運用 Java 之對等式平行計算中介軟體之設計

Peer-to-Peer Middleware Design for Parallel Computing in Java

盧能彬　　林芳瑜　　楊漢鵬

長庚大學資訊管理研究所

E-mail: nplu@mail.cgu.edu.tw, m9244023@stmail.cgu.edu.tw,m9344018@stmail.cgu.edu.tw

摘要

本論文透過對等式網路技術，結合起網路節點上的硬體資源形成一個龐大的計算環境，並利用平行計算的架構，實現對等式平行計算系統的建置。此系統平台包括索引伺服器端及節點客戶端：索引伺服器端負責節點登入及節點狀態偵測；而節點客戶端則負責節點資源監控及計算物件的分配機制。藉此系統的建立，讓網路上的節點分享出其多餘的計算功能，並將其所釋放出來的資源結合起來。藉此，我們可將大規模的資料計算及分析的工作，分配到網路上閒置的電腦中執行，充份利用閒置資源來分擔工作負載量，達成資源與服務共享的目的。本研究利用 Java 1.4.2 版程式語言進行系統實作，並使用 Java RMI 的連接方式架構對等式網路。最後，在我們的系統平台上進行平行程式效能測試。從實驗結果得知，在七台電腦節點的參與下，可得到 5 至 6 倍的速度提升值，這樣的效能數值確實證明了本系統的優越性。

關鍵詞：對等式網路技術、平行計算、Java RMI(Java Remote Method Invocation)。

一、前言

隨著人類科技的發展，許多用以輔助研究領域的技術得以被提出，並且被大量使用。在研究的過程中，所產生的計算性資料數量龐大，大多已超乎現有單一電腦硬體所能處理的範圍，但為了解決這樣的計算需要，所必須付出龐大的成本，並不是每個研究團隊人所能負荷，因此利用網路連接來組合許多個人電腦與工作站來達成資料處理的平台漸成為主流[2, 6]，平行計算系統也因應而生。平行計算，即是利用多處理器系統，將一個擁有龐大計算量的工作，加以切割分散到多個處理器單元上，藉由多個處理器同時負責該計算工作的進行，來降低其執行的時間。

另一方面，隨著網路環境的成熟，對等式網路(peer-to-peer, P2P)架構逐漸受人重視[1, 7, 11]。P2P 可以是個軟體程式，同時擁有網路上客戶端（client）和伺服端（server）的雙重能力，透過兩端硬體資源分享，分享包括硬碟儲存、計算處理、以及通訊能力。而依據其運作的方式，大致上可分為 Pure Peer-to-Peer、Peer-to-peer with a simple discovery server、Peer-to-peer with discovery and lookup servers、及 Peer-to-peer with discovery, lookup, and content servers 等幾種[1]。為了讓使用者擁有的電腦分享出其多餘的計算能力，因此本論文提出了以對等式網路來架構此平行計算環境的理念，藉由網路上資源的分享，來整合高度的計算能力。

二、系統分析與設計

本節詳細介紹本研究如何結合起網路上不同電腦的計算資源。在本研究的系統中，除了具備一個對等式網路所對應的基本功能外，當使用者提出計算的需求時，系統會收集目前可提供計算資源的網路結點，並將計算分配到不同的電腦節點中執行，除此之外，每個節點可以設定分享的資源量，並隨時監控其他節點的硬體及網路相關資訊，讓網路上的節點資源獲得完善的管理。

2.1 系統架構

參與本研究的節點，分散在不同的地理位置上，不可能將每兩台電腦間實際建立實體線路，因此我們利用對等式網路技術結合各個節點的計算資源。透過網際網路中 TCP/IP 通訊協定，並結合應用層(Application Layer)建立起對等式網路。整體系統架構的設計，依據對等式網路架構中的運作方式，切割為兩大部分，一個為節點(peer)對索引伺服器(index server)的系統架構，另一個部分為節點(peer)間的系統架構。

2.1.1 節點對索引伺服器

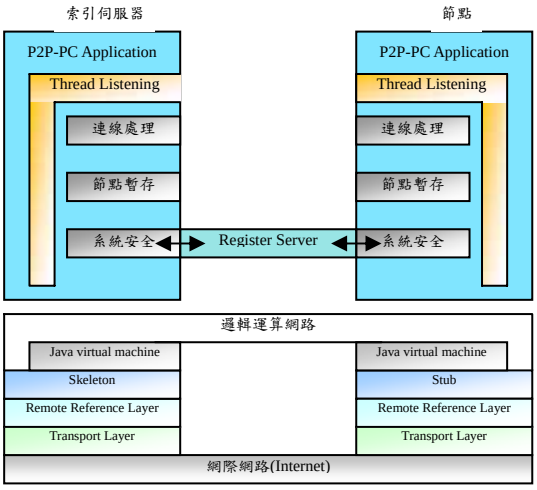

圖 2.1 系統架構圖(節點對索引伺服器)

節點與索引伺服器之間的系統架構，如圖 2.1 所示，主要功能在負責節點端對系統的註冊，透過此目錄主機，每個參與計算的節點可快速登入系統，並且找到其他可提供計算資源的節點。在此部分的架構中，功能上較於簡單，主要用作節點上線時的入口，並負責節點資訊的儲存，讓每個節點掌握彼此的位置資訊。

2.1.2 節點對節點

圖 2.2 為節點間系統架構，其主要在制定兩個節點端的連接模式，從最底層的網路硬體層、傳輸層，中間透過 Java Virtual Machine 的轉換，形成上層的運算網路。每個節點端，透過 thread 的功能，分別控制所有系統運作功能。

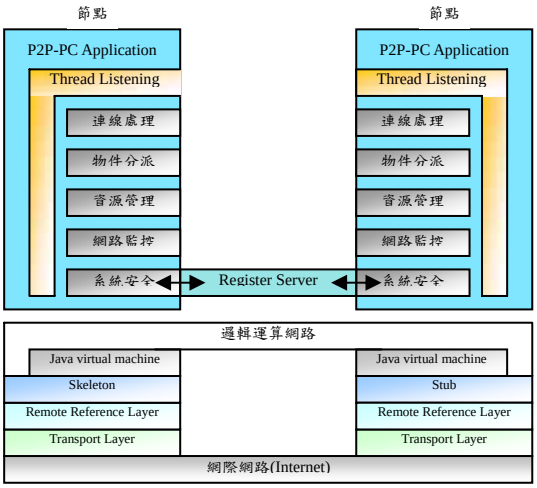

圖 2.2 系統架構圖(節點對節點)

