網規劃為基礎，及1%、0.75%與0.5%等三個不同的最小支持度，測試與收集探勘效能的結果，如表7所示。

表4：實驗一之格網環境與實驗設計

格網群組	格網規劃	資源組態		格網節點編號	序列資料庫
		Campus-1	Campus-2		
Campus-1	S1	2	0	09,10	C10-T2.5-S4-I1.25 C10-T2.5-S8-I1.25 C20-T2.5-S4-I1.25
	S2	4	0	09,10,11,12	
	S3	6	0	09,10,11,12,14,15	
	S4	8	0	09,10,11,12,13,14,15,16	
Campus-2	S1	0	2	01,02	
	S2	0	4	01,02,05,07	
	S3	0	6	01,02,03,05,06,07	
	S4	0	8	01,02,03,04,05,06,07,08	
Campus-1 & Campus-2	S1	1	1	01,09	
	S2	2	2	01,02,09,10	
	S3	3	3	01,02,07,09,10,11	
	S4	4	4	01,02,06,07,09,10,11,15	

表5：實驗一：C10-T2.5-S4-I1.25的實驗結果

		(a) 格網群組 Campus-1		(b) 格網群組 Campus-2		(c) 格網群組 Campus-1 &Campus-2	
Configuration	Support(%)	Time(sec.)	Speed Up	Time(sec.)	Speed Up	Time(sec.)	Speed Up
S1	1	44.621	0.784	38.716	0.904	35.22	0.993
	0.75	53.215	0.807	46.169	0.930	42.415	1.012
	0.5	94.8	1.511	72.277	1.981	75.046	1.908
	0.33	356.393	198.654	299.356	236.504	236.686	299.126
S2	1	37.213	0.940	32.331	1.082	32.022	1.093
	0.75	50.905	0.843	40.792	1.052	38.049	1.128
	0.5	90.854	1.576	71.28	2.009	64.899	2.207
	0.33	345.632	204.839	276.15	256.378	217.888	324.932

		(a) 格網群組 Campus-1		(b) 格網群組 Campus-2		(c) 格網群組 Campus-1 & Campus-2	
Configuration	Support(%)	Time(sec.)	Speed Up	Time(sec.)	Speed Up	Time(sec.)	Speed Up
S3	1	34.634	1.010	29.5	1.186	30.019	1.165
	0.75	48.342	0.888	40.689	1.055	35.696	1.202
	0.5	90.321	1.585	68.57	2.088	64.079	2.235
	0.33	323.527	218.834	258.526	273.856	215.845	328.008
S4	1	34.167	1.024	28.587	1.224	28.042	1.248
	0.75	48.2	0.890	37.906	1.132	33.912	1.266
	0.5	85.067	1.683	66.557	2.152	60.983	2.348
	0.33	269.561	262.645	245.809	288.024	205.59	344.369

表6：實驗一：C10-T2.5-S8-I1.25的實驗結果

		(a) 格網群組 Campus-1		(b) 格網群組 Campus-2		(c) 格網群組 Campus-1 & Campus-2	
Configuration	Support(%)	Time(sec.)	Speed Up	Time(sec.)	Speed Up	Time(sec.)	Speed Up
S1	1	31.313	0.733	26.566	0.864	26.405	0.869
	0.75	40.877	0.779	31.785	1.002	33.146	0.961
	0.5	59.746	0.818	45.742	1.068	45.939	1.064
	0.33	107.154	312.701	74.591	449.212	77.065	434.791
S2	1	30.754	0.746	25.84	0.888	25.076	0.915
	0.75	34.199	0.932	31.486	1.012	32.298	0.986
	0.5	52.148	0.937	42.047	1.162	40.063	1.220
	0.33	96.3	347.945	74.022	452.665	69.882	479.482
S3	1	23.82	0.964	20.26	1.133	20.197	1.137
	0.75	33.828	0.942	28.826	1.105	27.439	1.161
	0.5	50.491	0.968	40.566	1.204	39.372	1.241
	0.33	94.511	354.532	70.507	475.231	67.086	499.465
S4	1	23.523	0.976	19.759	1.162	19.457	1.180
	0.75	33.801	0.943	25.949	1.228	25.377	1.255
	0.5	49.663	0.984	37.708	1.296	37.08	1.318
	0.33	88.818	377.256	69.435	482.568	63.097	531.042

