行政院國家科學委員會專題研究計畫 成果報告

應用於格網與可擴充平行系統之通訊最佳化，廣播與品質服務排程技術之研究(第 3 年)
研究成果報告(完整版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 97-2221-E-216-011-MY3
執 行 期 間 ：99 年 08 月 01 日至 100 年 07 月 31 日
執 行 單 位 ：中華大學資訊工程學系

計 畫 主 持 人：許慶賢

計畫參與人員：碩士班研究生-兼任助理人員：張弘裕
碩士班研究生-兼任助理人員：蔡宗輝
碩士班研究生-兼任助理人員：朱元佑

處 理 方 式 ：本計畫涉及專利或其他智慧財產權，2 年後可公開查詢

中 華 民 國 100 年 10 月 29 日

行政院國家科學委員會補助專題研究計畫 ■成果報告 □期中進度報告

※※※※※※※※※※※※※※※※※※※※※※※※※※
※ 應用於格網與可擴充平行系統之通訊最佳化 ※
※ 廣播與品質服務排程技術之研究(3/3) ※
※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：☑個別型計畫 □整合型計畫
計畫編號：NSC 97-2221-E-216-011-MY3
執行期間：99 年 08 月 01 日至 100 年 07 月 31 日
執行單位：中華大學資訊工程學系

計畫主持人：許慶賢 中華大學資訊工程學系教授
共同主持人：
計畫參與人員： 陳世璋、陳泰龍（中華大學工程科學研究所博士生）
徐一中、陳柏宇、張弘裕、蔡宗輝、朱元佑
（中華大學資訊工程學系研究生）

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年☑二年後可公開查詢

中 華 民 國 100 年 10 月 28 日

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告 □期中進度報告

應用於格網與可擴充平行系統之通訊最佳化廣播與品質服務排程技術之研究(3/3)

計畫類別：☑個別型計畫　□整合型計畫
計畫編號：NSC 97-2221-E-216-011-MY3
執行期間：99 年 08 月 01 日至 100 年 07 月 31 日

計畫主持人：許慶賢　中華大學資訊工程學系教授
共同主持人：
計畫參與人員：　陳世璋、陳泰龍（中華大學工程科學研究所博士生）
徐一中、陳柏宇、張弘裕、蔡宗輝、朱元佑
（中華大學資訊工程學系研究生）

成果報告類型(依經費核定清單規定繳交)：□精簡報告　☑完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
☑出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年☑二年後可公開查詢

執行單位：中華大學資訊工程學系

中　華　民　國　100　年　10　月　28　日

目錄

行政院國家科學委員會專題研究計畫成果報告

應用於格網與可擴充平行系統之通訊最佳化廣播與品質服務排程技術之研究

計畫編號：NSC 97-2221-E-216-011-MY3
執行期限：99 年 8 月 1 日至 100 年 7 月 31 日
主持人：許慶賢　中華大學資訊工程學系教授

計畫參與人員：　陳世璋、陳泰龍（中華大學工程科學研究所博士生）
徐一中、陳柏宇、張弘裕、蔡宗輝、朱元佑
（中華大學資訊工程學系研究生）

一、中文摘要

本計劃是有關應用在格網環境中可擴充平行系統的通訊、廣播與品質服務排程技術之最佳化。本計畫預計執行期限為三年，第一年，我們開發適用於相同叢集網格架構的特殊型通訊局部化資料分割技術以及邏輯處理器與資料之間對應的技術。並且提出適用於不同叢集網格架構的通用型通訊局部化資料分割技術。利用 Hungarian method 根據已建立的動態評估外部通訊效能的模式在不同數量的處理器組合下，找出最好的資料分割方式。第二年，在廣播演算法的設計中，我們分為以圖為基礎與樹狀結構為基礎的演算法，在樹為基礎的演算法當中，我們規劃運用不同的演算法來改良，為了避免圖形中產生迴圈造成重複資料的傳遞，我們設計一個預先排程系統來解決。在高品質服務工作排程系統下，基於傳統的 Min-Min 品質需求排程的基礎，我們提出有效的重新排程演算法來改善減少工作時間或減少資源的使用量，為資源最佳化重新排程。第三年，結合服務計算的概念，我們提出以品質服務為基礎的資源與工作排程技術，包括資源成本(Economic Guided)與時間成本(Performance Guided)最佳化的技術。基於傳統的 Min-Min、Max-Min 排程，我們也提出二階段、或多階段的重新排程方法，藉以改善資源的使用效率以及滿足使用者的不同需求。結合格網經濟模型，這一個部份的研發成果，也導入實際的格網系統進行實驗與部屬。本計畫預計完成的研究項目，皆已經實作出來，並在相關期刊與研討會發表。

關鍵詞：格網計算、可擴充平行系統、服務式計算、可擴充計算、通訊最佳化、廣播演算法、P2P 技術、品質服務排程、異質性計算

二、緣由與目的

可擴充系統(Scalable System)、如多叢集系統、格網系統，已成為新的發展趨勢與廣泛被接受的計算平台。其中格網系統(Grid system)主要是整合了不同區域和系統的運算資源，建立一個虛擬並且擁有高度擴充性的資訊系統，用來服務各種不同層面的應用，包括科學運算、個人服務、社區或企業解決方案等。叢集式格網系統(Cluster grid)就是計算格網 (Computing grid system)的一個典型的例子。

如同分散式記憶體環境，在格網系統上執行應用程式，某些計算節點可能會因應程式的需要，透過網路傳送資料給其他的電腦節點。因此，在格網系統下要有效率的執行一個應用程式，資料的佈署位置往往是降低通訊成本最主要的關鍵。包括靜態資料儲存的位置、與程式執行期間的動態資料配置。這兩者這都是影響程式效能主要的關鍵。過去，有許多研究致力於平行系統的通訊最佳化技術。然而，由於格網系統的網路是動態的、平台是異質的、拓樸是多變的；加上應用領域的不同，過去的研究成果並無法直接套用在這樣的計算架構上。也因此，有越來越多的研究探討在這種可擴充式計算(Scalable Computing) 架構下的通訊與排程技術。

有效的廣播排程，可以避免通訊競爭的產生，並且在最短排程步驟內完成外部的訊息傳遞。有別於傳統的排程策略，我們建立多個適應於不同網路拓樸的生成樹，可以依據資訊的來源，選擇最適合的生成樹執行廣播排程。除此之外，亦根據網路中工作站的效能，再將所建構出來的排程做最佳化的調整。為了評估所提出來的排程技術效能，我們實作了演算法與其他廣播演算法，做效能上的比較。實驗結果顯示，在不同的網路架構之下，表現出相當不錯的效果，在高度異質性的網路環境，可以有更明顯的改善。

網格計算闡述一個簡單的概念，使用網格計算能夠整合伺服器、存儲系統以及網路，使之成為一個很大的高性能服務系統。其專門針對複雜科學計算的計算模式，這種計算模式是利用網際網路把分散在不同地理位置的電腦組織成一個虛擬的超級計算機。每個叢集都是分散在異質性的網路上，隨著網格計算的發展，在處理資料的分配上必須有更快且有效的方法讓平行應用程式可獲取更高的執行效率。

另外，在高品質服務的工作排程技術方面，傳統的 Greedy、Min-min、Max-min 工作排程演算法，無法套用在服務導向架構(Service Oriented Architecture)的系統中。主要的原因是某些應用程式可能會有特殊服務品質要求(QoS)。利用傳統的資料排程方法，除

了無法滿足有特殊需求的工作內容，其整體系統效能也會因此大打折扣。有鑑於此，因應新世代的服務式計算(Service Computing)與服務導向架構(Service Oriented Architecture)，在計算格網系統上的工作排程技術，勢必需要加入可以滿足這樣目標的技術，這也是目前及待解決與改良的問題。因為這些研究的方向使得格網的高效能運算技術有更多的發展空間與更多的研究方向。

如同前面所提到的，雖然有許多過去的研究都在探討這些問題，但是我們所要解決與提出的是，適應於動態格網架構，並且以服務導向架構(SOA)為基礎，提出滿足使用者服務品質(QoS)的解決方案。

鑒於可擴充計算(Scalable Computing)已經成為新的計算模式。不論是多叢集系統，格網系統，異質性網路系統，網路通訊則是這些系統共同倚賴的基本要素。因此，在可擴充式計算系統，降低網路通訊成本往往是許多研究的第一考量。從演算法的角度來說，最基本且重要的通訊機制就是訊息的廣播；而從資源管理與服務導向計算的角度而言，有效運用系統資源與工作排程的相關技術，最受到重視。因應新世代的服務式計算(Service Computing)與服務導向架構(Service Oriented Architecture)，在本計畫中，我們將研發多個應用於格網與可擴充平行系統之通訊最佳化、廣播與以品質服務為基礎的資源與工作排程技術。本計畫有三個主要的研究課題：一、發展適應於多叢集系統下之通訊最佳化技術；利用多叢集系統下之處理器重新排序技術改善排程工作的結果，此項成果可以直接應用在異質性格網系統。二、發展應用於異質性網路的訊息廣播技術；根據不同的網路拓樸，採用適當的排程策略，並且開發最佳化的評估模組，以實際的 work-load 分析效能。三、發產以品質服務為基礎的資源與工作排程技術，結合格網經濟模型，將研發成果導入實際的格網系統。

三、研究方法與成果

在發展應用於格網與可擴充平行系統之通訊最佳化、廣播與品質服務排程技術之研究，我們的工作主要包含以下研究課題：

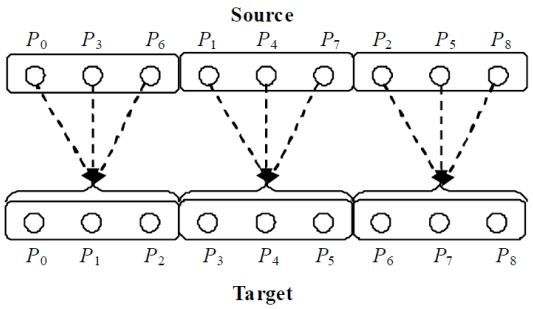
a. 多叢集系統通訊最佳化與動態資料配置
b. 最佳化叢集式格網拓墣研究
c. 適應於動態異質網路之訊息廣播技術

d. 以品質服務為基礎的資源與工作排程

e. 整合格網經濟模型與網頁服務

3.1 多叢集系統通訊最佳化與動態資料配置

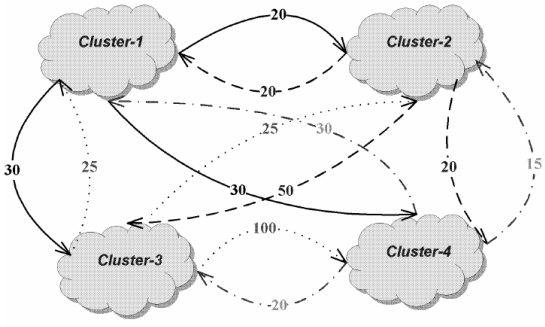
此項研究的主要目的是，為了減少資料透過網路在不同叢集系統之間傳輸，提高資料的局部性，使資料傳輸儘可能發生在同網域的部分，以降低網路的通訊成本。如圖一，來源端(Source)與目的端(Target)代表三個電腦叢集系統，每個叢集都有三個節點進行科學運算，為了減少通訊成本，降低傳輸資料透過網際網路的量，他們重新排序了節點的邏輯序號。原本要跨叢集的通訊變成不需要跨越叢集，通訊仍在本地端進行。當然，我們必須處理更一般化的例子，因為叢集系統往往是大小不一。對此，我們可以分成兩種情況討論。一種是對於特殊叢集格網拓墣，利用數學公式找出最佳化的資料切割；另一種則是針對一般型的格網拓墣，提出節點取代(Node Replacement)的策略。



圖一　　通訊局部化示意圖

另外，異質性格網系統不同的地方是每個電腦叢集可能擁有不同的計算節點，包含計算能力與頻寬限制。對新的計算環境，我們需要重新開發一套資料傳輸模組，針對異質性環境，包含不同大小的資料量和不同頻寬的流量限制，提供異質電腦叢集之間的通訊成本對照表，異質性網路頻寬其示意圖如圖二。

此外，我們亦提出在異質性多叢集網路下的兩種資料配置的方法用於執行平行應用程式時的資料分配。我們提出這兩個有效的資料分配方法應用在當執行平行應用程式時能夠讓資料局部化可以減少 cluster 的通訊成本。基於通訊成本的考量下，利用簡單的邏輯對應技術，使得在格網環境下所執行的平行應用程式獲得有效率的執行來提高效能。叢集格網是一個典型的由數個分散到各處的 PC 所組成的格網運算環境。圖三顯示不同叢集格網傳輸成本數值，與傳輸成本最佳化的例子。



圖二　多叢集之異質性網路頻寬

		cluster			
Trad.	BLOCK	C_1	C_2	C_3	C_4
P_0	A	60	40	125	105
P_1	B	150	220	100	80
P_2	C	120	100	425	30
P_3	D	90	70	100	95
P_4	E	150	190	200	60
P_5	F	90	100	350	60
P_6	G	120	100	75	85
P_7	H	150	160	300	40
P_8	I	80	80	275	75
P_9	J	130	150	50	90
P_{10}	K	150	130	400	20
P_{11}	L	70	60	200	90
P_{12}	M	140	200	25	95
P_{13}	N	150	100	500	0

(a)

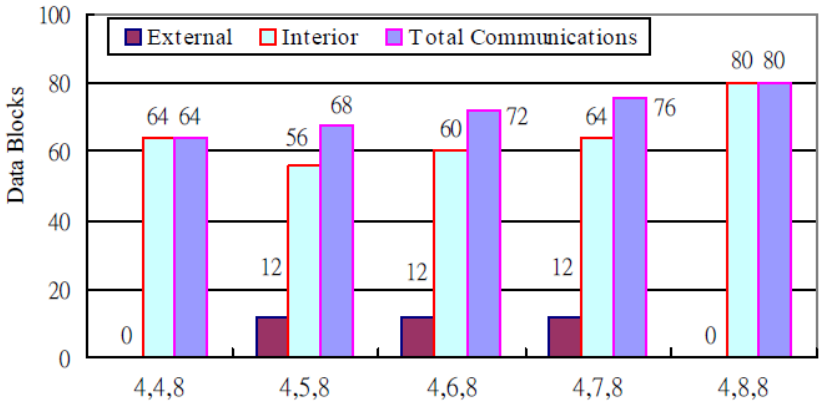
		cluster			
Trad.	BLOCK	C_1	C_2	C_3	C_4
P_0	A	60	40	125	105
P_5	F	90	100	350	60
P_8	I	80	80	275	75
P_{11}	L	70	60	200	90