由於參與系統的節點所運行的作業系統並不一致，如果要充分使用每個節點所擁有的系統資源，就必須使應用程式能夠跨越不同平台的限制，因此本研究利用 J2SE 的技術，藉由 Java Virtual Machine 可以讓應用程式在不同平台上運作，並控制不同網路中的節點，協調彼此之間的資源調度，使其運作起來類似一個大型電腦主機，並提供一致性的對等式平行計算功能。

2.2 系統分析

統一模塑語言(Unified Modeling Language, UML)是一套圖形語言，利用視覺化的圖形來訂定、建立、及記錄軟體開發的文件，可以用在軟體開發的各個階段，並利用標準化的方式表示出軟體的概念，包括程式語言的類別(Class)關係、資料庫綱要(Database Schema)、以及其他系統元件。本研究利用了使用案例圖(Use case diagram)及類別圖(Class diagram)來描述各個流程間的行為關係。

2.2.1 使用案例圖

在本系統中，主要有三個行為者(Actor)，中間動作主要歸類出其動作者負責功能的關係圖，如圖 2.3 所示。行為者分別為提出計算要求的節點端，主要取得提供服務節點的相關資訊及資源，進行物件分配；提供服務的節點端則提供相關計算資源；索引目錄主機則掌管所有在線上的節點位址。

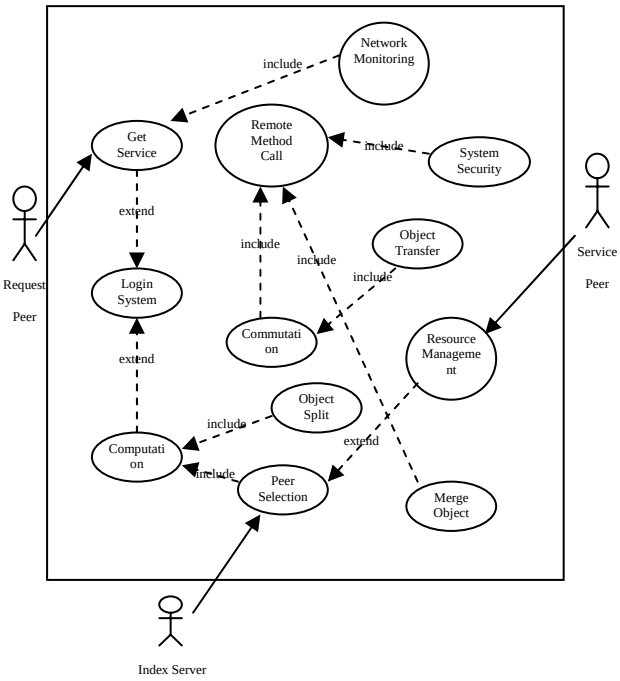

圖 2.3 使用案例圖

2.2.2 類別圖

因系統架構及功能的不同，區分成索引伺服器及節點端。索引伺服器端除了負責連線的功能外，主要是暫存節點資訊為最重要部分，此結合資料庫

設計的觀念進行資料的暫存動作，圖 2.4 為索引伺服器端的類別關係圖。

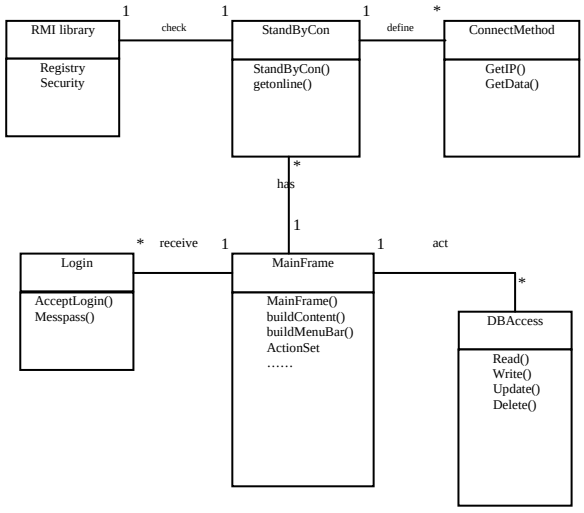

圖 2.4 類別關係圖(索引伺服器端)

節點端除了基本的連線功能外，主要還包含了節點資源的整合，另外即是處理平行化計算工作的進行程序，圖 2.5 為節點端的類別關係圖。

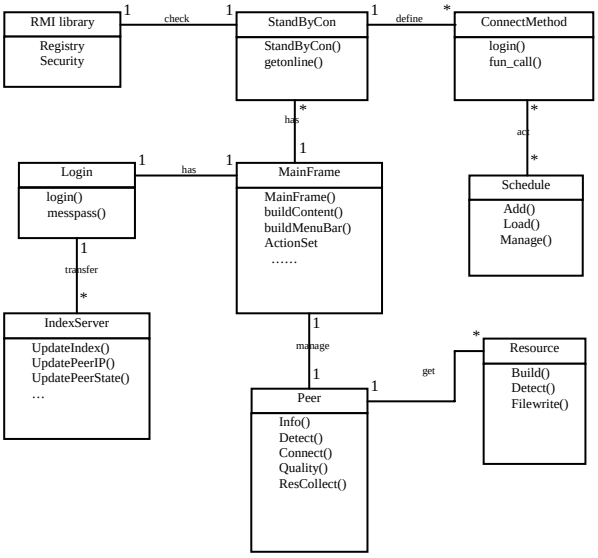

圖 2.5 類別關係圖(節點端)

2.3 系統單元

在本研究的架構設計中，當計算排程(schedule)載入應用程式中進行計算時，系統會先進行節點資源的配置，中間過程須利用到許多相關系統功能，其中包括：(一)節點之間的連線處理、(二)硬體資源的管理設定、(三)排程或計算物件的切割、(四)節點網路監控、以及(五)系統連線的安全。

2.3.1 連線處理

不同電腦主機間交換資訊，大多直接利用 socket 通訊[5]，但 socket 卻必須自己解決程式間通訊協定及資料交換的方式。為了簡化這些程序，Java 提供了 RMI (Remote Method Invocation) 的界面[13]，讓使用者不需接觸低階的 socket，而能讓不同主機上的 Java 物件能夠相互交換資料。遠端物件(remote object)存放在各個伺服器端，而各有其一個遠端界面(remote interface)與其對應。用戶端可以引用這些已定義好的遠端物件，如同呼叫於本地端電腦的物件一樣，其通訊架構如圖 2.6 所示。