表7：實驗一：C20-T2.5-S4-I1.25的實驗結果

		(a) 格網群組 Campus-1		(b) 格網群組 Campus-2		(c) 格網群組 Campus-1 & Campus-2	
Configuration	Support(%)	Time(sec.)	Speed Up	Time(sec.)	Speed Up	Time(sec.)	Speed Up
S1	1	265.311	1.355	237.807	1.511	223.465	1.608
	0.75	422.91	279.086	308.823	382.188	338.642	348.534
	0.5	1579.631	287.768	1510.683	300.902	1383.88	328.474
S2	1	250.21	1.436	201.801	1.781	187.024	1.922
	0.75	411.713	286.676	307.539	383.783	300.043	393.372
	0.5	1488.23	305.442	1399.477	324.813	1319.468	344.509
S3	1	246.152	1.460	194.928	1.844	182.717	1.967
	0.75	402.561	293.194	304.561	387.536	295.22	399.798
	0.5	1425.61	318.859	1309.911	347.022	1297.888	350.237
S4	1	212.882	1.688	184.361	1.950	164.773	2.181
	0.75	394.668	299.057	303.37	389.058	264.458	446.303
	0.5	1328.102	342.269	1248.223	364.172	1225.959	370.786

（二）實驗二

實驗目的：在格網運算環境中，主要以規劃格網節點個數為2、4、6與8個等資源使用量，搭配S1、S2、S3與S4等四種分散式MSP之格網規劃，如表8所示，檢驗與比較加速探勘效能的結果，且得以產生遞增加速效能的效果。而使用混合區域校園*TOP-8 GNs*(＝{GN_1, GN_2, GN_6, GN_7, GN_9, GN_{10}, GN_{11}, GN_{12}})具備高速運算能力與大容量儲存機制的格網節點，對於加速效能也可以有具體影響。表8為實驗二之格網環境與實驗設計。

表8：實驗二之格網環境與實驗設計

格網群組	格網規劃	資源組態	格網節點編號	序列資料庫
Campus-1 & Campus-2(TOP-8 GNs)	S1	2	01,09	C10-T2.5-S4-I1.25 C10-T2.5-S8-I1.25 C20-T2.5-S4-I1.25
	S2	4	01,02,09,10	
	S3	6	01,02,07,09,10,11	
	S4	8	01,02,06,07,09,10,11,12	

實驗二的*C*10-*T*2.5-*S*4-*I*1.25與*C*10-*T*2.5-*S*8-*I*1.25實驗內容，主要以四種格網規劃為基礎，及1%、0.75%、0.5%與0.33%等四個不同的最小支持度，測試與收集探勘效能的結果，如表9所示。實驗二的*C*20-*T*2.5-*S*4-*I*1.25實驗內容，主要以四種格網規劃為基礎，及1%、0.75%與0.5%等三個不同的最小支持度，測試與收集探勘效能的結果，如表10所示。

表9：實驗二：C10-T2.5-S4-I1.25與C10-T2.5-S8-I1.25的實驗結果

		C10-T2.5-S4-I1.25		C10-T2.5-S8-I1.25	
Configuration	Support(%)	Time(sec.)	Speed Up	Time(sec.)	Speed Up
S1	1	38.896	0.899	24.887	0.922
	0.75	47.075	0.912	30.578	1.042
	0.5	74.928	1.911	45.814	1.066
	0.33	239.912	295.103	75.941	441.226
S2	1	35.416	0.988	23.213	0.989
	0.75	41.368	1.038	26.486	1.203
	0.5	68.011	2.106	41.007	1.191
	0.33	224.338	315.590	68.606	488.400
S3	1	32.65	1.072	21.118	1.087
	0.75	40.163	1.069	26.268	1.213
	0.5	64.051	2.236	39.473	1.238
	0.33	214.445	330.149	67.598	495.682
S4	1	30.706	1.139	19.534	1.175
	0.75	38.426	1.117	25.524	1.248
	0.5	60.769	2.357	36.326	1.345
	0.33	205.367	344.743	62.63	535.001