(b)

圖三　(a) 不同叢集格網傳輸成本數值 (b) 傳輸成本最佳化數值

我們所提出的資料分配技術，目的是為了降低通訊的成本並加速平行程式在處理資料時所花費的時間，這個理論是運用在不同通訊成本所組成的環境上。由於我們的理論分析與程式模擬在不同通訊成本下所執行的結果都比原先未經過處理器重排的配置方法明顯有顯著的改善，並且優於在未考慮通訊成本情況下的配置方法。顯示這兩

種資料配置方式是可達到降低通訊成本的目的以及增進平行程式執行的效率。圖四顯示出不同叢集格網拓墣之通訊成本比較。



圖四　　不同叢集格網拓墣之通訊成本比較

3.2　異質網路訊息廣播技術

根據前一部分在通訊最佳化的研究成果，我們進一步發展一般化的通訊最佳化技術，應用在異質性網路的訊息廣播技術。根據一般常用的多埠(Switch-Based)通訊模式，我們提出適應於樹狀結構或一般網路架構下的訊息廣播技術。在格網系統中，節點之間的通訊藉由異質的網路頻寬來廣播與傳送，我們利用以樹狀圖形為基礎的網路架構，設計一個廣播機制，使邏輯節點透過最佳的廣播技術，將資料分佈在適當的電腦，並且使得所需廣播的資訊只經過最少的路徑、避免訊息在網路中造成迴圈、以及橋接器產生訊息轉送過程的衝突。

由於等待廣播的資料是存放在網路節點上，當使用者執行資訊廣播的期間，其餘網路節點必須按照廣播起始節點順序來讀取。在這種情形之下，廣播排程有規律性的週期分配，網路上廣播排程的運算負荷將會減少許多。為了減少這種龐大廣播運算成本，有幾個重要的課題是我們需要考慮的：

1. 挑選適當的演算法減少廣播路徑選擇的時間
2. 減少網路頻寬的佔用
3. 降低網路通訊的成本

我們利用更簡單的廣播排程演算法，設計出更有效率的廣播路徑選擇，除了同樣可以減少複雜的路徑運算之外，更可以縮短廣播時間。

以網路交換器為基礎的多埠網路通訊模式，如異質性網路工作站 Heterogeneous Networks Of Workstations (HNOW)的通訊架構模式；圖六所顯示的 HNOW 網路模式應用

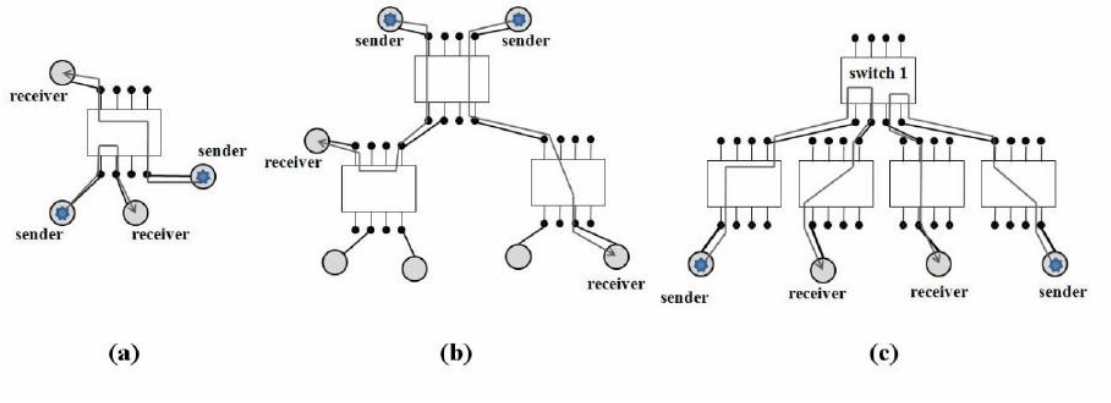
於訊息廣播上較為複雜的排程模式，此架構主要由 Switch、Workstation、Bidirectional Link 組成。分為以下三種情形：

一、　所有的傳送端與接收端皆連接於同一個橋接器(Switch)，如圖五(a)

二、　所有的傳送端連接於相同的橋接器，但是接收端連接相對於傳送端在不同的橋接器上，如圖五(b)

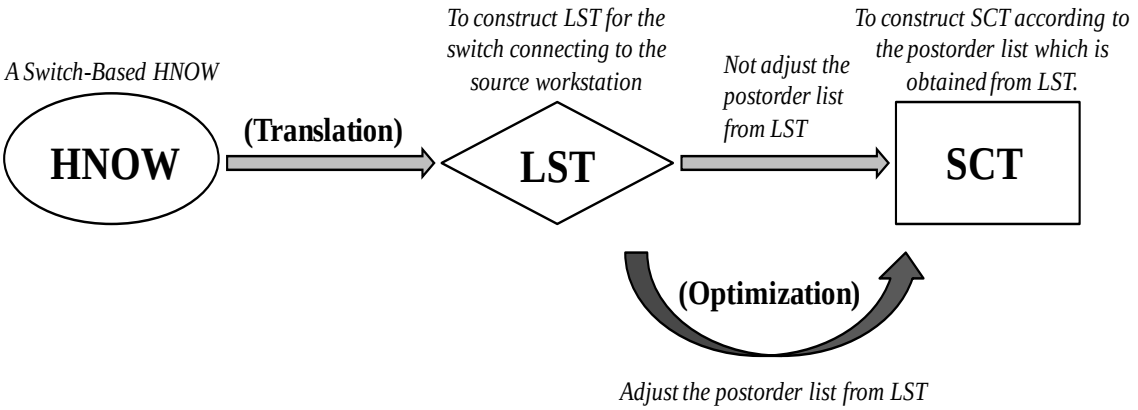
三、　每個傳送端與相對應的接收端接連接於不同的橋接器上，如圖五(c)

對於資訊廣播的排程系統下，在程式執行的各個階段中，所有節點皆同時可廣播與接收訊息。這種方法有時候會產生訊息碰撞所耗費的時間。



圖五　異質性網路工作站(HNOW)架構傳送端與接收端相關位置

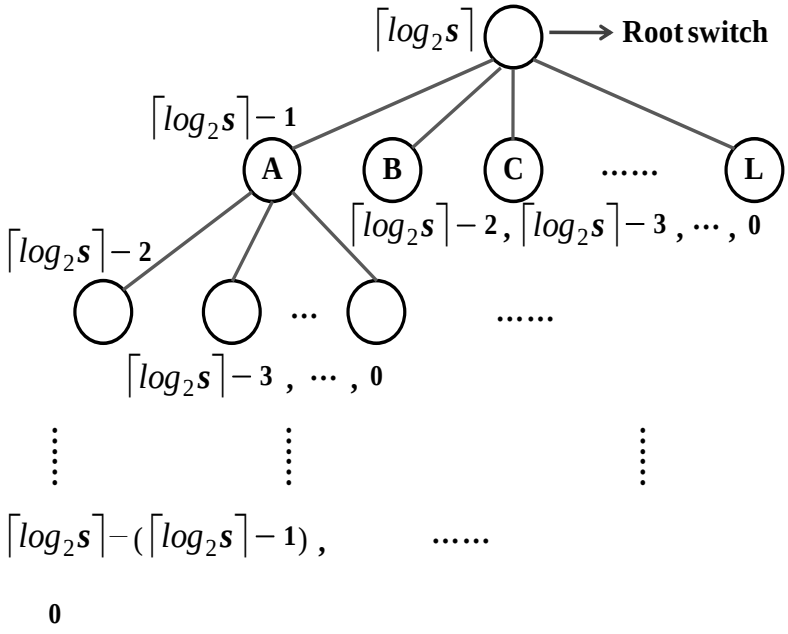
針對 HNOW 的網路架構下，我們開發新的方法 Location Aware Broadcast Scheme (LABS)，提昇網路通訊時的頻頻壅塞問題與降低檔案廣播至錯誤路徑的機率。針對不同的網路結構，我們考量節點數量 s，限制樹狀結構的寬度與高度，避免訊息廣播時的廣播風爆與衝撞問題的產生，在各個節點建立不同的廣播路徑樹狀結構 Location Oriented Spanning Tree (LST)，與其它相關方法比較其執行上的效能，並且執行效能的評估與測試，利用不同的工具偵測網路當時的效能，再根據所測得的數據給予各個網域不同的權重值，決定工作與資料廣播的方式。對於分散式訊息廣播的技術，整合出一套完整的資料配置技術。



圖六　　LABS 演算法流程示意圖

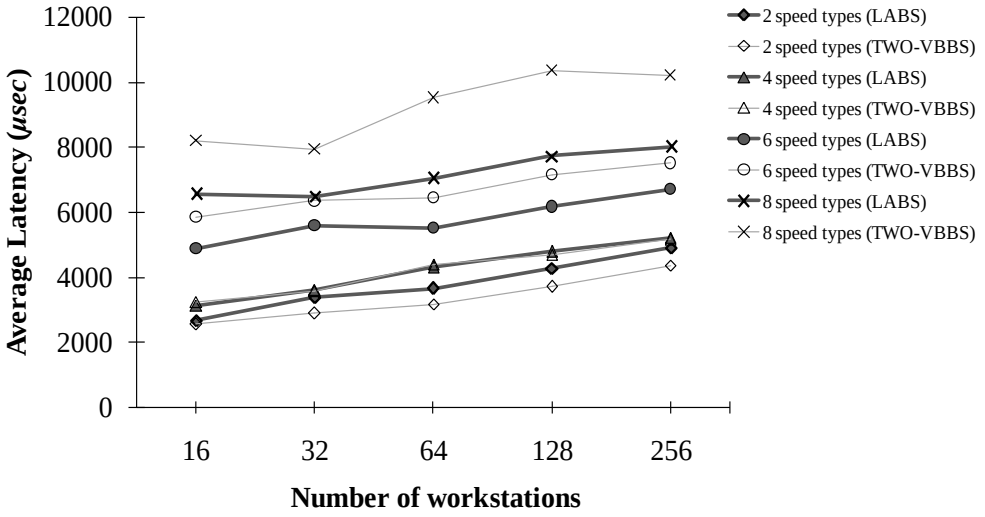
對於所設計的 LABS 排程演算法，利用處理器與訊息傳送、接收的關係，可以容易的找出通訊時最少步驟。演算法主要由圖形中的最 Binomial-Like Tree 所組成(如圖七)：主要將訊息依照節點前後順序排好，在傳送訊息的部分，是依據 Binomial-Like Tree 排好的節點順序，將訊息一個一個傳送。

我們所設計的演算法將對於訊息廣播排程的缺點與網路頻寬的浪費加以節省，原則是處理器和訊息的關係依照不會衝突的原則來建樹設計演算法，主要步驟依據網路節點的子樹大小與運算權重的大小排序，接著將其訊息順序由來源端送到目的端網路節點。我們也由數據證明所設計的 LABS 演算法之效能比其他方法有更短的網路延遲時間。

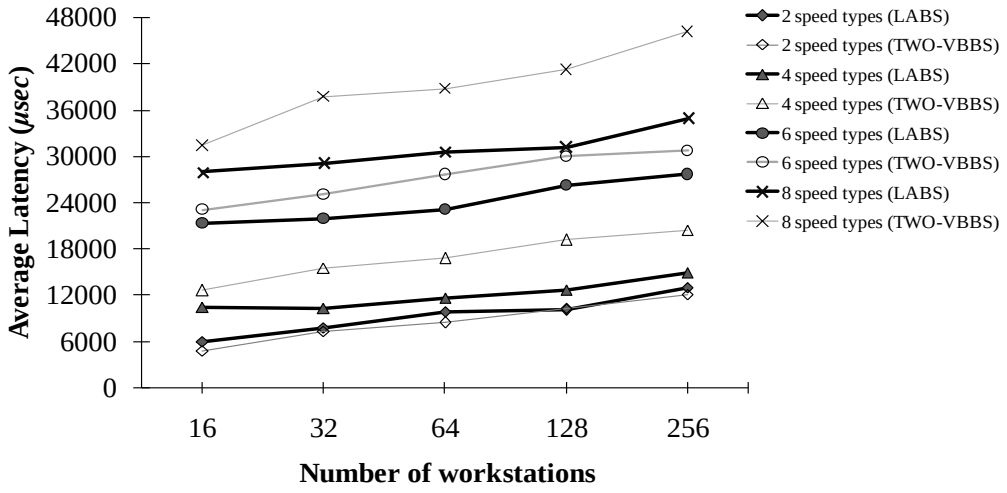


圖七　　Location Oriented Spanning Tree (LST) 結構

圖八為資料量訊息較小(2048 *flits*)時所產生的數據。圖九為資料量訊息較大(10240 *flits*) 時所產生的數據比較。由數據可以得知在不同的網路速度下(2~8 Speed Types)，LABS 都比相關的其他方法有更少的網路訊息傳送延遲，提升網路使用效率。



圖八 HNOW 網路資料傳送數據比較(2048 *flits*)



圖九 HNOW 網路資料傳送數據比較(10240 *flits*)

3.3 以品質服務為基礎的資源與工作排程

傳統的 Greedy、Min-min、Max-min、STA、STP、MTP 工作排程演算法，無法套用在服務導向架構(Service Oriented Architecture)的系統中。我們開發以品質服務為基礎之時間成本最佳化排程技術(Makespan Optimization Rescheduling)與品質服務為基礎之資源成本最佳化排程技術(Resource Optimization Rescheduling)。

實施的方法上，我們首先發展兩種基本類策略，一是工作時間最佳化重新排程(MOR)