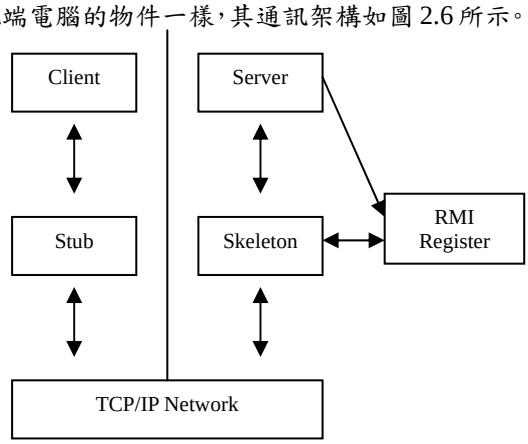

圖 2.6 RMI 通訊架構圖[5]

利用 RMI 機制連線時，需要輸入節點 IP 位址及通訊埠，才能順利連接其他網路節點，進行物件的傳送。

2.3.2 物件分配原則

計算排程要進入系統執行前，會進行物件的分派[8]，而分派的物件要傳送到哪個節點中進行計算的動作，都是影響整個計算的效能。物件資料主要是由資料以及計算方法函式所組成，經過切割程序後，再透過 RMI 的功能，將呼叫的方法函數與切割資料包裝成一個物件，分散傳送出去分配到各個節點中進行計算的工作，如圖 2.7 所示。

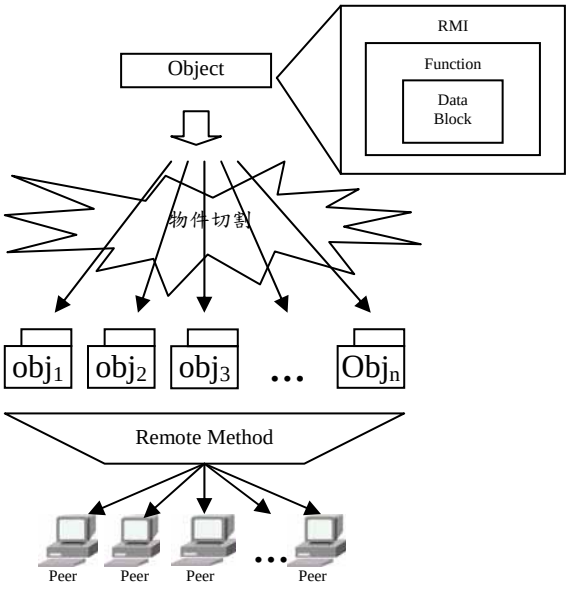

圖 2.7 物件分配示意圖

因為物件須透過網路傳送，所以物件的大小傳送及次數會影響通訊的時間，但是每個平行程式計算方法函數複雜度皆不一，且所切割成的物件數量多寡也不一致，加上節點 CPU 速度和網路速度等相關因素，讓物件的切割很難有一定的標準。因此我們考慮以方法複雜度、網路傳輸快慢以及 CPU 速度來決定物件切割的大小及物件分配原則。首先針對計算方法的複雜度來訂定資料切割區塊大小，如表 2.1 所示。

表 2.1 資料切割原則

計算方法複雜度	高	低
資料切割區塊	小	大

在一定的資料量下，當計算方法複雜度高時，所需的計算量會較高，因此考慮將資料切割區塊縮小；反之，當計算方法複雜度較低時，將資料切割區塊放大。接著再由網路速度及 CPU 速度來決定物件分配原則，我們將理想狀況訂定為 CPU 速度及網路速度皆很快的情況下為基準點，給予一定的物件分配量，而當網路速度快而 CPU 處理速度慢時，即降低分配物件數量，增加通訊次數；而當網路速度慢，CPU 處理速度快時，增加分配物件數量，減少通訊次數。分類情況如表 2.2 所示。

表 2.2. 物件分配原則

CPU / 網路	快	慢
快	分配量不變	分配量減少
慢	分配量增加	分配量減少 分配量增加

其中而當網路速度與 CPU 速度皆慢時，我們將其情況再細分為兩種：其中一種為，當 CPU 計算的速度高於網路傳輸的速度，即 CPU 執行計算的時間低於網路傳輸的時間，我們選擇增加分配物件數量，減少通訊次數；反之，當 CPU 計算的速度低於網路傳輸的速度，選擇減少分配物件數量，增加通訊次數。

2.3.3 資源管理

在本架構中，網路節點的選擇是影響計算效能的重大部分。而透過網路使用者的資源分享，讓計算得以順利進行。而計算所運用到的硬體不外乎處理器(CPU)、記憶體(Memory)、以及儲存空間(Storage space)等，使用者可以自行決定提供多少硬體資源供其他節點使用，除此之外，系統也必須偵測使用者所使用的硬體資源情況，提供計算執行時，挑選適合的節點參與計算。以下針對本論文所運用的資源參數，分別介紹其功能。

- 處理器(CPU)時脈與使用率

各節點的 CPU 使用率代表此節點目前的 CPU 使用的程度，利用此指標來判定此電腦目前的狀態為何，因此加入此指標作為判定系統的參數。

- 記憶體使用量

各節點的記憶體使用量代表此節點目前的記憶體使用的程度，利用此指標來判定此電腦目前可分享出的記憶體量多寡，尤其記憶體的可用量會影響程式執行的效能，而從此指標下，也可看出系統狀態繁忙或是正處於閒置的狀態，亦會影響此系統計算能力，因此加入此指標作為判定系統的參數。

- 硬碟的可用空間

電腦的硬碟空間會影響所執行程式的資料輸出與輸入，或是作為中間計算資料暫存空間，當計算量越大時，所處理的暫存資料量也會增加。因此，儲存空間的多寡，也會影響程式計算的進行。

- 要求連線的數目

依各節點電腦效能的不同，可接受連線的數目也不盡相同，可接受連線的數量越多，代表可處理的工作量也較多，因此也是此系統考慮的範圍之一。

2.3.4 節點監控

網路的連結架構，會直接影響到點與點之間的資料傳送與接收，如果系統中的節點分佈在不同的網域下，正常提供服務的機率就會不同，甚至無法提供連線的要求，因為網路的架構必須納入此研究的考量中，也必須仔細評估每個節點在所出的網路環境下所能提供的連線服務正常的機率。因此，必須在索引伺服器端部分納入一項功能，即對於資料庫中的登錄的節點做長期的偵測統計，以供系統在選擇可用節點的一項評估依據。