表10：實驗二：C20-T2.5-S4-I1.25的實驗結果

			C20-T2.5-S4-I1.25
Configuration	Support(%)	Time(sec.)	Speed Up
S1	1	208.849	1.721
	0.75	322.833	365.602
	0.5	1145.153	396.950
S2	1	188.905	1.903
	0.75	284.151	415.372
	0.5	1060.218	428.750
S3	1	171.255	2.099
	0.75	278.658	423.560
	0.5	1053.463	431.499
S4	1	159.926	2.247
	0.75	260.85	452.476
	0.5	1002.76	453.317

三、實驗分析與討論

本研究設計了兩個實驗，主要基於規劃格網節點數量2、4、6與8個的資源遞增數量，結合單一區域校園/混合區域校園與混合區域校園*TOP-8 GN*$_s$等2種不同實驗環境，作為本研究檢驗格網運算環境於序列型樣探勘的基礎。以下則依據二個實驗分析的問題進行討論：

（一）格網實驗可加速序列型樣的探勘效能：透過實驗一與實驗二的所有實驗結果顯示，兩個不同型態的實驗設計皆可有效加速序列型樣的探勘效能。因此，本研究的系統確實有助於提升整體資料探勘的效能。

（二）遞增格網節點數量可遞增加速探勘效能：在實驗一與實驗二的實驗內容中，本研究使用了三種不同的工作分派策略於分散式序列型樣探勘。隨著格網實驗環境使用遞增格網節點資源，設計了S1～S4等四種格網規劃，加以調整與安排格網節點的任務處理順序及指派特定格網節點處理特定探勘任務，可達到遞增加速探勘效能的效果。

（三）根據實驗一的表7顯示，當最小支持度為0.75%時，部份實驗結果的加速效能優於最小支持度為0.5%的加速效能。主要原因為*C*20-*T*2.5-*S*4-*I*1.25資料庫的資料儲存特性，針對設定最小支持度為0.75%及0.5%時，分別得以探勘到不同長度與數量的頻繁性序列。其中以執行T_{LS}、$T_{SQ}(LS_{k=2})$與$T_{SQ}(LS_{k>2})$等三個任務時，最小支持度為0.75%的加速效果較佳。搭配本實驗所使用的格網環境、格網資源、資料庫與任務切割模式等條件，形成了最佳的分散式格網探勘模式。因此，產生此特殊的實驗結果現象。

（四）最佳的格網運算環境：透過實驗一與實驗二等2個不同型態的整體實驗結果顯示，實驗二的混合區域校園*TOP-8 GN*$_s$屬於本研究最佳的格網運算環境。一方面*TOP-8 GN*$_s$的格網節點屬於本格網運算環境中，最符合高速運算能力與大容量儲存機制

的8個格網節點；另一方面，在劃分資料庫與探勘任務，及指派探勘任務之時，可提供更完善的調整機制。

陸、結論與未來研究

經由格網運算環境於序列型樣探勘的研究方法與設計、格網運算系統的設計與實作、及大量性實驗分析與討論的過程，提出本研究對於設計與實作的貢獻與發現，及未來可供持續進行之研究與建議。

一、結論

本研究依據研究方法與設計，及系統的實作過程中，如何改善傳統GSP演算法探勘效能不佳之問題，進一步說明其貢獻與發現。

1. 本研究所設計與實作的格網運算環境於序列型樣探勘系統(GC-MSP)，可透過資料庫與探勘任務的劃分，及探勘任務的規劃與指派，實行大容量資料庫的分散式序列型樣探勘。
2. 本研究的實驗設計使用了3個不同參數型態的序列資料庫，透過格網運算環境及單一運算主機的運算機制，分別測試與收集探勘結果。並檢驗格網運算環境得以加速探勘效能的效果。因此，如何針對每一分散式任務，選擇合適的*STSNCD*、*MTMNCD*或*MTMNPD*工作分派策略，再搭配使用合適的格網規劃模式，將可更有效地提升加速探勘效能的效果。
3. 本研究所提出的格網運算環境於序列型樣探勘之設計與實作，經由兩個不同格網運算環境（實驗一與實驗二所使用之格網運算環境）的策略，拆解探勘任務實行鬆散耦合化的分散處理，可有效減輕以往單一運算主機使用大量儲存與運算資源，及解決探勘效能不佳的問題。其中，混合區域校園*TOP-8 GN*$_S$屬於本研究最佳的格網運算環境。然而，大量性實驗結果更顯示了本研究所實作的GC-MSP系統有助於加速整體序列型樣探勘的效能。