之測試，主要是利用將負載較重的節點上之工作移到其他閒置節點來降低工作時間二是資源最佳化重新排程(ROR)測試，應用組合型程式分析，針對資料的部分，我們主要設計將具有最小工作數量的節點上的工作，移動到其他雖然正在工作中、但尚有閒置空間的處理器上，用以減少處理器資源的使用量，進而空出節點來提供其他運算工作的提前執行。

MOR 演算法，我們主要區分為兩個部份：一、資料分配相依於 QoS 標準配置技術；二、依照比例將負載較重的運算節點的資料移至負載較輕的節點。

針對第一部份，我們將 QoS 為導向的演算法所計算出最佳的排程，再根據 QoS 所產生出來的資料與處理器對應架構，在第二部份可以有效率將先前的資料配置計算出負載較重的處理器，將較小的工作移動至負載較輕的節點進行運算，其優點除了能減少資料重複傳輸的時間之外，進而能降低整體系統程式執行時間。

表一為 MOR 演算法之模擬效能對照表，分別比較了 Min-Min、QoS Guided Min-Min、MOR 與 Improved Ratio (MOR 比 QoS Guided Min-Min)。

表一　　MOR 演算法之效能比較表

(a) (N_R=50, Q_R=30%, Q_T=20%, H_T=1, H_C=1)

Task Number (N_T)	200	300	400	500	600
Min-Min	978.2	1299.7	1631.8	1954.6	2287.8
QoS Guided Min-Min	694.6	917.8	1119.4	1359.9	1560.1
MOR	597.3	815.5	1017.7	1254.8	1458.3
Improved Ratio	14.01%	11.15%	9.08%	7.73%	6.53%

(b) (N_T=500, Q_R=30%, Q_T=20%, H_T=1, H_C=1)

Resource Number (N_R)	50	70	90	110	130
Min-Min	1931.5	1432.2	1102.1	985.3	874.2
QoS Guided Min-Min	1355.7	938.6	724.4	590.6	508.7
MOR	1252.6	840.8	633.7	506.2	429.4
Improved Ratio	7.60%	10.42%	12.52%	14.30%	15.58%

(c) (N_T=300, N_R=50, Q_T=20%, H_T=1, H_C=1)

Q_R%	15%	30%	45%	60%	75%
Min-Min	2470.8	1319.4	888.2	777.6	650.1
QoS Guided Min-Min	1875.9	913.6	596.1	463.8	376.4
MOR	1767.3	810.4	503.5	394.3	339.0
Improved Ratio	5.79%	11.30%	15.54%	14.99%	9.94%

(d) (N_T=300, N_R=50, Q_R=40%, H_T=1, H_Q=1)

Q_T%	15%	30%	45%	60%	75%
Min-Min	879.9	1380.2	1801.8	2217.0	2610.1
QoS Guided Min-Min	558.4	915.9	1245.2	1580.3	1900.6
MOR	474.2	817.1	1145.1	1478.5	1800.1
Improved Ratio	15.07%	10.79%	8.04%	6.44%	5.29%

(e) (N_T=500, N_R=50, Q_R=30%, Q_T=20%, H_Q=1)

H_T	1	3	5	7	9
Min-Min	1891.9	1945.1	1944.6	1926.1	1940.1
QoS Guided Min-Min	1356.0	1346.4	1346.4	1354.9	1357.3
MOR	1251.7	1241.4	1244.3	1252.0	1254.2
Improved Ratio	7.69%	7.80%	7.58%	7.59%	7.59%

(f) (N_T=500, N_R=50, Q_R=30%, Q_T=20%, H_T=1)

H_Q	3	5	7	9	11
Min-Min	1392.4	1553.9	1724.9	1871.7	2037.8
QoS Guided Min-Min	867.5	1007.8	1148.2	1273.2	1423.1
MOR	822.4	936.2	1056.7	1174.3	1316.7
Improved Ratio	5.20%	7.11%	7.97%	7.77%	7.48%

使用 ROR 演算法的條件時必須當某計算節點只分配到微小的工作而整體工作時間與 MOR 演算法相等時，幫助排程系統快速得到一組重配置的方法，所須使用的運算節點需搜尋該區域網格環境中所使用到的處理器個數，當所有的處理器分配到該工作執行後不影響整體工作時間時，即完成所有動作。表二為 ROR 演算法之模擬效能對照表。

表二　　ROR 演算法之效能比較表

(a) (N_R=100, Q_R=30%, Q_T=20%, H_T=1, H_Q=1)

Task Number (N_T)	200	300	400	500	600
QoS Guided Min-Min	100	100	100	100	100
ROR	39.81	44.18	46.97	49.59	51.17
Improved Ratio	60.19%	55.82%	53.03%	50.41%	48.83%

(b) (N_T=500, Q_R=30%, Q_T=20%, H_T=1, H_Q=1)

Resource Number (N_R)	50	70	90	110	130
QoS Guided Min-Min	50	70	90	110	130
ROR	26.04	35.21	43.65	50.79	58.15
Improved Ratio	47.92%	49.70%	51.50%	53.83%	55.27%

(c) (N_T=500, N_R=50, Q_T=20%, H_T=1, H_Q=1)

Q_R%	15%	30%	45%	60%	75%
QoS Guided Min-Min	50	50	50	50	50
ROR	14.61	25.94	35.12	40.18	46.5
Improved Ratio	70.78%	48.12%	29.76%	19.64%	7.00%

(d) (N_T=500, N_R=100, Q_R=40%, H_T=1, H_Q=1)

Q_T%	15%	30%	45%	60%	75%
QOS Guided Min-Min	100	100	100	100	100
ROR	57.74	52.9	48.54	44.71	41.49
Improved Ratio	42.26%	47.10%	51.46%	55.29%	58.51%

(e) (N_T=500, N_R=100, Q_R=30%, Q_T=20%, H_Q=1)

H_T	1	3	5	7	9
QOS Guided Min-Min	100	100	100	100	100
ROR	47.86	47.51	47.62	47.61	47.28
Improved Ratio	52.14%	52.49%	52.38%	52.39%	52.72%

(f) (N_T=500, N_R=100, Q_R=30%, Q_T=20%, H_T=1)

H_Q	3	5	7	9	11
QOS Guided Min-Min	100	100	100	100	100
ROR	54.61	52.01	50.64	48.18	46.53
Improved Ratio	45.39%	47.99%	49.36%	51.82%	53.47%

以品質服務(QOS)為導向的工作排程演算法以下面的為主：

for all tasks *ti* in meta-task *Mv* (in an arbitrary order)
    for all hosts m_j (in a fixed arbitrary order)
        CTij = etij + dtj
    end for
end for
do until all tasks with QoS request in *Mv* are mapped
for each task with high QoS in *Mv*,
find a host in the QoS qualified host set that obtains the earliest completion time
    end for
    find the task *tk* with the minimum earliest completion time
    assign task *tk* to the host *ml* that gives it the earliest completion time
    delete task *tk* from *Mv*
    update *dtl*
    update *CTil* for all *i*
end do
do until all tasks with non-QoS request in *Mv* are mapped
    for each task in *Mv*

find the earliest completion time and the corresponding host
end for
find the task *tk* with the minimum earliest completion time
assign task *tk* to the host *ml* that gives it the earliest completion time
delete task *tk* from *Mv*
update *dtl*
update *CTil* for all *i*
end do

其中，*CT* : Completion time … …….(系統完成時間)
dt : delay time ……..(網路延遲時間)
et : execute time ……...(單一工作執行時間)
i : the job ID …….(單位工作編號)
j: the machine ID …….(處理器編號)

以工作時間最佳化重新排程 Makespan Optimization Rescheduling (MOR)的工作排程演算法如下：

QOS guided scheduling algorithm …..
for CT_j in all machines
find out the machine with maximum makespan CT_{max} and set it to be the standard
end for
do until no job can be rescheduled
for job *i* in the found machine with CT_{max}
for all machine *j*
according to the job's QOS demand, find the adaptive machine *j*
if the execute time of job *i* in machine *j* + the CT_j < makespan
rescheduling the job *i* to machine *j*
update the CT_j and CT_{max}
exit for
end if
next for
if the job *i* can be reschedule
find out the new machine with maximum CT_{max}
exit for
end if
next for
end do

以運算節點資源最佳化重新排程的 Rseource Optimization Rescheduling (ROR)的工作排程演算法如下：

QOS guided scheduling algorithm …..
for *m* in all machines
find out the machine *m* with minimum count of jobs
end for

do until no job can be rescheduled
for job *i* in the found machine with minimum count of jobs
for all machine *j*
according to the job's QOS demand, find the adaptive machine *j*
if the execute time of job *i* in machine *j* + the CT_j <= makespan CT_{max}
rescheduling the job *i* to machine *j*
update the CT_j
update the count of jobs in machine *m* and machine *j*
exit for
end if
next for
next for
end do

四、結論與討論

整合格網系統下的訊息廣播演算法與品質服務工作排程技術，我們開發以網頁為基礎的工作排程選擇器。使用者可以依據個別需要選擇排程系統以及分配的狀態，設定不同廣播區域的範圍，進而傳送資料至可以提供運算的節點，讓程式在格網下執行更具彈性。整個計畫的最終目的在於，整合叢集式格網系統的通訊最佳化技術、資訊廣播系統、與品質服務工作排程系統，設計易於操作的使用者介面，並結合格網經濟模型的機制，導入實際的格網系統。

此外在校園學術網路上，我們進行系統平台的大量部署，利用校園內的分散硬碟空間，成功建置大型的資料格網系統。另外，我們進行資料領域特定語言的設計與各種平台函式庫開發，並且在學術網路上進行建置與部署並且測試其效能。我們也與其他學校進行緊密的整合，並進行許多細節的修正。另外，建構以服務為導向的格網經濟模型，應用於各種 Web 服務，進而滿足不同使用者的需求。格網經濟模型研究的重點在於同時考量供給者的維運成本與滿足消費者的不同需求(QoS)，發展一套未來可以導入企業網路的格網經濟架構。

我們改良格網系統資料處理模型，發展能夠大量處理資料的應用程式介面，利用資料格網的管理分析工具，瞭解資料的分佈及網路，運算資源的狀態，整理出資料的分佈模式。讓資料能夠運用其存放節點的運算資源，讓每筆資料不需要透過集中式的運算，達成輕鬆地移動或複寫自己而提升整體資料的容錯率。此外，我們也實做出分散式中介資料服務，高輸出虛擬檔案系統。我們的中介軟體顯示了在節點

數量，檔案數量成長時，也有容錯的特性。本計畫預計完成的研究項目，皆已經實作出來，並在相關期刊與研討會發表。

本子計畫所開發的資源與工作排程系統，相較於現有的系統，採用以服務導向架構為基礎的概念，具有設計上的彈性、與系統的輕量化，可以滿足不同使用者的需求(Quality of Service)、以及高執行效率的特性。此架構具創新性；開發的元件都符合可擴充式計算(Scalable Computing)的架構，加上 Grid 技術日漸成熟，未來將我們所開發的技術與系統與既有的格網平台結合，亦是相當可行的作法。

在軟體開發上，我們使用軟體工程較新的敏捷(Agile)開發模式，並與開放原始碼社群整合。未來可以提供使用者一個安全、便利的格網平台，提供管理者一個集中式的管理介面，與提供開發者一個具高度擴充性及相容性的系統架構，使得未來管理的人力及時間成本大幅降低。從格網系統管理的角度來看，這也是本計劃的另一個創新與貢獻。

因應未來以服務為導向的資訊服務，我們結合格網經濟模型與 QoS 最佳化計算於 Web 服務中，讓格網系統有更實際的應用價值；在未來格網系統的普及與推廣工作上，也有很大的幫助。

服務計算結合 Grid 的研究還在起步階段。未來，我們將把研究重點放這一個部份，預期對格網技術會有重要的影響。下表整理出本計劃主要的貢獻。

	過去技術	完成計畫後狀況
技術面	● 通訊最佳化的研究集中在平行與分散式記憶體系統。 ● 訊息廣播技術、資源與工作排程技術，未考量格網動態的特性，經濟模型、服務與品質導向機制。	● 針對叢集式格網的動態特性，所提出來的解決方案更符合未來可擴充式計算的系統(Scalable Computing System)架構。可以容易導入不同的可擴充式系統。 ● 考量以服務導向架構與格網經濟模型，提出滿足使用者需求、與系統最佳化的資源與工作排程策略。就核心技術何言，此架構具創新性與可性的。由於 Grid 中介軟體技術日趨成熟，未來整合這些核心技術在既有的格網系統是指日可待的。
使用面	● 只有從事高性能計算、或相關應用領域的科學家會考慮使用格網系統。	● 讓格網系統平民化，滿足不同的使用者的使用情境。 ● 一般使用者 (未接觸過格網的使用者) - 能夠直接進入我們所設計好的 Portal，快速取得需要的服務。

系統面	● 以往採用網路式的關連式資料庫來建立索引或是儲存中介資料的資料格網。	● 改善舊有設計，而更符合格網概念要求 ● 採用了非網路式的資料庫，所有的節點都能夠快速地建立索引並且搜尋自己所負責的中介資料，而將通訊及快取的工作交給 XML 及 HTTP 來承擔。

下面我們歸納本計畫主要的成果:

- 完成同質性多叢集系統架構通訊最佳化
- 完成異質性多叢集系統架構通訊最佳化
- 完成叢集式格網拓墣最佳化
- 完成靜態格網拓墣資料佈署
- 完成動態格網拓墣資料佈署
- 完成多種樹狀架構廣播技術研究
- 完成通用型廣播演算法研究
- 完成訊息轉送表設計
- 完成以品質服務為基礎之資源成本最佳化排程技術
- 完成以品質服務為基礎之時間成本最佳化排程技術
- 完成二階段重排(re-scheduling)技術
- 完成與格網經濟模型系統整合與使用者介面
- 發表 4 篇 SCI 國際期刊

✓ Ching-Hsien Hsu and Shih-Chang Chen, "Efficient Selection Strategies towards Processor Reordering Techniques for Improving Data Locality in Heterogeneous Clusters", Accepted for publication, The Journal of Supercomputing (SCI, EI, IF=0.615), 2010.