網路品質是影響此平行計算系統的重要因素之一，網路傳輸將考慮傳輸的速度與連線的穩定性，利用 Tracert Route 及 Ping 功能，找出與所連接節點中間的轉接次數及回應時間(response time)，避免資料在傳送的過程浪費太多時間與減少資料的遺失。

2.3.5 系統安全

利用 RMI 進行資料的交換，必須進行遠端物件(remote object)的註冊，此部分由 RMI Register 產生的 security manager 所負責，用來執行必要的安全檢查，防止載入的物件類別(class)執行不當的操作。

三、系統功能實作

3.1 系統實作功能

由於對等式計算中介軟體程式的設計，是為了讓許多使用者透過這樣的程式，來成就彼此分享資源的目的，因此系統在設計時，就要考慮到網路上使用者所使用的作業系統平台的差異，為了連接起不同平台上的硬體資源，因此採用跨平台的應用程式來設計系統，其所實作的功能會在這個章節詳細的介紹，並且針對索引伺服器端及節點端，分別展示出系統畫面。

➢ 索引伺服器端

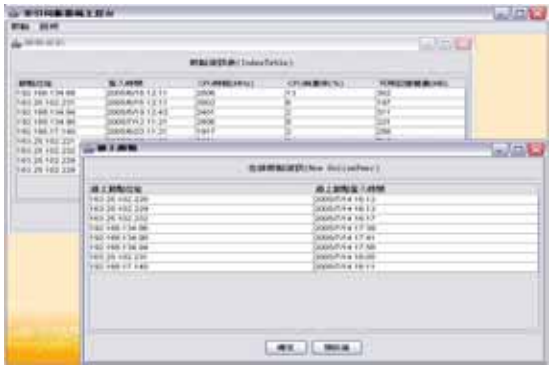

圖 3.1 索引伺服器端介面圖

如圖 3.1 所示，索引伺服器端主要負責目前線上節點位址及節點資訊的暫存，讓各節點快速取得目前的可用資源情況。我們將此功能區分階層如圖 3.2。

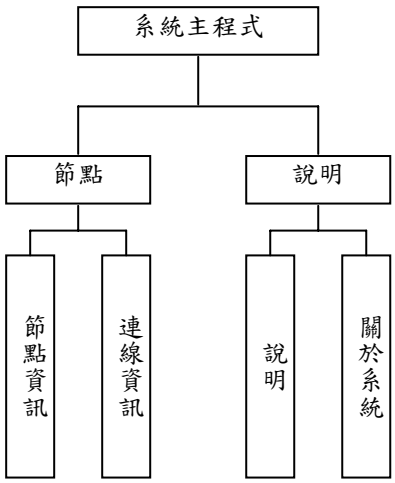

圖 3.2 索引伺服器端功能階層架構

➢ 節點端

節點端利用了 Eclipse-SDK 軟體加以封裝成一個可執行檔，以便於使用者安裝，使用時只需將程式直接啟動即可，如圖 3.3 所示。

圖 3.3 節點端系統啟動

當系統啟動時，程式必須連接到目錄索引伺服器，並且執行一些應用程式初始化的動作，而為了讓使用者了解目前的執行情況，在系統執行畫面中，訊息欄可顯示目前程式執行的階段。

➢ 程式主程式

圖 3.4 主程式介面

在圖 3.4 的左上方，有一列系統功能選單，內容分為：「計算」、「節點」與「說明」，圖 3.5 則為節點端的功能階層架構。

- 計算：新增排程(新增一個計算排程並設定計算參數)、載入排程(將設定好的排程載入執行)、排程資訊、系統組態(設定分享的資源量)。
- 節點：節點資訊(硬體狀態及上線與否)，連線資訊(節點連線狀態)、節點品質(節點上線統計值)。
- 說明：系統使用說明，及關於本系統。

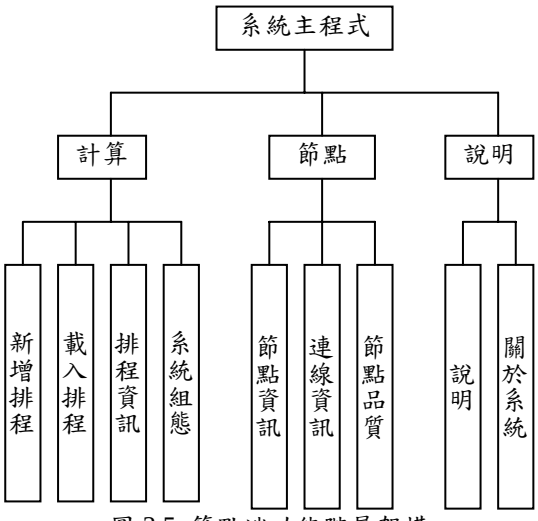

圖 3.5 節點端功能階層架構

➢ 計算功能

圖 3.6 新增計算排程介面

新增排程主要讓使用者挑選欲計算的平行程式，並設定相關的計算參數並進行儲存的動作，如圖 3.6。接著利用功能選單中的「載入排程」，將排程啟動並進行計算。而透過「排程資訊」，可了解目前系統中所有排程的執行情況，如圖 3.7 所示。

圖 3.7 排程資訊介面

➢ 系統組態設定

系統的基本設定，都可藉由這個部分的操作介面來達成，主要設定節點欲提供出來的系統資源量，包括 CPU 使用率上限、可用記憶體比率、硬碟空間量、以及節點最高連線數等，參見圖 3.8。

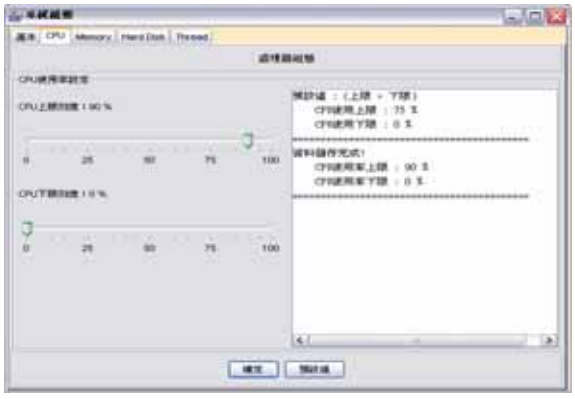

圖 3.8 系統基本組態設定

➢ 系統資源狀態

系統狀態分為兩個部分，主要有「節點連接狀態」和「節點硬體狀態」。節點連接狀態讓使用者了解目前搜尋到的所有節點是否登入系統中，並參與計算的進行；節點硬體狀態，主要讓使用者了解參與計算的節點，所擁有的硬體資源目前使用的情況，可供作挑選適合計算節點之用途，參見圖 3.9。