二、未來研究

本研究的系統實作過程中，發現了以下四點值得討論的研究問題點：

1. GC-MSP系統的分散運算機制，受限於各個格網節點的系統程序與資源使用量的影響、主機設備所隱藏潛在低效能的運算能力、及可能遭受相關網路通訊的資料傳輸問題等。一方面影響執行探勘任務的成功率，另一方面將影響加速資料探勘效能的成效。
2. 實行分散式MSP的過程中，每一分散式任務所執行資料庫與任務的切割模式，及格網資源的選擇，將影響整體探勘效能的結果。因此，如何切割適當數量的資料庫與任務，選擇多個運算與儲存能力較佳的運算主機執行T_{LS}與T_{SQ}兩個任務，讓探

勘效能得以達到最佳的加速效果，而不至於產生較差的加速效能。

3. 實行分散式MSP的過程中，本研究主要以自行定義資料庫與任務的切割模式，進行運算工作規劃的編排，及工作分派器的任務指派，嘗試尋找最佳的運算工作規劃、工作分派策略、及格網資源使用量。然而，比較於動態式資料庫與任務的切割模式而言，動態式的切割模式將較不容易實作。因此，如何規劃與制定動態式資料庫與任務的切割模式，將有待本研究進一步分析與探討。
4. 運算資料的資料量與此格網運算環境的效能有其必然之關係。也就是說，資料量大時，運算所需時間較長，更需要藉由格網的運算環境來改善。然而，若運算資料間並無序列之關係存在，是否需透過格網環境的運算以縮減運算時間，則有其探究之必要。因此，資料量之成長在此格網運算環境下之效能變化，可列為進一步的探討的議題。

參考文獻

1. 張昭憲、周定賢，民94，『以動態任務分配為基礎之分散式循序樣本探勘系統』，第十六屆國際資訊管理學術研討會，輔仁大學主辦。
2. Agrawal, R. and Srikant, R.,“Mining Sequential Patterns,”*in Proceedings of the 11th International Conference on Data Engineering*, March 1995, pp.3-14.
3. Agrawal, R. and Shafer, J. C.,“Parallel Mining of Association,”*IEEE Transactions on Knowledge and Data Engineering* (8:6), 1996, pp.962-969.
4. Ali, A., Anjum, A., Azim, T., Bunn, J. J., Mehmood, A., McClatchey, R., Newman, H. B., Rehman, W., Steenberg, C., Thomas, M., Lingen, F., Willers, I., and Zafar, M. A.,“Resource Management Services for a Grid Analysis Environment,”*in Proceedings of the 34th International Conference on Parallel Processing Workshops*, June 2005, pp.53-60.
5. Alpdemir, M. N., Mukherjee, A., Paton, N. W., Watson, P., Fernandes, A. A. A., Gounaris, A., and Smith, J.,“Service-Based Distributed Querying on the Grid,”*in Proceedings of the 1st International Conference on Service-Oriented Computing*, December 2003, pp.467-482.
6. Cannataro, M. and Comito, C.,“A Data Mining Ontology for Grid Programming,”*in the 1st International Workshop on Semantics in Peer-to-Peer and Grid Computing*, 2003, pp.113-134.
7. Cannataro, M. and Talia, D.,“Knowledge Grid: An Architecture for Distributed Knowledge Discovery,”*Communication of ACM* (46:1), January 2003, pp.89-93.
8. Cao, J., S. Jarvis, A. and Saini, S.,“ARMS: An Agent-Based Resource Management System for Grid Computing,”*Scientific Programming* (10:2), 2002, pp.135-148.
9. Chen, M. S., Han, J., and Yu, P. S.,“Data Mining: An Overview from a Database

Perspective,” *IEEE Transactions on Knowledge and Data Engineering* (8), 1996, pp.866-883.

10. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S., “The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets,” *Journal of Network and Computer Applications* (23:3), 2000, pp.187-200.
11. Erwin, D. W. and Snelling, D. F., “UNICORE: A Grid Computing Environment,” *Lecture Notes in Computer Science* (2150), 2001, pp.825-834.
12. Ferreira, L., Berstis, V., Armstrong, J., Kendzierski, M., Neukoetter, A., Takagi, M., Bing-Wo, R., Amir, A., Murakawa, R., Hernandez, O., Magowan, J. and Bieberstein, N., “Introduction to Grid Computing with Globus,” *IBM International Technical Support Organization*, September 2003, pp.131-143.
13. Foster, I. and Kesselmany, C., “Globus: A Metacomputing Infrastructure Toolkit,” *The International Journal of Supercomputer Applications and High Performance Computing* (11:2), 1997, pp.115-128.
14. Foster, I., Kesselman, C., Nick, J., and Tuecke, S., “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration,” Globus Project, 2002 (available online at http://www.globus.org/research/papers/ogsa.pdf)
15. Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J., “Knowledge Discovery in Databases - An Overview,” *AI Magazine* (13), 1992, pp.57-70.
16. Globus Alliance (available online at http://www.globus.org/)
17. Grimshaw, A. S., Wulf, W. A., French, J. C., Weaver, A. C., and Reynolds, P. F., “Legion: The Next Logical Step toward a Nationwide Virtual Computer,” *University of Virginia*, Technical Report, August 1994, pp.1-23.
18. Gualnik, V. and Karypis, G., “Parallel Tree-Projection-Based Sequence Mining Algorithms,” *Parallel Computing* (30), 2004, pp.443-472.
19. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. and Hsu, M. C., “FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining,” *in Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp.355-359.
20. Han, J., Pei, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. C., “Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach,” *IEEE Transactions on Knowledge and Data Engineering* (16:11), 2004, pp.1424-1440.
21. IBM Quest Data Mining Project, 1996, “Quest Synthetic Data Generation Code.” (available online at http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html)
22. JSR-000101 Java APIs for XML based RPC (Proposed Final Draft) (available online at http://jcp.org/aboutJava/communityprocess/first/jsr101/)
23. Masseglia, F., Cathala, F. and Poncelet, P., “The PSP Approach for Mining Sequential

Patterns," *in Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, 1998, pp.174-184.

24. Natarajan, R., Sion, R., Apte, C., and Narang, I. S., "A Grid-Based Approach for Enterprise-Scale Data Mining," *in Workshop on Data Mining and the Grid at the 4th IEEE International Conference on Data Mining*, November 2004, pp.1-8.
25. Rahman, R. M., Barker, K., and Alhajj, R., "Replica Selection in Grid Environment: A Data-Mining Approach," *in Proceedings of the 2005 ACM Symposium on Applied Computing*, March 2005, pp.695-700.
26. Roughan, M. and Zhang, Y., "Secure Distributed Data-Mining and its Application to Large-Scale Network Measurements," *ACM SIGCOMM Computer Communication Review* (36:1), 2006, pp.7-14.
27. Silberschatz, A. and Galvin, P., *Operating System Concepts*, 6th Edition, John Wiley & Sons, 2003.
28. Srikant, R. and Agrawal, R., "Mining Sequential Patterns: Generalizations and Performance Improvements," *in Proceedings of the 5th International Conference on Extending Database Technology* (Lecture Notes in Computer Science 1057), 1996, pp.3-17.
29. Swany, M. and Wolski, R., "Building Performance Topologies for Computational Grids," *International Journal of High Performance Computing Applications* (18:2), 2004, pp.255-265.
30. Zaki, M., "Parallel and Distributed Association Mining: A Survey," *IEEE Concurrency*, (7:4), 1999, pp.14-25.