✓ Ching-Hsien Hsu and Tai-Lung Chen, "Performance and Economization Oriented Scheduling Techniques for Managing Applications with QoS Demands in Grids", Accepted, *International Journal of Ad-Hoc and Ubiquitous Computing*(IJAHUC), Vol. 5, No. 4, pp. 219-226, 2010. (SCIE, EI) (IF=0.66)

✓ Ching-Hsien Hsu, Laurence T. Yang, Frode Eika Sandnes and Zhen Liu, "Toward Merging Ubiquitous and Grid Services", Journal of Internet Technologies (JIT), Vol. 11, Issue 1, pp. 1-2, January 2010. (SCIE, EI)

✓ Ching-Hsien Hsu, Chi-Guey Hsu, Shih-Chang Chen and Tai-Lung Chen, "Message Transmission Techniques for Low Traffic P2P Services", *International Journal of Communication Systems*(IJCS), Vol. 22, No. 9, pp. 1105-1122, September 2009 (DOI: 10.1002/dac.1010) (SCI, EI, IF=0.394)

- 發表 6 篇國際研討會論文

- ✓ Chia-Wei Chu, Ching-Hsien Hsu, Hsi-Ya Chang, Shuen-Tai Wang and Kuan-Ching Li, "Parallel File Transfer for Grid Economic" Proceedings of the 4th ICST International Conference on Scalable Information Systems (InfoScale 2009), Hong Kong, June, 2009, Lecture Notes of the Institute for Computer Science, Social Informatics and Telecommunications Engineering, (ISBN: 978-3-642-10484-8) Vol. 18, pp. 76-89, (DOI: 10.1007/978-3-642-10485-5_6) (EI)
- ✓ Yun-Chiu Ching, Ching-Hsien Hsu and Kuan-Ching Li, "On Improving Network Locality in BitTorrent-Like Systems" Proceedings of the 4th ICST International Conference on Scalable Information Systems (InfoScale 2009), Hong Kong, June, 2009, Lecture Notes of the Institute for Computer Science, Social Informatics and Telecommunications Engineering, (ISBN: 978-3-642-10484-8) Vol. 18, pp. 58-75, (DOI: 10.1007/978-3-642-10485-5_5) (EI)
- ✓ Ching-Hsien Hsu, Yen-Jun Chen, Kuan-Ching Li, Hsi-Ya Chang and Shuen-Tai Wang, "Power Consumption Optimization of MPI Programs on Multi-Core Clusters" Proceedings of the 4th ICST International Conference on Scalable Information Systems (InfoScale 2009), Hong Kong, June, 2009, Lecture Notes of the Institute for Computer Science, Social Informatics and Telecommunications Engineering, (ISBN: 978-3-642-10484-8) Vol. 18, pp. 108-120, (DOI: 10.1007/978-3-642-10485-5_8) (EI)
- ✓ Shih-Chang Chen, Ching-Hsien Hsu, Tai-Lung Chen, Kun-Ming Yu, Hsi-Ya Chang and Chih-Hsun Chou, "A Compound Scheduling Strategy for Irregular Array Redistribution in Cluster Based Parallel System," Proceedings of the 2nd Russia-Taiwan Symposium on Methods and Tools for Parallel Programming (MTPP 2010), LNCS 6083, pp. 68-77, 2010. (EI)
- ✓ Ching-Hsien Hsuand Tai-Lung Chen, "Adaptive Scheduling based on Quality of Services in Heterogeneous Environments", IEEE Proceedings of the 4th International Conference on Multimedia and Ubiquitous Engineering (MUE), Cebu, Philippines, Aug. 2010.
- ✓ Ching-Hsien Hsu, Tai-Lung Chen and Kun-Ho Lee, "QoS Based Parallel File Transfer for Grid Economics" IEEE Proceedings of the 2009 International Conference on Multimedia Information Networking and Security (MINES 2009), pp. 653-657, Wuhan, China, November, 2009. (EI)

五、計畫成果自評

本計畫之研究成果達到計畫預期之目標。研究團隊共計發表了四篇國際期刊與六篇國際研討會論文，其中，[45] Performance and Economization Oriented Scheduling Techniques for Managing Applications with QoS Demands in Grids 是改善 QoS-Guided 的排程。[46] Scheduling of Job Combination and Dispatching Strategy for Grid and Cloud System 是有關在異質性系統或格網環境下將工作排程最佳化，而[47] QoS Based Parallel File Transfer for Grid Economics 是改善 QoS-Guided 的排程。[48] Message Transmission Techniques for Low Traffic P2P Services 則是訊息傳送最佳化技術研究。

本計畫有目前研究成果，感謝國科會給予機會、也感謝許多合作的學校、教授、同學協助軟硬體的架設、測試、與協助機器的管理。另外，對於參與研究計畫執行同學的認真，本人亦表達肯定與感謝。

六、參考文獻

[1] [1] Jesper Andersson, Morgan Ericsson, Welf Löwe, and Wolf Zimmermann, "Lookahead Scheduling for Reconfigurable GRID Systems," *10th International Europar '04: Parallel Processing*, vol. 3149, pp. 263-270, 2004.

[2] [2] Luiz Angelo, Barchet-Steffenel and Grégory Mounié, "Scheduling Heuristics for Efficient Broadcast Operations on Grid Environments," *Proceedings of Parallel and Distributed Processing Symposium*, pp. 8, 2006.

[3] [3] Shah Asaduzzaman and Muthucumaru Maheswaran, "Heuristics for Scheduling Virtual Machines for

Improving QoS in Public Computing Utilities," *9th International Conference on Computer and Information Technology* –ICCIT-2006.

[4] [4] Henri E. Bal, Aske Plaat, Mirjam G. Bakker, Peter Dozy, and Rutger F.H. Hofman, "Optimizing Parallel Applications for Wide-Area Clusters," *Proceedings of the 12th International Parallel Processing Symposium IPPS'98*, pp 784-790, 1998.

[5] [5] Mohammad Banikazemi, Vijay Moorthy and Dhabaleswar K. Panda, "Efficient collective communication on heterogeneous networks of workstations," *Proceedings of International Conference on Parallel Processing*, pp. 460 - 467 Aug. 1998.

[6] [6] O. Beaumont, A. Legrand, L. Marchal and Y. Robert, "Complexity results and heuristics for pipelined multicast operations on heterogeneous platforms," *Proceedings of International Conference on ICPP 2004*, pp. 267 -274, 2004.

[7] [7] O. Beaumont, A. Legrand and Y. Robert, "Optimal algorithms for scheduling divisible workloads on heterogeneous systems," *Proceedings of the 12 th IEEE Heterogeneous Computing Workshop*, 2003.

[8] [8] O. Beaumont, A. Legrand, L. Marchal and Y. Robert, "Optimizing the Steady-State Throughput of Broadcasts on Heterogeneous Platforms Heterogeneous Platforms," *In Technical Report RR-2003-34LIP, ENS Lyon, France*, June 2003.

[9] [9] O. Beaumont, A. Legrand, L. Marchal and Y. Robert, "*Pipelining Broadcasts on Heterogeneous Platforms,"IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 4, pp. 300 – 313, April 2005.

[10] [10] O. Beaumont, L. Marchal and Y. Robert, "Broadcast Trees for Heterogeneous Platforms," *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium*, pp. 80b, 2005.

[11] [11] P. Bhat, C. Raghavendra and V. Prasanna. "Efficient collective communication in distributed heterogeneous systems," *Journal of Parallel and Distributed Computing*, pp. 15 – 24, 2003.

[12] [12] M. Faerman, A. Birnbaum, H. Casanova and F. Berman, "Resource Allocation for Steerable Parallel Parameter Searches," *Proceedings of GRID'02*, 2002.

[13] [13] A. Faraj, P. Patarasuk and X. Yuan, "Bandwidth Efficient All-to-all Broadcast on Switched Clusters," *The 2005 IEEE Cluster 2005, Sept. 27-30, 2005*.

[14] [14] Ligang He, Stephen A. Jarvis, Daniel P. Spooner, Xinuo Chen , Graham R. Nudd, "Hybrid Performance-oriented Scheduling of Moldable Jobs with QoS Demands in Multiclusters and Grids," *Grid and Cooperative Computing (GCC 2004)*, vol. 3251, pp. 217–224, 2004.

[15] [15] Ching-Hsien Hsu, Guan-Hao Lin, Kuan-Ching Li and Chao-Tung Yang, "Localization Techniques for Cluster-Based Data Grid," *Proceedings of the 6th ICA3PP, Melbourne, Australia*, 2005.

[16] [16] Ching-Hsien Hsu, Tzu-Tai Lo and Kun-Ming Yu "Localized Communications of Data Parallel Programs on Multi-cluster Grid Systems," *European Grid Conference*, LNCS 3470, pp. 900 – 910, 2005.

[17] [17] Jih-Woei Huang and Chih-Ping Chu, "An Efficient Communication Scheduling Method for the Processor Mapping Technique Applied Data Redistribution," *The Journal of Supercomputing*, vol. 37, no. 3, pp. 297-318, 2006.

[18] [18] Jang-uk In, Paul Avery, Richard Cavanaugh, Sanjay Ranka, "Policy Based Scheduling for Simple Quality of Service in Grid Computing," *18th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2004)*, pp. 23, 2004.

[19] [19] Florin Isaila and Walter F. Tichy, "Mapping Functions and Data Redistribution for Parallel Files," *Proceedings of IPDPS 2002 Workshop on Parallel and Distributed Scientific and Engineering Computing with Applications*, Fort Lauderdale, April 2002.

[20] [20] Bahman Javadi, J.H. Abawajy and Mohammad K. Akbari " *Performance Analysis of Interconnection Networks for Multi-cluster Systems," Proceedings of the 6th ICCS*, LNCS 3516, pp. 205 – 212, 2005.

[21] [21] Bahman Javadi, Mohammad K. Akbari and Jemal H. Abawajy, " *Performance Analysis of Heterogeneous Multi-Cluster Systems," Proceedings of ICPP*, 2005.

[22] [22] S. Lennart Johnsson and Ching-Tien Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Trans. Computers*,vol. 38, no. 9, pp. 1249-1268, Sep. 1989.

[23] [23] E. T. Kalns and L. M. Ni, "Processor mapping techniques toward efficient data redistribution," *IEEE TPDS*, vol. 6, no. 12, pp. 1234-1247, 1995.

[24] [24] S. Khuller and Y. Kim, "On broadcasting in heterogeneous networks," *In Proceedings of the 16th Annual ACM Symposium on Parallel Architectures and Algorithms*, 2004.

[25] [25] Junsu Kim, Sung Ho Moon, and Dan Keun Sung, "Multi-QoS Scheduling Algorithm for Class Fairness in High Speed Downlink Packet Access," *Proceedings of IEEE Personal, Indoor and Mobile Radio Communications Conference (PIMRC 2005)*, vol. 3, pp. 1813-1817, 2005.

[26] [26] Jens Koonp and Eduard Mehofer, "Distribution assignment placement: Effective optimization of redistribution costs," *IEEE TPDS*, vol. 13, no. 6, June 2002.

[27] [27] Chao Lin, "Efficient broadcast in a heterogeneous network of workstations using two sub-networks," *Proceedings of 7th International Symposium on Parallel Architectures*, Algorithms and Networks, pp. 273 - 279, May 2004.

[28] [28] Chao Lin, "Efficient contention-free broadcast in heterogeneous network of workstation with multiple send and receive speeds," *Proceedings Eighth IEEE International Symposium on Computers and Communication*, 2003 (ISCC 2003), pp. 1277 – 1284, vol.2, 2003.

[29] [29] Chao Lin, Yu-Chee Tseng and Jang-Ping Sheu, "Efficient Single-node Broadcast in Switched-based Network of Workstations with Network Partitioning," *Proceedings of Tenth International Conference on Computer Communications and Networks*, pp. 68-74, 2001.

[30] [30] Jong Sik Lee, "Data Distribution Management Modeling and Implementation on Computational Grid," *Proceedings of the 4th GCC, Beijing, China, 2005*.

[31] [31] Victor E. Mendia and Dilip Sarkar, "Optimal Broadcasting on the Star Graph," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 4, pp. 389 - 396, July 1992.

[32] [32] M.A. Moges and T.G. Robertazzi, "Grid scheduling divisible loads from multiple sources via linear programming," 1*6th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, pp. 423-428, 2004.

[33] [33] J. Moore and M. Quinn, "Generating an Efficient Broadcast Sequence Using Reflected Gray Codes, "*IEEE Trans. Parallel and Distributed Systems*, vol. 8, no. 11, pp. 1117-1122, Nov. 1997.

[34] [34] Aske Plaat, Henri E. Bal, and Rutger F.H. Hofman, "Sensitivity of Parallel Applications to Large Differences in Bandwidth and Latency in Two-Layer Interconnects," *Proceedings of the 5th IEEE High Performance Computer Architecture HPCA'99*, pp. 244-253, 1999.

[35] [35] Sriram Ramanujam, Mitchell D. Theys, "Adaptive Scheduling based on Quality of Service in Distributed Environments," I*nternational Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pp. 671-677, 2005.

[36] [36] Adele A. Rescigno, "Optimal Polling in Communication Networks," *IEEE Trans. on Parallel and Distributed System*, vol. 8, no. 5, pp. 449 - 461 May 1997.

[37] [37] Gerald Sabin, Rajkumar Kettimuthu, Arun Rajan and P Sadayappan, "Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment," *in the Proc. of the 9th International Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes In Computer Science*; Vol. 2862, pp. 87-104 , June 2003.

[38] [38] Fernando G. Tinetti and Andrés Barbieri, "An Efficient Implementation for Broadcasting Data in Parallel Applications over Ethernet Clusters," *Proceeding of 17th International Conference on Advanced Information Networking and Applications*, pp. 593 - 596 March 2003.

[39] [39] Fernando G. Tinetti and E. Luque, "Efficient Broadcasts and Simple Algorithms for Parallel Linear Algebra Computing in Clusters," *International Proceeding of Parallel and Distributed Processing Symposium*, pp. 8, 2003.

[40] [40] Weizhe Zhang, Hongli Zhang, Hui He, Mingzeng Hu, "Multisite Task Scheduling on Distributed Computing Grid," *Lecture Notes in Computer Science*, vol. 3033, pp. 57–64, 2004.