圖 3.9 系統資源狀態圖

➢ 節點評估

為了讓計算執行更穩定執行，必須從所選擇的節點進行，因此系統會記錄包括節點回應時間、上線次數統計、以及總上線時間，目的在於優先挑選出連線回應較快、節點參與比率高，及提供計算資源時間較長的節點，將其納入所選擇的節點，以做為整個計算排程的資源節點來源，參見圖 3.10。

圖 3.10 節點評估介面

3.2 資料庫設計

在資料庫設計方面，分為兩個部分來設計，一個是索引伺服器的資料庫，另一個是節點端的資料庫。

➢ 目錄索引伺服器

索引伺服器利用兩個資料表(table)，分別為 IndexTable 資料表。主要記錄節點網路位址、節點硬體相關資訊，詳細請參見表 3.1。

表 3.1 資料表規格(IndexTable)

編號	P2P-01	資料表名稱	IndexTable	
索引		資料表說明	參與節點資訊	
序號	欄位名稱	欄位代號	型態	備註
01	*節點位址	Peer_IP	VC	*號為主鍵
02	上線時間	Peer_Time	VC	
03	處理器時脈	Frequency	INT	
04	系統負載率	Loading	INT	
05	記憶體量	Memory	INT	

OnlinePeer 資料表：記錄目前正參與此平行計算環境的節點，包含節點的 IP 位址，定時偵測時間記錄，請參見表 3.2。

表 3.2 資料表規格(OnlinePeer)

編號	P2P-02	資料表名稱	OnlinePeer	
索引		資料表說明	線上節點資訊	
序號	欄位名稱	欄位代號	型態	備註
01	*節點位址	Online_IP	VC	*號為主鍵
02	偵測時間	Online_Time	VC	

- 節點端

節點端利用三個資料表(table)，分別為 PIPTable 資料表：記錄線上其他參與節點的網路位址，及其線上狀態，詳細請參見表 3.3。

表 3.3 資料表規格(PIPTable)

編號	P2P-03	資料表名稱	PIPTable	
索引		資料表說明	記錄偵測節點資訊	
序號	欄位名稱	欄位代號	型態	備註
01	*節點位址	Peer_IP	VC	*號為主鍵
02	節點狀態	Peer_State	INT	

PeerConnect 資料表：記錄目前參與排程計算的節點相關資訊，包含節點的參與情況、網路統計數值等等，請參見表 3.4。

表 3.4 資料表規格(PeerConnect)

編號	P2P-04	資料表名稱	PeerConnect	
索引		資料表說明	參與計算節點資訊	
序號	欄位名稱	欄位代號	型態	備註
01	*節點位址	Peer_IP	VC	*號為主鍵
02	上線統計	Total_Online	INT	
03	回應時間	Response_T	INT	
04	連接時間	Connect_T	INT	

Mission 資料表：記錄目前排程的執行情況，包括排程的名稱、排程目前狀態，以及啟動執行的時間等等，請參見表 3.5。

表 3.5 資料表規格(Mission)

編號	P2P-05	資料表名稱	Mission	
索引		資料表說明	排程執行情況	
序號	欄位名稱	欄位代號	型態	備註
01	*排程名稱	M_Name	VC	*號為主鍵
02	排程狀態	M_State	INT	
03	啟動時間	M_STime	VC	

3.3 系統運作流程

為了詳細介紹整個系統運作流程，首先須定義系統中三個不同角色，包括目錄索引伺服器、服務要求節點、服務節點。發出計算要求服務的節點稱為服務要求節點，接收物件並計算的節點稱為服務節點，他們之間都必須透過 RMI 的連線請求至索引伺服器，藉以登入系統，取得線上節點相關資訊。而之前所提到的連線處理、物件分配、資源管理等，皆交由系統自動處理，對於使用者而言，只需啟動系統，系統會抓取相關資訊，資料正確無誤的話，即能登入系統，進入計算排程的介面。

3.3.1 節點與索引伺服器

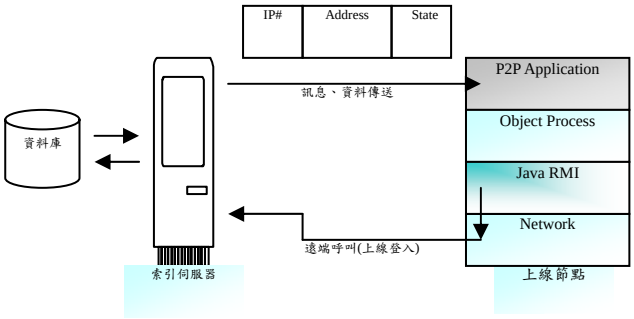

圖 3.11 節點與索引伺服器

圖 3.11 為節點與索引伺服器間的流程示意圖，當節點連上網路時，首先必須啟動系統並登入目錄索引伺服器，透過 RMI security manager 驗證連線的要求，通過後可接著傳送包括節點編號、節點 IP 位址、及節點目前的狀態等等節點資訊。以上步驟完成後，才算正式完成系統的啟動。

3.3.2 要求服務之節點與提供服務之節點

圖 3.12 為要求服物節點與提供服務節點的流程示意圖，節點只有在計算進行時才會進行連線，兩者是以對等的角度進行資料的溝通，並透過 RMI 連線來進行訊息及物件交換。而 RMI 的功能讓服務要求節點進行計算時，類似在自身節點執行般地使用其他服務節點的資源。

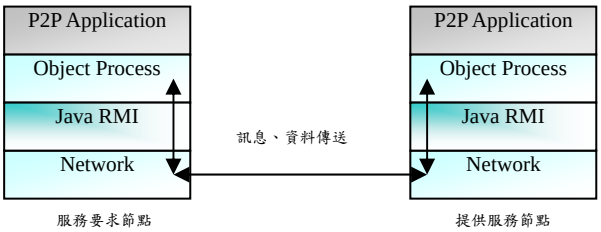

圖 3.12 要求服務之節點與提供服務之節點

四、系統模擬

本節我們針對所開發的系統進行測試，藉由數個不同演算法，透過平行化的動作，實際進行相關效能數據的測試，並分析其數據資料。本章首先介紹本系統的實驗環境，包括所運用到的電腦設備，以及其所處的網路環境。接著進行各個平行化程式架構分析，並進行效能的量測。