[41] [41] Ching-Hsien Hsu, Bing-Ru Tsai, Tai-Lung Chen and Shih-Chang Chen, "Scheduling for Atomic Broadcast Operation in Heterogeneous Networks with One Port Model," *Accepted, The Journal of Supercomputing (SCI, EI), Kluwer Academic Publisher*, 2009.

[42] [42] Ching-Hsien Hsu, Tai-Lung Chen and Jong-Hyuk Park, "On improving resource utilization and system throughput of master slave jobs scheduling in heterogeneous systems," *Journal of Supercomputing, Springer*, Vol. 45, No. 1, pp. 129-150, July 2008. (SCI, EI).

[43] [43] Ching-Hsien Hsu and Yung-Chneg Chang, "Job Rescheduling Techniques for Optimizing Resource Utilization and Makespan in Grid", *Proceedings of the 5th Workshop on Grid Technology and Applications (WoGTA'08)*, Dec. 2008.

[44] [44] Ching-Hsien Hsu, Justin Zhan, Wai-Chi Fang and Jianhua Ma, "Towards Improving QoS-Guided Scheduling in Grids," *IEEE Proceedings of the third ChinaGrid Annual Conference (ChinaGrid 2008)*, Dunhunag, Gansu, China.

[45] [45] Ching-Hsien Hsu and Tai-Lung Chen, "Performance and Economization Oriented Scheduling Techniques for Managing Applications with QoS Demands in Grids", *International Journal of Ad-Hoc and Ubiquitous Computing*, Vol. 5, No. 4, pp. 219-226, 2010.

[46] [46] Tai-Lung Chen, Ching-Hsien Hsu and Shih-Chang Chen, "Scheduling of Job Combination and

Dispatching Strategy for Grid and Cloud System," *Proceedings on the 5rd International Conference on Grid and Pervasive Computing (GPC'10)*, LNCS 6104, pp. 612-621, May 2010.

[47] [47] Ching-Hsien Hsu, Tai-Lung Chen and Kun-Ho Lee, "QoS Based Parallel File Transfer for Grid Economics" *Proceedings of the 2009 International Conference on Multimedia Information Networking and Security (MINES'09)*, pp. 653-657, November 2009.

[48] [48] Ching-Hsien Hsu, Chi-Guey Hsu, Shih-Chang Chen and Tai-Lung Chen, "Message Transmission Techniques for Low Traffic P2P Services", *International Journal of Communication Systems*, Vol. 22, No. 9, pp. 1105-1122, September 2009.

出席國際學術會議心得報告

計畫名稱	應用 P2P 與 Web 技術發展以 SOA 為基礎的格網中介軟體與經濟模型
計畫編號	NSC 97-2628-E-216-006-MY3
報告人姓名	許慶賢
服務機構及職稱	中華大學資訊工程學系教授
會議名稱	The 2nd Russia-Taiwan Symposium on Methods and Tools of Parallel Programming Multicomputers (MTPP 2010)
會議/訪問時間地點	海參威, 俄羅斯 / 2010.05.16-19
發表論文題目	A Compound Scheduling Strategy for Irregular Array Redistribution in Cluster Based Parallel System

參加會議經過

會議時間	行程敘述
2010/05/16	(上午) 10:00 會場報到 11:00 參訪研究中心 (半日) (下午) 6:00 committee meeting (晚上) 7:00 參加歡迎茶會
2010/05/17	(上午) 9:00 開幕致詞 9:10 聽取 Parallel Algorithm 相關論文發表 11:00 聽取 Models and Tools 相關論文發表 (下午) 2:00 聽取 Parallel Programming 相關論文發表

2010/05/18	(上午) 9:00　發表論文 11:00 聽取 System Algorithm 相關論文發表 (下午) 2:00 聽取 Numerical simulation 相關論文發表 4:00 參訪 Far East National University (晚上) 7:00 參加晚宴
2010/05/19	(上午) 9:00 聽取 Simulation 相關論文發表

MTPP-10 是台俄雙邊在平行計算研究領域主要的研討會。這一次參與 MTPP-10 ，本人擔任會議議程主席，除了發表相關研究成果以外，也在會場與多位國外教授交換研究心得，並且討論未來可能的合作。

這一次參與 MTPP-10 除了發表我們最新的研究成果以外，也在會場中，向多位國內外學者解釋我們的研究內容，彼此交換研究心得。除了讓別的團隊知道我們的研究方向與成果，藉此，我們也學習他人的研究經驗。經過兩次的雙邊研討會交流，雙方已經找到共同研究的題目，兩邊的團隊也將於今年(2010 年)8 月開始撰寫研究計畫書，進行更密切的合作。

這一次在 Vladivostok, Russia 所舉行的國際學術研討會議議程共計四天。開幕當天由俄羅斯方面的 General Co-Chair，RSA 的 Victor E. Malyshkin 教授，與敝人分別致詞歡迎大家參加這次的第二屆 MTPP 2010 國際研討會。接著全程參與整個會議的流程，也聽取不同論文發表，休息時間與俄羅斯的學者教授交換意見和資訊。本人發表的論文在會議第三天的議程九點三十分發表（A Compound Scheduling Strategy for Irregular Array Redistribution in Cluster Based Parallel System）。本人主要聽取 Parallel and Distributed 、Grid、Cloud 與 Multicore 相關研究，同時獲悉許多新興起的研究主題，並了解目前國外學者主要的研究方向。最後一天，我們把握機會與國外的教授認識，希望能夠讓他們加深對台灣研究的印象。這是一次非常成功的學術研討會。

主辦第一、二屆台俄雙邊學術研討會，感受良多。論文篇數從第一屆的 30 篇到第二屆的 50 篇，也讓本人感受到這個研討會的進步成長。台方參與的教授學生超過 15 個學研單位，包括台大、清華、交大、中研院、成大、中山、等等。俄方也有超過 10 個學研單位的參與。值得一提的是，這一次的論文集我們爭取到 Springer LNCS 的出版，並且在 EI 索引。這一個研討會與發表的論文，其影響力已達到國際的水準。


ФУНДАМЕНТАЛЬНАЯ
БИБЛИОТЕКА
RESEARCH INFORMATION INSTITUTE
FUNDAMENTAL
LIBRARY
110 лет ДВГУ (1899-2009)

A Compound Scheduling Strategy for Irregular Array Redistribution in Cluster Based Parallel System

Shih-Chang Chen[1], Ching-Hsien Hsu[2], Tai-Lung Chen[1], Kun-Ming Yu[2],
Hsi-Ya Chang[3] and Chih-Hsun Chou[2*]

[1] College of Engineering
[2] Department of Computer Science and Information Engineering
Chung Hua University, Hsinchu, Taiwan 300, R.O.C.
[3] National Center for High-Performance Computing, Hsinchu 30076, Taiwan
{scc, robert, tai}@grid.chu.edu.tw, yu@chu.edu.tw, jerry@nchc.org.tw, chc@chu.edu.tw

Abstract. With the advancement of network and techniques of clusters, joining clusters to construct a wide parallel system becomes a trend. Irregular array redistribution employs generalized blocks to help utilize the resource while executing scientific application on such platforms. Research for irregular array redistribution is focused on scheduling heuristics because communication cost could be saved if this operation follows an efficient schedule. In this paper, a two-step communication cost modification (T2CM) and a synchronization delay-aware scheduling heuristic (SDSH) are proposed to normalize the communication cost and reduce transmission delay in algorithm level. The performance evaluations show the contributions of proposed method for irregular array redistribution.

1 Introduction

Scientific application executing on parallel systems with multiple phases requires appropriate data distribution schemes. Each scheme describes the data quantity for every node in each phase. Therefore, performing data redistribution operations among nodes help enhance the data locality.

Generally, data redistribution is classified into regular and irregular redistributions. `BLOCK`, `CYCLIC` and `BLOCK-CYCLIC`(c) are used to specify array decomposition for the former while user-defined function, such as `GEN_BLOCK`, is used to specify array decomposition for the latter. High Performance Fortran version 2 provides `GEN_BLOCK` directive to facilitate the data redistribution for user-defined function. To perform array redistribution efficiently, it is important to follow a schedule with low communication cost.

With the advancement of network and the popularizing of cluster computing research in campus, it is a trend to join clusters in different regions to construct a complex parallel system. To performing array redistribution on this platform, new techniques are required instead of existing methods.

Schedules illustrate time steps for data segments (messages) to be transmitted in appropriate time. The cost of schedules given by scheduling heuristics is the summation of cost of every time steps while cost of each time step is dominated by the message with largest cost. A phenomenon is observed that most local transmissions, which are happened in a node, do not dominate the cost of each step although they are in algorithm level for existing methods. In other words, they are overestimated. Since a node can send and receive only one message in the same time step [5], the arranged position of each message becomes important. Therefore, a *two-step communication cost modification* (*T2CM*) and a *synchronization delay-aware scheduling heuristic* (*SDSH*) are proposed to deal with the overestimate problems, reduce overall communication cost and avoid synchronization of schedules in algorithm level.

The rest of this paper is organized as follows: Section 2 gives a survey of existing works related to array redistribution. Section 3 gives notations, terminology and examples to explain each parts of scheduling heuristics. The proposed techniques are described in section 4. Section 5 presents the results of the comparative evaluation, while section 6 concludes the paper.

2 Related Work

Array redistribution techniques have been developed for regular array redistribution and `GEN_BLOCK` redistribution in many papers. Both kinds of redistribution issues require at least two sorts of techniques. One is communication sets identification which decomposes array for nodes; the other one is communication scheduling method which derives schedules to shorten the overall transmission cost for redistributions.
ScaLAPACK [9] was proposed to identify communication sets for regular array redistribution. Guo *et al*. [2] proposed a symbolic analysis method to help generate messages for `GEN_BLOCK` redistribution. Hsu *et al*. [3] proposed the *Generalized Basic-Cycle Calculation* method to shorten the communication for generalized cases. The research on prototype framework for distributed memory platforms is proposed by Sundarsan *et al.* [11] who developed a method to distribute multidimensional block-cyclic arrays on processor grids. Karwande *et al*. [8] presented *CC-MPI* with the compiled communication technique to optimize collective communication routines. Huang *et al*. [6] proposed a flexible processor mapping technique to reduce the number of data element exchanging among processors and enhance the data locality. To reduce indexing cost, a processor replacement scheme was proposed [4]. With local matrix and compressed *CRS* vectors transposition schemes the communication cost can be reduced significantly. Combining the advantages of relocation scheduling algorithm and divide-and-conquer scheduling algorithm, Wang *et al*. [12] proposed a method with two phases for `GEN_BLOCK` redistribution. The first phase acts like relocation algorithm, but the contentions avoidance mechanism of second phase will not be proceeded immediately while contentions happened. To minimize the total communication time, Cohen *et al*. [1] supposed that at most *k* communication can be performed at the same time and proposed two algorithms with low complexity and fast heuristics. A study [7] focusing on the cases of local redistributions and inter-cluster redistribution was given by Jeannot and Wagner. It compared existing scheduling methods and described the difference among them. Rauber and Runger [10] presented a data-re-distribution library to deal with composed data structures which are distributed to one or more processor groups for executing multiprocessor task on distributed memory machines or cluster platforms. Hsu *et al*. [5] proposed a two-phase degree-reduction scheduling heuristic to minimize the overall communication cost. The proposed method derives each time step of a complete schedule by performing degree reduction technique while the number of messages of each node representing the degree of each vertex in algorithm level.

3 Preliminary

Following are notations, terminology and examples to explain each parts of scheduling heuristics for `GEN_BLOCK` redistribution. To improve data locality, multi-phase scientific problems require appropriate data distribution schemes for specific phases. For example, to distribute array for two different phases on six nodes, which are indexed from 0 to 5, two strings, {13, 20, 17, 17, 12, 21} and {16, 18, 13, 16, 29, 8},

are given, where the array size is 100 units. These two strings provide necessary information for nodes to generate messages to be transmitted among them. Fig. 1 shows these messages marked from m_1 to m_{11} and are with information such as data size, source node and destination node in the relative rows.

Scheduling heuristics are developed for providing solutions of time steps to reduce total communication cost for a `GEN_BLOCK` redistribution operation. In each step, there are several messages which are suggested to be transmitted in the same time step. To help perform an efficient redistribution, scheduling methods should avoid node contention, synchronization delay and redundant transmission cost. It is also important to follow policies of messages arrangement, i.e. with the same source nodes, messages should not be in the same step; with the same destination nodes, messages should be in different step; a node can only deal with one message while playing whether source node or destination node. These messages that cannot be scheduled together called conflict tuples, for example, a conflict tuple is formed with messages m_1 and m_2. Note that if a node can only deal with a message while it is a source/destination node, the number of steps for a schedule must be the equal to or more than the number of messages from/to these nodes. In other words, the minimal number of time steps is equal to the maximal number of messages in a conflict tuple, CT_{max}.

Information of messages			
No. of message	Data size	Source node	Destination node
m_1	13	0	0
m_2	3	1	0
m_3	17	1	1
m_4	1	2	1
m_5	13	2	2
m_6	3	2	3
m_7	13	3	3
m_8	4	3	4
m_9	12	4	4
m_{10}	13	5	4
m_{11}	8	5	5

Fig. 1. Information of messages generated from given schemes to be transmitted on six nodes which are indexed from 0 to 5

Fig. 2 gives a schedule with low communication cost and arranges messages in the number of minimal steps. In this result, there are three time steps with messages sent/received to/from different nodes. The values beside $m_{1\sim11}$ are data size, the cost of each step is dominated by the largest one. Thus, m_3, m_1 and m_8 dominate step 1, 2 and 3, and the estimated cost are 17, 13 and 4, respectively. To avoid node contentions, messages m_1 and m_2 are in separate steps due to destination nodes of both messages are the same. Based on same argument, m_2 and m_3 are in separate steps due to both messages are members of a conflict tuple. The total cost which represents the performance of a schedule is the summation of all cost of steps. In other words, a schedule with lower cost is better than another one with higher cost in terms of performance.