4.1 實驗環境

在電腦設備方面，除建置 1 部目錄索引伺服器外，共有 8 部電腦節點參與系統測試。詳細設備清單整理如下。

➢ 硬體設備部分

系統實際測試時，會有 1 部電腦作為主控節點，其餘 7 部則作為整個計算進行的從屬節點，詳細硬體設備清單如表 4.1。

表 4.1 硬體設備列表

CPU：Intel Pentium 4 2.8GHz RAM：512MB DDR HD：80GB	CPU：Intel Celeron 2.66GHZ RAM：512MB DDR HD：80GB
4 部	2 部
CPU：Intel Pentium 4 3.0GHz RAM：512MB DDR HD：80GB	CPU：Intel Pentium 4 2.4GHz RAM：256MB DDR HD：40GB
1 部	2 部

➢ 網路環境部分

我們將實驗納入的電腦節點，利用獨立網路交換器(switch)將其連線，每個電腦給予各自的電腦位址(IP address)，供判別網路位置之用， 其連接方式如圖 4.1 所示。

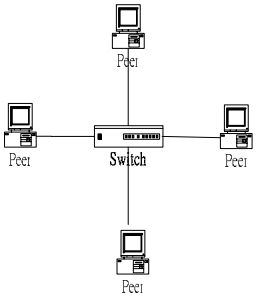

圖 4.1 節點網路架構圖

4.2 程式平行化流程

傳統循序程式通常以線性方式在執行，所以不用擔心同步存取變數的問題，或是同時有兩個程式存取通訊通道。而平行程式與循序程式在流程上則有很大的差異，因此許多執行步驟就要在程式一開始撰寫時設計完成。通常我們很難直接將一個應用程式轉換到大的平行處理系統中。因此，Ian Foster 提出一個清楚的設計平行系統的方法，稱為有條理的設計「Methodical Design」[3]，將程式設計流程分為四個步驟，如圖 4.2 所示。

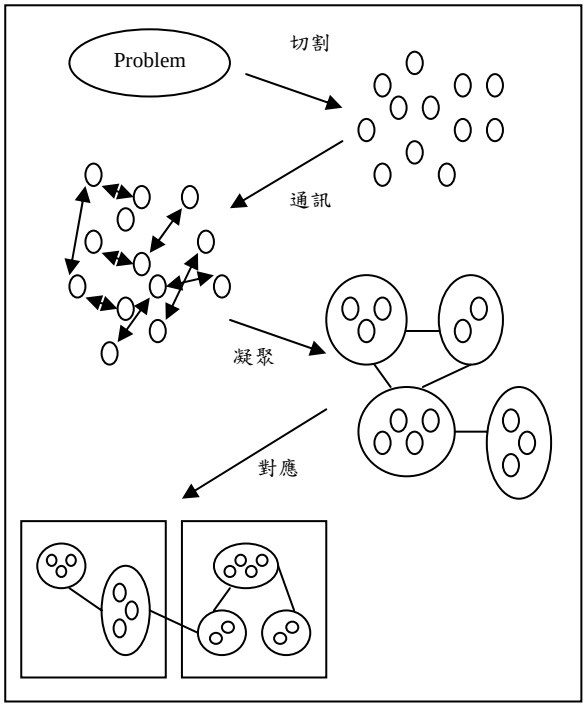

圖 4.2 PCAM 流程圖[3]

➢ 切割 (Partitioning)

將欲執行的問題範圍分解成幾個可以平行處理的較小工作。好的切割方式必須依據問題大小及每次計算參與的資料區塊大小所決定，因此，切割又細分為資料切割(Domain Decomposition)與功能切割(Functional Decomposition)兩種方式。

➢ 通訊 (Communication)

當一個平行程式執行時，會有一個主控行程(Master Control Process)及多個從屬行程(Slave Process)存在不同的處理單元中，主控行程會發送計算任務要求，從屬行程則是回傳計算結果，在這過程中或進行多次的通訊程序，通訊的次數及時間長短是影響效能的一個重要關鍵，而通訊又跟資料切割區塊的大小有很大的關聯。因此，必須評估被分割的物件於何時需要進行通訊、通訊資料量，並挑選出最有效率的通訊通道。

➢ 凝聚 (Agglomeration)

當各個從屬行程結束計算，必須將資料回傳給主控行程端，並作彙整及寫入硬碟，但每個從屬行

程依據其處理器能力不同，而有計算速度不一致的情況產生，因為有可能會有先分派的工作晚傳回，後分派的早傳回，產生同步上的問題。較簡單的處理方法是讓主控行程等待所有資料回傳完成後，再進行資料整合寫入的動作，這樣雖會浪費較多的記憶體空間，也會讓計算負載有不平衡的情況產生，但這樣的作法在處理上會比較容易實現。

- 對應 (Mapping)

此部分工作在於將前三階段對應到真正的平行處理環境中。在行程的對應上，主要分為兩種。一種是一個從屬行程只對應到一個處理單元；另外則是多個從屬行程對應到單一處理單元，至於採用何種方法，得視平行化對象來決定。此外，在設計演算法及程式時，也並非針對某個硬體作業平台，不過通常同質性系統的效能會優於異質性系統。

接著我們實際將程式執行流程利用此平行化方法的四個步驟進行探討切割、通訊考量、凝聚、以及對應部分，參照圖 4.3 所示。將前兩步驟定位在工作切割及通訊，後兩步驟強調針對硬體進行實際測試及效能評估，藉此達成平行計算的目標，。

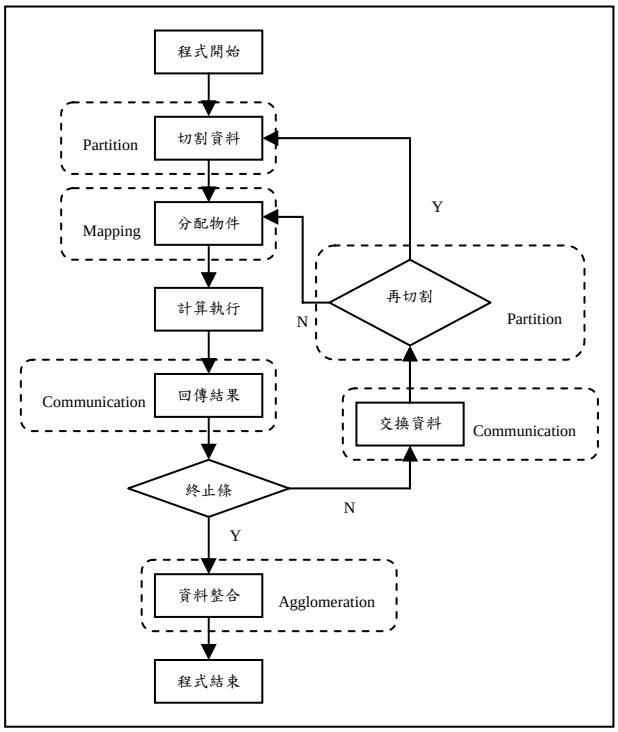

圖 4.3 程式平行化流程

4.3 平行化對象與架構比較

針對此研究架構，我們挑選了五個性質不同的演算法程式，依據 Methodical Design 的方法來設計平行程式[3]，並進行平行程式的設計與撰寫[12]，其中各個程式架構各有其特性，如表 4.2、表 4.3 所示。從質數篩選開始至 N 后問題，平行化的程度越來越複雜，從單純的資料物件切割方式，轉換到以函數物件呼叫的型態；單純傳送物件資料轉至節點間進行頻繁溝通過程；資料整合困難度提高；單純節點分配轉而需考慮切割、通訊、整合等因素，才能進行節點對應步驟[9, 10]。

表 4.2 平行程式架構比較(1)

	切割	通訊
質數篩選	切割篩選範圍	篩選結果傳送
循序搜尋	切割搜尋範圍	搜尋結果傳送
合併排序	切割合併資料	排序結果傳送
快速排序	切割排序資料	1. 排序結果傳送 2. 節點交換排序資料
N 后問題	切割函數呼叫	1. 放置解法回傳 2. 節點更新解法資料

表 4.3 平行程式架構比較(2)

	凝聚	對應
質數篩選	合併篩選結果	分配給所有節點
循序搜尋	合併搜尋結果	分配給所有節點
合併排序	合併排序資料	分配給所有節點
快速排序	合併排序資料	1. 分配給所有節點 2. 本地端節點
N 后問題	整合所有解法	1. 分配給所有節點 2. 本地端節點

除此之外，本研究所測試的計算環境主要透過網路進行物件資料的傳送，此部分直接影響到切割與通訊之間的平衡性，依據對網路速度的依存程度，此五個平行程式大致上可分為兩個不同的類型，質數篩選、循序搜尋、以及合併排序可歸類為一組，不需要透過網路交換大量資料；而快速排序及 N 后問題則歸類為另一組，計算複雜度高且計算量大，需要藉由網路交換大量資料，整理如表 4.3。

表 4.3 平行程式架構比較(3)

	物件切割類型	資料交換數	資料交換量
質數篩選	資料分解	較少	較少
循序搜尋	資料分解	較少	較少
合併排序	資料分解	較少	較少
快速排序	資料分解	大量	較多
N 后問題	功能分解	大量	較多

4.4 效能模擬測試

在效能測試的部分，考慮到所參與計算的節點所擁有的計算效能不盡相同，所以為了計算效能上的考量，採用將效能較好的節點優先納入的原則。接著我們將所有節點連接上線，實際測試節點數對於速度提升的影響[4]。

實驗測試結果如圖 4.4 所示，五個平行程式的速度提升值在 7 個節點參與時，提升值皆介於 5 至 7 倍之間，質數搜尋的速度提升值最高，而 N 后問題的提升值最少。主要是程式平行化架構上的

差異性所導致的現象。

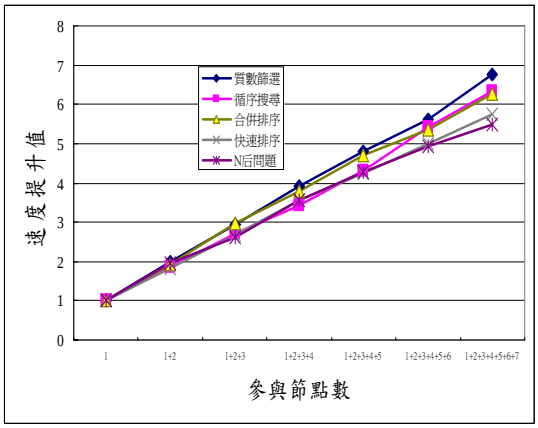

圖 4.4 參與節點數與速度提升值

4.5 討論

從測試結果中得知，在理想網路環境下，透過對等式環境來進行平行計算，確實能提升計算速度，提高效能。但平行計算並不是增加處理單元，就能加快計算處理的速度，平行程式的架構與網路傳輸皆影響到了其速度提升的幅度。程式架構影響平行化成本，網路則是影響通訊成本，必須儘量降低兩者占有整個計算時間的比例，也由於所有節點皆透過網路連接，因此通訊成本成為系統最大的考量。所以我們修改實驗環境，納入了多台不同網域的電腦節點，讓網路傳輸的特性突顯出來，節點連接示意圖如圖 4.5 所示。

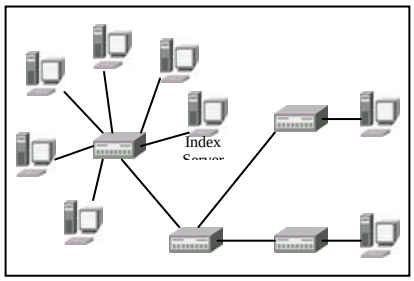

圖 4.5 節點配置示意圖

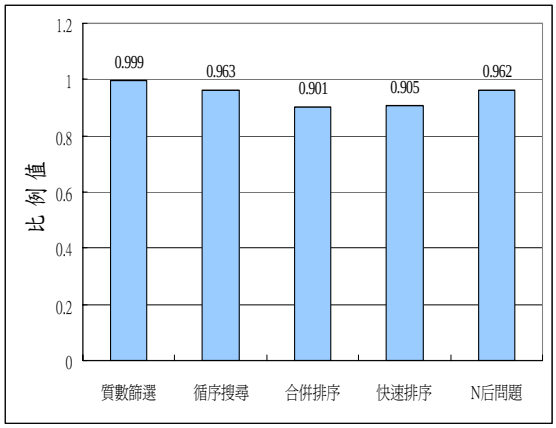

圖 4.6 速度提升變化比例圖

透過這樣的網路節點分配環境，重新測試整個系統的效能。從圖 4.6 可以發現，當納入了不同網域的電腦之後，所有的平行程式執行的效能皆有下降的情況，速度提升值只有原本的 0.999~0.901 倍。

因為加入了不同網域間的電腦，使得整個對等式環境下，節點進行通訊傳輸時的成本提升，因此我們進一步測試了網路環境通訊延遲(Commutation Latency)的情況，如圖 4.7 所示。