A result of scheduling heuristics		
No. of step	No. of message	Cost of step
Step 1	m_3(17), m_5(13), m_7(13) , m_{10}(13)	17
Step 2	m_1(13), m_6(3), m_9(12), m_{11}(8)	13
Step 3	m_2(3), m_4(1), m_8(4)	4
Total cost		34

Fig. 2. A result of scheduling messages with low communication cost and minimal steps

The result in Fig. 2 schedules messages in three steps, which is the number of minimal steps or CT_{max}. The total cost is small which representing low communication cost due to messages with larger cost and messages with smaller cost are in separate steps. However, the schedule can still be better by providing a cost normalization method and a new scheduling technique to avoid synchronization delay among nodes during message transmissions in next section.

4 The Proposed Method

In this paper, a *two-step communication cost modification* (*T*2*CM*) and a *synchronization delay-aware scheduling heuristic* (*SDSH*) are proposed to normalize the communication cost of messages and reduce transmission delay in algorithm level. The first step of *T*2*CM* is a *local reduction* operation, which deal with the message happened in local memory. In other words, candidates are transmissions whose source node and destination node are the same node. For example, m_1, m_3, m_5, m_7, m_9 and m_{11} are such kind of transmissions which happened inside nodes. The second step is a *inter amplification* method, which is responsible for transmissions happened across clusters. Assumed there are two clusters, and node 0~2 are in cluster 1, other nodes are in cluster 2. Then m_6 is such message which is transmitted from cluster 1 to cluster 2. Both operations are responsible for different kind of transmissions due to the heterogeneity of network bandwidth. The *local reduction* operation reduces simulated cost of messages to 1/8 which is evaluated from PC clusters that connected with 100Mbps layer-2 switch. On same argument, *inter amplification* operation increases cost of messages five times. The cost then becomes more practical for real machines when scheduling heuristics try to give a perfect schedule with low communication cost. For previous research, the difference does not exist in algorithm level of scheduling heuristics in and could result in erroneous judgments and high communication cost.

Fig. 3 gives the results of *local reduction* and *inter amplification* operations modifying data size for messages $m_{1\sim11}$. The given schedule in Fig. 2 becomes the results in Fig. 4. Difference of Fig. 2 and Fig. 4 shows the schedule could be improved and explains the explain the erroneous judgments. First, the dominators in step 1 and 2 are changed to others whose estimated cost is larger in Fig. 4. For example, the m_3 and m_1 are replaced by m_{10} and m_6 for both steps, respectively. Second, the cost of step 1 and step 2 are changed due to new dominators are chosen in both steps. Furthermore, the synchronization delay is small in algorithm level but results in more node idle time in practical. For instance, the cost of m_3, m_5, m_7 and m_{10} are 17, 13, 13 and 13 are close to each other in step 1 in Fig. 2. But it is quite different in practical in

Fig. 4, they should be 2.125, 1.625, 1.625 and 13, respectively. Node 1, 2 and 3 must wait for node 4 and 5 to proceed next step because when the transmissions of m_3, m_5 and m_7 are finished, the transmission of m_{10} is still on the way.

Information of messages			
No. of message	Data size	Source node	Destination node
m_1	**1.625**	0	0
m_2	**3**	1	0
m_3	**2.125**	1	1
m_4	**1**	2	1
m_5	**1.625**	2	2
m_6	**15**	2	3
m_7	**1.625**	3	3
m_8	**4**	3	4
m_9	**1.5**	4	4
m_{10}	**13**	5	4
m_{11}	**1**	5	5

Fig. 3. The *local reduction* and *inter amplification* operations derive new data size for messages $m_{1\sim11}$

A result of scheduling heuristics		
No. of step	No. of message	Cost of step
Step 1	m_3(2.125), m_5(1.625), m_7(1.625), m_{10}(13)	13
Step 2	m_1(1.625), m_6(15), m_9(1.5), m_{11}(1)	15
Step 3	m_2(3), m_4(1), m_8(4)	4
Total cost		32

Fig. 4. The results with new dominators and cost

The proposed *synchronization delay-aware scheduling heuristic* is a novel and efficient method to avoid delay among clusters and shorten communication cost while performing `GEN_BLOCK` redistribution. To avoid synchronization delay, the transmissions happened in local memory are scheduled together in one single step instead of separating them among time steps like the results in Fig. 4. Other messages are pre-proceeded by *inter amplification* and then scheduled by a low cost scheduling method which selects messages with smaller cost to shorten the cost of a step and avoid the node contentions. Fig. 5 shows the results of *SDSH* which is with low synchronization delay and is contention free. There are two reasons making the results in Fig. 5 better than the results in Fig. 4. First, *SDSH* successfully avoids synchronization delay by congregating m_1, m_3, m_5, m_7, m_9 and m_{11} in step 3. It also helps reduce the cost of a step. Second, messages m_6 and m_{10} are the most important transmissions in the schedule due to their communication cost can dominate any steps. It is a pity that they are separated in two steps in Fig. 4 due to the node contentions. For example, it is impossible to move m_6 to step 1 to shorten the cost of step 2 due to

m_5 and m_7. The message m_5 owns node 2 as source node and so does m_6. Both messages cannot be scheduled in the same step. Similarly, m_6 and m_7 cannot be scheduled together due to destination node. On same argument, it is impossible to move m_{10} to step 2 due to m_9 and m_{11}. If m_5, m_7, m_9 and m_{11} can be placed in other step, it would be possible to place m_6 and m_{10} together to minimize the communication cost of the results. *SDSH* successfully places them in step 3 and then schedules m_6 and m_{10} in step 1 to shorten the cost of other steps. This operation also successfully avoids node contentions that happened in Fig. 4.

A result of the proposed method		
No. of step	No. of message	Cost of step
Step 1	$m_2(3)$, $m_6(15)$, $m_{10}(13)$	15
Step 2	$m_4(1)$, $m_8(4)$	4
Step 3	$m_1(1.625)$, $m_3(2.125)$, $m_5(1.625)$, $m_7(1.625)$, $m_9(1.5)$, $m_{11}(1)$	2.125
Total cost		21.125

Fig. 5. A result of proposed method with low synchronization delay and contention free

5 Performance Evaluation

To evaluate the proposed method, it is compared with a scheduling method, *TPDR* [5]. The simulator generates schemes (strings) for 8, 16, 32, 64 and 128 nodes, and there are three nodes in a cluster. To constrain the data size of each node, the lower bound and upper bound of each value in the strings are 1 and the value that array size divided by the number of nodes, where the array size is 10,000. If the array is distributed on eight nodes, the lower bound and the upper bound of data size are 1 and 1250 for each node, respectively.

Fig. 6 shows the results of comparisons between *SDSH* and *TPDR*. For each set of node, the number on the right side represents the cases that *SDSH* performs better, *TPDR* performs better or tie cases. In the simulation results for 8 nodes, the proposed method wins 813 cases which is less than 90% because it is easy for both methods to find the same results when performing `GEN_BLOCK` redistribution on few number of nodes. Therefore, the number of tie cases is over than 10%, and is much more than the results of other sets. When performing `GEN_BLOCK` redistribution with more nodes, *SDSH* outperforms *TPDR*, and *TPDR* loses over 92% cases in the rest of the comparisons. Note that the proposed method always find the best results in over 93% cases including the tie cases in all comparisons. It also shows the contribution of *SDSH* for shortening transmission cost and avoiding synchronization delay.

Results of evaluations			
Num. of nodes	*SDSH*	*TPDR*	Same
8	813	76	111
16	946	43	11
32	950	48	2
64	914	79	7
128	903	96	1
Percentage	90.52%	6.84%	2.64%
Total	4526	342	132

Fig. 6. The results of both methods on five sets of nodes with 5,000 cases in total

The attributes of generated cases dependents on the number of nodes, for example, higher CT_{max} and lower communication cost are with higher number of nodes. It is hard to find the same schedules for two scheduling heuristics with larger number of nodes. Fig. 7 shows the information of cases which are used to evaluate the *SDSH* and *TPDR*.

Attributes of given cases				
Num. of nodes	CT_{max}	Average CT_{max}	Cost of 1,000 cases	
			SDSH	*TPDR*
8	6	3.271	6733580	7953932
16	8	3.762	5733523	6983753
32	10	4.246	3564076	4354899
64	10	4.661	2412444	2781670
128	11	5.009	1282008	1520884

Fig. 7. Attributes of given cases for five set of nodes

CT_{max} of results with 128 nodes is 11 which is almost two times larger than the CT_{max} of results with 8 nodes. The average CT_{max} also grows with higher number of nodes. The total cost of schedules given by both methods for 1000 cases with different number of nodes explains the contribution of *SDSH* in Fig. 6. The proposed method provides better schedules and the improves the communication cost about 15% while comparing to *TPDR*. It also explains how *SDSH* outperforms its competitor. Overall speaking, *SDSH* is a novel, efficient and simple method to provide solutions for scheduling communications of `GEN_BLOCK` redistribution.

6 Conclusions

To perform `GEN_BLOCK` redistribution efficiently, research focused on developing scheduling heuristic to shorten communication cost in algorithm level. In this paper, a *two-step communication cost modification* (*T2CM*) and a *synchronization delay-aware scheduling heuristic* (*SDSH*) are proposed to normalize the transmission cost and reduce synchronization delay. The *two-step communication cost modification*

provides *local reduction* and *inter amplification* operations to enhance the importance of messages. The *SDHC* deal with messages separately to avoid synchronization delay and reduce the cost. The performance evaluation shows that the proposed methods outperforms its competitor in 92% cases and improves about 15% on overall communication cost.

References

[1] Cohen, J., Jeannot, E., Padoy, N., Wagner, F.: Messages Scheduling for Parallel Data Redistribution between Clusters. IEEE Transactions on Parallel and Distributed Systems 17(10), 1163-1175 (2006)

[2] Guo, M., Pan, Y., Liu, Z.: Symbolic Communication Set Generation for Irregular Parallel Applications. The Journal of Supercomputing 25(3), 199-214 (2003)

[3] Hsu, C.-H., Bai, S.-W., Chung, Y.-C., Yang, C.-S.: A Generalized Basic-Cycle Calculation Method for Efficient Array Redistribution. IEEE Transactions on Parallel and Distributed Systems 11(12), 1201-1216 (2000)

[4] Hsu, C.-H., Chen, M.-H., Yang, C.-T., Li, K.-C.: Optimizing Communications of Dynamic Data Redistribution on Symmetrical Matrices in Parallelizing Compilers. IEEE Transactions on Parallel and Distributed Systems 17(11), (2006)

[5] Hsu, C.-H., Chen, S.-C., Lan, C.-Y.: Scheduling Contention-Free Irregular Redistribution in Parallelizing Compilers. The Journal of Supercomputing 40(3), 229-247 (2007)

[6] Huang, J.-W., Chu, C.-P.: A flexible processor mapping technique toward data localization for block-cyclic data redistribution. The Journal of Supercomputing 45(2), 151-172 (2008)

[7] Jeannot, E., Wagner, F.: Scheduling Messages For Data Redistribution: An Experimental Study. The International Journal of High Performance Computing Applications 20(4), 443-454 (2006)

[8] Karwande, A., Yuan, X., Lowenthal, D. K.: An MPI prototype for compiled communication on ethernet switched clusters. Journal of Parallel and Distributed Computing 65(10), 1123-1133 (2005)

[9] Prylli, L., Touranchean, B.: Fast runtime block cyclic data redistribution on multiprocessors. Journal of Parallel and Distributed Computing, 45(1), 63-72 (1997)

[10] Rauber, T., Rünger G.: A Data Re-Distribution Library for Multi-Processor Task Programming. International Journal of Foundations of Computer Science 17(2), 251-270 (2006)

[11] Sudarsan, R., Ribbens, C. J.: Efficient Multidimensional Data Redistribution for Resizable Parallel Computations. In: Fifth International Symposium on Parallel and Distributed Processing and Applications, 182-194 (2007)

[12] Wang, H., Guo, M., Wei, D.: Message Scheduling for Irregular Data Redistribution in Parallelizing Compilers. IEICE Transactions on Information and Sysmtes E89-D(2), 418-424 (2006)

出席國際學術會議心得報告

計畫名稱	應用 P2P 與 Web 技術發展以 SOA 為基礎的格網中介軟體與經濟模型
計畫編號	NSC 97-2628-E-216-006-MY3
報告人姓名	許慶賢
服務機構及職稱	中華大學資訊工程學系教授
會議名稱	The 12th IEEE International Conference on Computational Science and Engineering (CSE-09)
會議/訪問時間地點	Vancouver, Canada / 2009. 08. 29-31
發表論文題目	Data Distribution Methods for Communication Localization in Multi-Clusters with Heterogeneous Network

參加會議經過

會議時間	行程敘述
2009/08/29	(上午) 8:00 **會場**報到、**聆聽** Keynote Speech Privacy, Security, Risk and Trust in Service-Oriented Environments by Stephen S. Yau 9:00 發表論文 10:30 聽取 Parallel Algorithm 相關論文發表 (下午) 1:00 **聆聽** Keynote Speech Elections with Practical Privacy and Transparent Integrity by David Chaum 2:00 聽取 Grid Computing相關論文發表 3:30 主持 Session (晚上) 7:30 參加歡迎茶會

2009/08/30	(上午) 9:00 聆聽 Keynote Speech Cache-Aware Scheduling and Analysis for Multicores by Wang Yi 10:30 聽取 P2P 相關論文發表 (下午) 1:00 聆聽 Keynote Speech Network Analysis and Visualization for Understanding Social Computing by Ben Shneiderman 2:15 參加Panel discussion 3:45 主持 Session (晚上) 7:30 參加晚宴
2009/08/31	(上午) 8:00 聆聽 Keynote Speech White Space Networking - Is it Wi-Fi on Steroids? by Prof. Victor Bahl 10:30 聽取 Network Management 相關論文發表 (下午) 1:30 聆聽 Keynote Speech Computational Science and Engineering in Emerging Cyber-Ecosystems by Prof. Manish Parashar 2:00 主持 Session

這一次在 Vancouver, Canada 所舉行的國際學術研討會議共計三天。這三天每天早上下午都各有穿插專題演講，共邀請了七個不同領域的專家學者給予專題演講，第一天邀請了 Dr. Stephen S. Yau (Arizona State University, USA)與 David Chaum 分別針對 Privacy, Security, Risk and Trust in Service-Oriented Environments 和 Elections with Practical Privacy and Transparent Integrity 給予專題演講揭開研討會的序幕，接下來兩天也分別有 Wang Yi (North Eastern University, China) 、Ben Shneiderman、Dr. Fei-Yue Wang、Prof. Victor Bahl 和 Prof. Manish Parashar (Rutgers University)五位專家學者針對 Cache-Aware Scheduling and Analysis for Multicores 、Network Analysis and Visualization for Understanding Social Computing、Social

Computing Applications and Trends、White Space Networking - Is it Wi-Fi on Steroids?、Computational Science and Engineering in Emerging Cyber-Ecosystems 五個不同的題目給予精闢的演講。本人在這次研討會擔任兩個場次的 Chair 並發表了一篇論文，論文題目為 Data Distribution Methods for Communication Localization in Multi-Clusters with Heterogeneous Network ，擔任 Chair 分別為 CSE-09 (Session A16) 和 SEC-09(Session A27)，本人參與了三天全程會議，也選擇了一些相關領域之場次來聆聽論文發表。

Data Distribution Methods for Communication Localization in Multi-Clusters with Heterogeneous Network

Shih-Chang Chen[1], Ching-Hsien Hsu[2] and Chun-Te Chiu[2]
[1] *Institute of Engineering and Science*
[2] *Department of Computer Science and Information Engineering*
Chung Hua University, Hsinchu, Taiwan 300, R.O.C.
chh@chu.edu.tw

Abstract

Grid computing integrates scattered clusters, servers, storages and networks in different geographic locations to form a virtual super-computer. Along with the development of grid computing, dealing with the data distribution requires a method which is faster and more effective for parallel applications in order to reduce data exchange between clusters. In this paper, we present two methods to reduce inter-cluster communication cost based on the consideration to different kinds of communication cost and a simple logic mapping technology. Our theoretical analyses and simulation results show the proposed methods are better than the methods without reordering processor and considering the communication cost. The performance evaluation shows that the proposed methods not only reduce communication cost successfully but also achieve a great improvement.