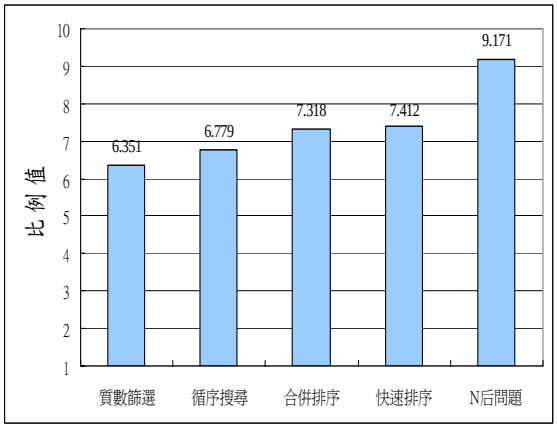

圖 4.7 通訊延遲變化比例圖

由於各個平行程式通訊量的不同，通訊延遲的時間上升到理想環境下測試結果的 6.351~9.171 倍不等。接著我們把圖 4.6 跟圖 4.7 做一個對應比較，前四個平行程式在通訊延遲上升的影響下，導致整個計算的速度提升效能呈現下降的情況，比較特別的是 N 后問題，在通訊延遲上升的影響下，速度提升效能值卻比前四個程式的效能值還高，因此我們進一步考慮了程式中計算時間與通訊時間兩者的比例(computation/commutation ratio)，如圖 4.8 所示。

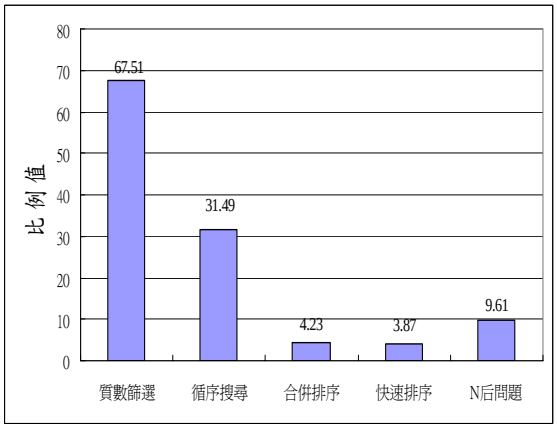

圖 4.8 計算時間與通訊時間比例圖

從這樣的結果可以發現，N 后問題的計算時間與通訊時間比例值，比合併排序及快速排序問題的比例皆高出許多，這也直接說明了一個事實，雖然 N 后問題具備較高的通訊延遲，但也因其具備龐大的計算時間，讓通訊延遲造成的效能影響幅度較低。但在這裡，並不考慮與質數篩選及循序搜尋進行比較的動作，從表 4.2 的程式架構比較中，這兩

者在通訊次數與通訊量上，與 N 后問題的性質差異較大，但也反應出一個情況，質數篩選及循序搜尋因為計算時間與通訊時間比例值較高的緣故，是比較適合利用這樣對等式環境來進行平行計算的工作，藉此提高效能，減少計算的時間。

從整個對等式平行計算架構來看，決定整個系統計算效能的因素大致上可分為計算方法複雜度與網路傳輸速度。計算方法複雜度影響了資料物件切割區塊大小；網路傳輸速度除了影響資料物件切割區塊大小，也影響了物件分配量大小，而物件區塊大小及物件分配量大小決定了最終整個通訊延遲的時間長短，讓計算的效能提升值受限於此因素。因此整個對等式網路的連接，必須仔細評估網路傳輸速度的快慢及網路通訊的穩定性，這兩者是關係到對等式平行計算中計算速度能否提升的關鍵因素，並讓平行計算的價值完全發揮出來，達成高效能的目標。

五、結論

在本論文中，我們利用對等式網路的環境，建立了一個平行計算的系統環境，藉由相關計算理論的分析與系統建置，透過 UML 規劃出系統的藍圖，進而使用 Java 程式語言開發出一個具高效能的對等式平行計算中介軟體系統，使得在開放性的網路下，依然能夠有效利用電腦資源，提高計算處理的速度。同時，透過需求性的動態電腦節點配置，達到虛擬平行電腦叢集的組合。

從測試數據中得知，利用 7 台電腦網路節點實際進行平行計算，經測試結果顯示，每個平行程式皆有至少五倍的速度提升值，但若能再更深入的了解演算法程式的資料結構和流程，相信有可能找出更好的平行化方法。因此在到達參與節點上限前，應該能讓計算速度再提升，增加運算的效能。

六、參考文獻

[1] 李卓俊，鄭博文，2004，「對等式(P2P)網路技術」，臺灣通訊雜誌，pp.40。

[2] Abbas, A., Grid computing：A Practical Guide to Technology and Applications, Charles-River Media, USA, January, 2004.

[3] Foster, I., Design and Building Parallel Programs：Concepts and Tools for Parallel Software Engineering, Addison-Wesley, USA, 1995.

[4] Gustafson, J.L., “Reevaluating Amdahl’s Law,” Communications of the ACM, May 1988, Volume 31, Issue 5, pp.532-533.

[5] Harold, E.R., Java Network Programming, O’Reilly, June, 2000

[6] Kai, H. and Zhiwei, X., Scalable Parallel Computing: Technology, Architecture, Programming, McGraw Hill, USA, 1998.

[7] Leuf, B., Peer-to-Peer : Collaboration and Sharing over the Internet, Addison-Wesley, USA, 2002.

[8] Martin, B.E. and Pedersen, C.H., “An object-based taxonomy for distributed computing systems,” IEEE Computer, 1991, Volume 24, Issue 8, pp.17-27

[9] Mattson, T.G., Sanders, B.A., Massingill, B.L., Patterns for Parallel Programming, Addison-Wesley, USA, 2005.

[10] Michael, J.Q., Parallel Programming in C with MPI and OpenMP, First Edition, McGraw-Hill, USA, 2004.

[11] Schollmeier, R., ”A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications,” Peer-to-Peer Computing, 2001, Proceedings. First International Conference, 27-29 Aug. 2001, pp.101-102.

[12] Wilkinson, B. and Allen, M., Parallel Programming ： Techniques and Applications Using Networked workstation and Parallel Computer, Second Edition, Prentice-Hall, USA, 2005.

[13] A Sun Developer Network Site, ,Core Java, Java Remote Method Invocation (Java RMI) Reference, http://java.sun.com/products/jdk/rmi/reference/docs/index.html.