1. Introduction

Computing grid system [5] integrates geographically distributed computing resources to establish a virtual and high expandable parallel environment. Cluster grid is a typical paradigm which is connected by software of computational grids through the Internet. In cluster grid, computers might exchange data through network to other computers to run job completion. This consequently incurs two kinds of communication between grid nodes. If the two grid nodes are geographically belong to different clusters, the messaging should be accomplished through the Internet. We refer this kind of data transmission as external communication. If the two grid nodes are geographically in the same space domain, the communications take place within a cluster; we refer this kind of data transmission as interior communication. Intuitionally, the external communication is usually with higher communication latency than that of the interior communication. Therefore, to efficiently execute parallel programs on cluster grid, it is extremely critical to avoid large amount of external communications.

Array redistribution is usually required for efficiently redistributing method to execute a data-parallel program on distributed memory multi-computers. Some efficient communication scheduling methods for the Block-Cyclic redistribution had been proposed which can help reduce the data transmission cost. The previous work [9, 10] presents a generalized processor reordering technique for minimizing external

communications of data parallel program on cluster grid. The key idea is that of distributing data to grid/cluster nodes according to a mapping function at data distribution phase initially instead of in numerical-ascending order.

In this paper, we consider the issue of real communication cost among a number of geographically grid nodes which belong to different clusters. Method proposed previously has less communication cost by reordering logic id of processors. Base on this idea, two new processor reorder techniques are proposed to adapt the heterogeneous network environment.

This paper is organized as follows. Section 2 presents related work. In section 3, we provide background and review previously proposed processor reorder techniques. In section 4, the Global Reordering technique is proposed for processor reordering in section 4.1. The Divide and Conquer Reordering technique is proposed in section 4.2. In section 5, we present the results of the evaluation of the new schemes. Finally we have the conclusions and future work in section 6.

2. Related Work

Research work on computing grid have been broadly discussed on different aspects, such as security, fault tolerance, resource management [4, 6], job scheduling [1, 20, 21, 22], and communication optimizations [2]. Commutating grid is characterized by a large number of interactive data exchanges among multiple distributed clusters over a network. Thus, providing a reliable response in reasonable time with limited communication and computation resources for reducing the interactive data exchanges is required. Jong Sik Lee [16] presented a design and development of a data distribution management modeling in computational grid.

For the issue of communication optimizations, Dawson *et al*. [2] addressed the problems of optimizations of user-level communication patterns in the local space domain for cluster-based parallel computing. Plaat *et al*. analyzed the behavior of different applications on wide-area multi-clusters [3, 19]. Similar research works were studied in the past years over traditional supercomputing architectures [11, 14]. Guo *et al*. [7] eliminated node contention in communication step and reduced communication steps with the schedule table. Y. W. Lim *et al*. [18] presented an efficient algorithm for Block-Cyclic data realignments. Jih-Woei Huang and Chih-Ping Chu [8] presented a unified approach to construct optimal communication schedules for the processor mapping technique applying Block-Cyclic redistribution. The proposed method is founded on the processor mapping technique and can more efficiently construct the required communication schedules than other optimal scheduling methods. A processor mapping technique presented by Kalns and Ni [15] can minimize the total amount of communicating data. Namely, the mapping technique minimizes the size of data that need to be transmitted between two algorithm phases. Lee *et al*. [17] proposed similar method to reduce data communication cost by reordering the logical processors' id. They proposed four algorithms for logical processor reordering. They also compared the four reordering algorithms under various conditions of communication patterns. There is significant improvement of the above research for parallel applications on distributed memory multi-computers. However, most techniques are applicable for applications running on local space domain, like single cluster or parallel machine. Ching-Hsien Hsu *et al*.

[9] presented an efficient method for optimizing localities of data distribution when executing data parallel applications. The data to logical grid nodes mapping technique is employed to enhance the performance of parallel programs on cluster grid.

For a global grid of clusters, these techniques become inapplicable due to various factors of Internet hierarchical and its communication latency. More and more multi-clusters under heterogeneous network environment in which the performance issue was of primary importance on. Bahman Javadi *et al.* [12, 13] proposed an analytical model for studying the capabilities and potential performance of interconnection networks for multi-cluster systems. In this following discussion, our emphasis is on minimizing the communication costs for data parallel programs on cluster grid and on enhancing data distribution of communication localities with heterogeneous network.

3. Research Model

3.1 Identical Cluster Grid

To explicitly define the problem, upon the number of clusters (C), number of computing nodes in each cluster (n_i), $1 \leqq i \leqq C$, the number of sub-blocks (K) and <G(C):{n_1, n_2, n_3, …, n_c}> presents the cluster grid model with n_i computing nodes in each cluster. The definition of symbols is shown in Table 1.

Table 1 The definition of symbols.

C	The number of clusters.
K	The degree of refinement
n_i	The number of computing nodes in each cluster.
G(C):{n_1, n_2, n_3, …, n_c}	The cluster grid model

We consider two models of cluster grid when performing data reallocation. Figure 1 shows an example of localization technique for explanation. The degree of data refinement is set to three (K = 3). This example also assumes an identical cluster grid that consists of three clusters and each cluster provides three nodes to join the computation. In algorithm phase, in order to accomplish the fine-grained data distribution, processors partition its own block into K sub-blocks and distribute them to corresponding destination processors in ascending order of processors' id that specified in most data parallel programming languages. For example, processor P_0 divides its data block A into a_1, a_2, and a_3; it then distributes these three sub-blocks to processors P_0, P_1 and P_2, respectively. Because processors P_0, P_1 and P_2 belong to the same cluster with P_0; therefore, these three communications are interior. However, the same situation on processor P_1 generates three external communications. Because processor P_1 divides its local data block B into b_1, b_2, and b_3. It then distributes these three sub-blocks to processors P_3, P_4 and P_5, respectively. As processor P_1 belongs to *Cluster*-1 and processors P_3, P_4 and P_5 belong to *Cluster*-2, there are three external communications. Figure 1(a) summarizes all messaging patterns of this example into the communication

table. Messages $\{a_1, a_2, a_3\}$, $\{e_1, e_2, e_3\}$ and $\{i_1, i_2, i_3\}$ are presented interior communications ($|I| = 9$); all the others are external communications ($|E| = 18$).

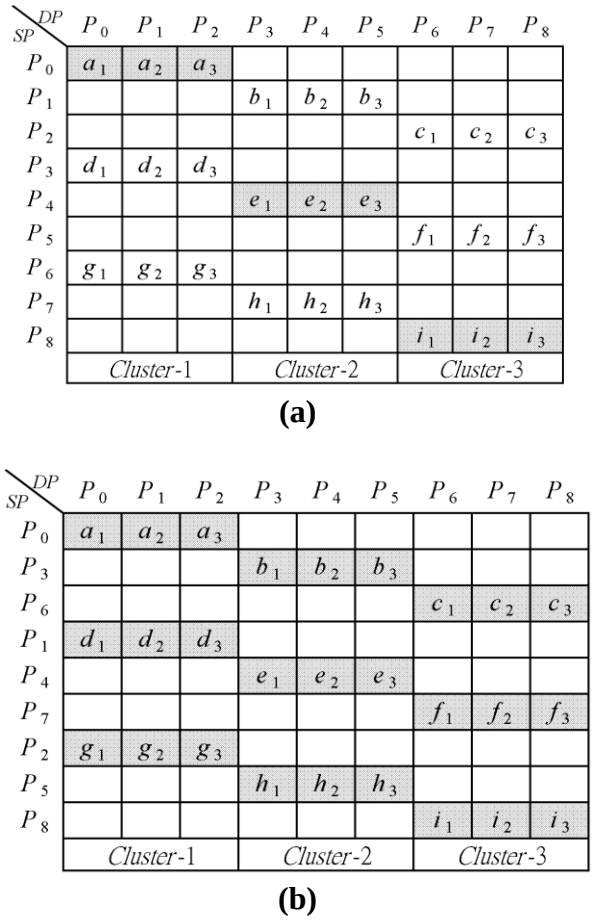
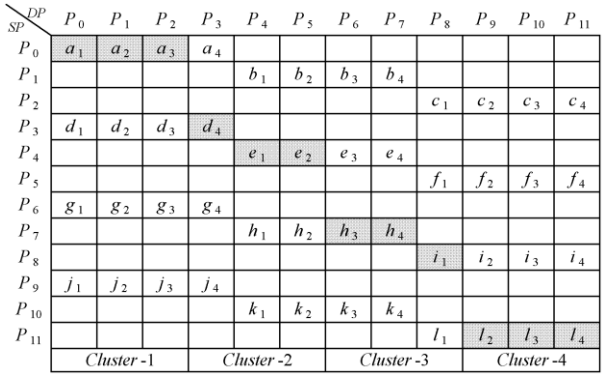


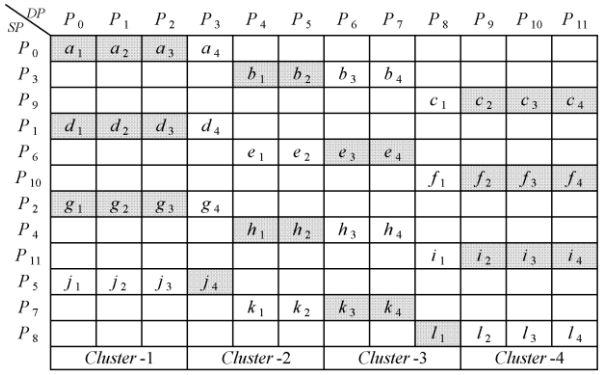
Figure 1. Communication tables of data reallocation over the cluster grid. (a) Without data mapping. (b) With data mapping.

The idea of changing logical processor mapping [15, 16] is employed to minimize data transmission time of runtime array redistribution in the previous research works. In the cluster grid, we can derive a mapping function to produce a realigned sequence of logical processors' id for grouping communications into the local cluster. Given an identical cluster grid with C clusters, a new logical id for replacing processor P_i can be determined by $\text{New}(P_i) = (i \bmod C) * K + (i / C)$, where K is the degree of data refinement. Figure 1(b) shows the communication table of the same example after applying the above reordering scheme. The source data is distributed according to the reordered sequence of processors' id, i.e., $<P_0, P_3, P_6, P_1, P_4, P_7, P_2, P_5, P_8>$ which is computed by mapping function. Therefore, we have $|I| = 27$ and $|E| = 0$.

For the case of K (degree of refinement) is not equal to n (the number of grid nodes in each cluster), the mapping function becomes impracticable. In this subsection, the previous work proposes a grid node replacement algorithm for optimizing distribution localities of data reallocation. According to the relative position of the first of consecutive sub-blocks that produced by each processor, we can determine the best target cluster as candidate for node replacement. Combining with a load balance policy among clusters, this algorithm can effectively improve data localities. Figure 2 gives an example of data reallocation on the cluster grid, which has four clusters. Each cluster provides three processors. The degree of data refinement is set to four ($K = 4$). Figure 2(a) demonstrates an original reallocation communication patterns. We observe that $|I| = 12$ and $|E| = 36$.



(a)



(b)

Figure 2. Communication tables of data reallocation on the identical cluster grid. (C = 4, n = 3, K = 4) (a) Without data mapping. (b) With data mapping.

If we change the distribution of block B to processors reside in *cluster*-2 (P_3, P_4 or P_5) or *cluster*-3 (P_6, P_7 or P_8) in the source distribution, we find that the communications could be centralized in the local cluster for some parts of sub-blocks. Because *cluster*-2 and *cluster*-3 will be allocated the same number of sub-blocks in the target distribution, therefore processors belong to these two clusters have the same priority for node replacement. In this way, P_3 is first assigned to replace P_1. For block C, most sub-blocks will be reallocated to processors in *cluster*-4, therefore the first available node P_9 is assigned to replace P_2. Similar determination is made to block D and results P_1 replace P_3. For block E, *cluster*-2 and *cluster*-3 have the same amount of sub-blocks. Processors belong to these two clusters are candidates for node replacement. However, according to the load balance policy among clusters, *cluster*-2 remains two available processors for the node replacement while *cluster*-3 has three; our algorithm will select P_6 to replace P_4. Figure 2(b) gives the communication tables when applying data to logical grid nodes mapping technique. We obtain $|I|$ = 28 and $|E|$ = 20.

3.2 Non-identical Cluster Grid

Let's consider a more complex example in non-identical cluster grid, the number of nodes in each cluster is different. It needs to add global information of cluster grid into algorithm for estimating the best target cluster as candidate for node replacement. Figure 3 shows a non-identical cluster grid composed by four clusters. The number of processors provided by these *clusters* is 2, 3, 4 and 5, respectively. We also set the degree of refinement as K = 5. Figure 3(a) presents the table of original communication patterns that consists of 19 interior communications and 51 external communications. Applying our node replacement

algorithm, the derived sequence of logical grid nodes is <P_2, P_5, P_9, P_3, P_6, P_{10}, P_4, P_{11}, P_0, P_7, P_{12}, P_1, P_8, P_{13}>. Figure 3(b) gives the communication tables when applying data to logical grid nodes mapping technique. This data to grid nodes mapping produces 46 interior communications and 24 external communications. This result reflects the effectiveness of the node replacement algorithm in term of minimizing inter-cluster communication overheads.

SP \ DP	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}
P_0	a_1	a_2	a_3	a_4	a_5									
P_1						b_1	b_2	b_3	b_4	b_5				
P_2	c_5										c_1	c_2	c_3	c_4
P_3		d_1	d_2	d_3	d_4	d_5								
P_4							e_1	e_2	e_3	e_4	e_5			
P_5	f_4	f_5										f_1	f_2	f_3
P_6			g_1	g_2	g_3	g_4	g_5							
P_7								h_1	h_2	h_3	h_4	h_5		
P_8	i_3	i_4	i_5										i_1	i_2
P_9				j_1	j_2	j_3	j_4	j_5						
P_{10}									k_1	k_2	k_3	k_4	k_5	
P_{11}	l_2	l_3	l_4	l_5										l_1
P_{12}					m_1	m_2	m_3	m_4	m_5					
P_{13}										n_1	n_2	n_3	n_4	n_5
	Cluster-1		*Cluster*-2			*Cluster*-3				*Cluster*-4				

(a)

SP \ DP	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}
P_2	a_1	a_2	a_3	a_4	a_5									
P_5						b_1	b_2	b_3	b_4	b_5				
P_9	c_5										c_1	c_2	c_3	c_4
P_3		d_1	d_2	d_3	d_4	d_5								
P_6							e_1	e_2	e_3	e_4	e_5			
P_{10}	f_4	f_5										f_1	f_2	f_3
P_4			g_1	g_2	g_3	g_4	g_5							
P_{11}								h_1	h_2	h_3	h_4	h_5		
P_0	i_3	i_4	i_5										i_1	i_2
P_7				j_1	j_2	j_3	j_4	j_5						
P_{12}									k_1	k_2	k_3	k_4	k_5	
P_1	l_2	l_3	l_4	l_5										l_1
P_8					m_1	m_2	m_3	m_4	m_5					
P_{13}										n_1	n_2	n_3	n_4	n_5
	Cluster-1		*Cluster*-2			*Cluster*-3				*Cluster*-4				

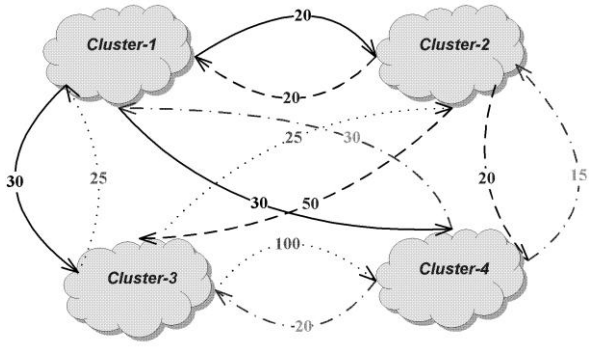
(b)

Figure 3. Communication tables of data reallocation on non-identical cluster grid. (a) Without data mapping. (b) With data mapping.

3.3 Communication Cost of Multi-Clusters with Heterogeneous Network

Examples in the above section do not consider the real communication status for multi-clusters over heterogeneous network communication. Figure 4(a) shows an example of four clusters with various inter-cluster communication costs. Each unit's block data must spend 20 units time from the *cluster*-1 transmission to *cluster*-2, but each unit's block data must spend 30 units time from the *cluster*-1 transmission to *cluster*-3. Figure 4(b) shows the table of inter-cluster communication costs. Therefore, we can calculate communication cost of data distribution for each processor over inter-cluster by this communication matrix. After calculating, the communication cost are 1865 and 885 according to

distribution scheme in Figure 3(a) and 3(b), respectively. But the proposed processor mapping methods provide new sequences of logical grid node which are <P_4, P_5, P_{11}, P_2, P_9, P_0, P_6, P_{10}, P_1, P_7, P_{12}, P_3, P_8, P_{13}> and < P_3, P_5, P_9, P_2, P_{10}, P_1, P_6, P_{11}, P_0, P_7 P_{12}, P_4, P_8, P_{13} > in next section. Consequently, the necessary costs of both sequences are 740 units. The result reflects the effectiveness of this sequence which has the less communications cost. In next section, we will to explain the research model and calculation of communication cost.



(a)

D / S	C_1	C_2	C_3	C_4
C_1	0	20	30	30
C_2	20	0	50	20
C_3	25	25	0	100
C_4	30	15	20	0

(b)

Figure 4. Communication model of Multi-Clusters with Heterogeneous Network. (a) Example of four clusters with various inter-cluster communication costs. (b) The communication matrix table.

3.4 Communication Model of Data Distribution in Multi-Clusters

To set the communication cost of inter-cluster as $V_{(i,j)}$. The communication cost of distribute data block from C_1 to C_3 is denoted $V_{(1,3)}$. Assume there is block A (β=A) from node P of C_i, total cost formula denoted $W(\beta)_{i.}$. $W(\beta)_i = (\beta_1 * V_{(i,1)} + \beta_2 * V_{(i,2)} + \ldots + \beta_j * V_{(i,j)})$. $(1 \leqq i, j \leqq C)$. $\beta_1, \beta_2, \ldots, \beta_{j-1}$ and β_j represent number of sub-blocks that P_i has to send from C_1 to C_1, C_2, …, C_{j-1}, C_j. Figure 5 shows the communication cost of data distribution from each node according to distribution scheme in Figure 4(b). There is the data block A on logic nodes P_0 within a grid model $C = 4$, $K = 5$, <G(4):{2, 3, 4, 5}>. Assume the sub-blocks a_1, a_2 of block A on P_0 needs to be redistributed from C_1 to C_1, the a_3, a_4, a_5 needs to be redistributed from C_1 to C_2, no data is redistributed from C_1 to C_3, C_4. The communication cost of redistributing block A from P_0 and P_2 are $W(A)_1 = (2*0 + 3*20 + 0*30 + 0*30) = 60$ and $W(A)_2 = (2*20 + 3*0 + 0*50 + 0*20) = 40$, respectively. Accordingly, $W(A)_3 = 125$, $W(A)_4 = 105$.

		cluster			
Trad.	BLOCK	C_1	C_2	C_3	C_4
P_0	A	60	40	125	105
P_1	B	150	220	100	80
P_2	C	120	100	425	30
P_3	D	90	70	100	95
P_4	E	150	190	200	60
P_5	F	90	100	350	60
P_6	G	120	100	75	85
P_7	H	150	160	300	40
P_8	I	80	80	275	75
P_9	J	130	150	50	90
P_{10}	K	150	130	400	20
P_{11}	L	70	60	200	90
P_{12}	M	140	200	25	95
P_{13}	N	150	100	500	0

Figure 5. The total communication cost of grid model (C = 4, K = 5, < G (4): {2, 3, 4, 5}>)

4. Processor Mapping Methods

According to communication cost, a candidate processor's id can be chosen according to minimum distribution cost. Therefore, the first processor mapping method is proposed called Processor Mapping using Global Reordering (*GR*). Another method rests on the cluster base, after all data redistribution costs of one cluster are arranged in an order, choosing a candidate processor's id according to the number of processor of its cluster This method is called Processor Mapping using Divide and Conquer Reordering (*DCR*).

4.1 Global Reordering Algorithm

We propose a processor mapping scheme which requires the communication information of inter-cluster. First, the minimum cost is selected using Greedy algorithm. This algorithm, Processor Mapping using Global Reordering (*GR*), is without the complex logic procedures of operation. To achieve the result of processor mapping that has the least communication cost, the key idea is to choose the minimum communication cost from global candidates. The transmission rate between each site over the internet is different because of the various network devices. The cluster can easily measure the transmission rate by the present technology and keep it in each cluster. The system can obtain transmission rates and produce an *n*n* cost matrix. The combination of communication costs can be calculated using the cost matrix and data redistribution pattern. According to the costs, the data block with minimum cost can be chosen to be assigned a processor id first. In the choice process, two kinds of situations occur possibly. To assign a processor id to a data block for distributing: (1) the data block with the chosen minimum cost would be ignored if this data block has already been assigned to another candidate (processor id) previously. (2) if no more processor id can be offered from the selected cluster, the selecting process will continue to find the

next global minimum cost.

To a select processor id for redistributing a data blocks according to the communication cost in Figure 5, *GR* will first select P_{13} for N, P_{10} for K, P_{13} for N, P_8 for M,…, P_0 for F and P_5 for B. A new sequence of logical grid node is provided which is <P_4, P_5, P_{11}, P_2, P_9, P_0, P_6, P_{10}, P_1, P_7, P_{12}, P_3, P_8, P_{13}> and the necessary communication cost is 740 units, accordingly.

According to the method described above, the code of algorithm is shown as follows:

```
For P=0 to n-1
    Determine how many cost matrix t in
    every cluster
EndFor
Order by t
While (Replacement s not complete)
    If (cluster have processor)
        select target cluster processor id P
        that has minimum cost from t
    EndIf
EndWhile
```

Figure 6. Processor Mapping using Global Reordering Algorithm.

4.2 Divide and Conquer Reordering Algorithm

The proposed method is introduced in this section called Processor Mapping using Divide and Conquer Reordering Algorithm (*DCR*). The *GR* method employs the greedy algorithm to choose the minimum cost for processor mapping. The *DCR* method uses the Greedy algorithm to choose processor id for data block with minimum cost for each cluster first. The number of selected data block is equal to the number of processors provide in each cluster. Due to select several data blocks for each cluster with minimum cost and a processor id, cross match is not under consideration. Certainly, the results of processor mapping will not be perfect. Namely, conflict selections will possibly happen. To resolve the conflict situations, the conflict part can be regarded as a sub-grid model of the original grid model. Data blocks without conflict situation and selected processor id are excluded. *DCR* employs *GR* method to select processor id for rest of data blocks for a complete result.

To select data blocks with minimum cost for each cluster according to the communication cost in Figure 5, *DCR* will select *A* and *L* for C_1, *A*, *D* and *L* for C_2, *B*, *G*, *J* and *M* for C_3, and *C*, *E*, *H*, *K* and *N* for C_4. After select processor id for *B*, *C*, *D*, *E*, *G*, *H*, *J*, *K*, *M* and *N*, DCR employs GR to select processor id for *A*, *F*, *I* and *L* again. Then, a new sequence of logical grid node is provided which is < P_3, P_5, P_9, P_2, P_{10}, P_1, P_6,

P_{11}, P_0, P_7 P_{12}, P_4, P_8, P_{13} >. Accordingly, the necessary communication cost is 740 units.

According to the method described above, the code of algorithm is shown as follows:

```
For P=0 to n-1
    Determine how many cost matrix t on every cluster
EndFor
For t=1 to C
    Order by cost from t
EndFor
While (Replacement s not complete)
    For P=0 to n-1
        If not (two or more cluster have the candidate)
            select the target cluster processor id P that has the minimum
cost
        EndIf
    EndFor
    reorder the remaining cost list from t
    select the target cluster processor id P by GR Algorithm
EndWhile
```

Figure 7. Processor Mapping using Divide and Conquer Reordering Algorithm.

5. Performance Evaluation

In this section, proposed techniques and methods without considering actual communication cost are implemented to simulate with different communication cost matrixes. The network bandwidth is different from 10Mb to 1Gb for heterogeneous network environment. Since 10Mb network equipments are almost eliminated, the value of transmission ratio is set from 10 to 30. The value is randomly produced to simulate patterns of communication cost matrix. The variance is set from 150 to 450 in simulations representing of network heterogeneity. The larger number of variance represents the larger network heterogeneity. Besides, *C* is set from 8 to 16, *K* is set from 16 to 64 for simulations. 100 difference communication cost matrix patterns are used to calculate communication costs for each variance case and average of the costs is the results of the theoretical value. The following figures show the results of each method.

Figure 8 shows the results on a grid consisted of 8 cluster, <G(8):{4, 4, 4, 6, 6, 6, 8, 8}> and *K* is equal to 16. Figure 8 illustrates the comparing results of four different methods. Original one does not consider the actual cost of reordering communications technology, *GR* is Processor Mapping using Global Reordering technology, *DCR* for the Processor Mapping using Divide and Conquer Reordering technology. Obviously, *GR* and *DCR* have less communication cost compared with the other two models. When the difference in the number of 150, *GR* and *DCR* can reduce about 33% cost compared with the traditional one which is without processor reordering. Both of them also reduce 6% communications cost while comparing with the Original one. While the variance is 450, the improvement slightly increases about 33% ~ 36%.

Figure 9 shows the results of the grid model with C = 16, K = 64 and <G(16):{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}> while comparing four different methods. Obviously, *GR* and *DCR* have less communication cost comparing with the other two models. *GR* and *DCR* can reduce about 26% to 29% cost while comparing with the traditional one which is without processor reordering. Both of them also reduce 11% communications cost while comparing with the Original one. Above simulation results show the proposed reordering technologies not only outperform previous processor reordering method but also successfully reduce communication cost on the heterogeneous network and improve the communication cost

6. Conclusions

In this paper, we have presented a generalized processor reordering method for communication localization with heterogeneous network. The methods of processor mapping technique are employed to enhance the performance of parallel programs on a cluster grid. Contribution of the proposed technique is to reduce inter-cluster communication overheads and to speed up the execution of data parallel programs in the underlying distributed cluster grid. The theoretical analysis and results show improvement of communication costs and scalable of the proposed techniques on multi-clusters with heterogeneous network environment.

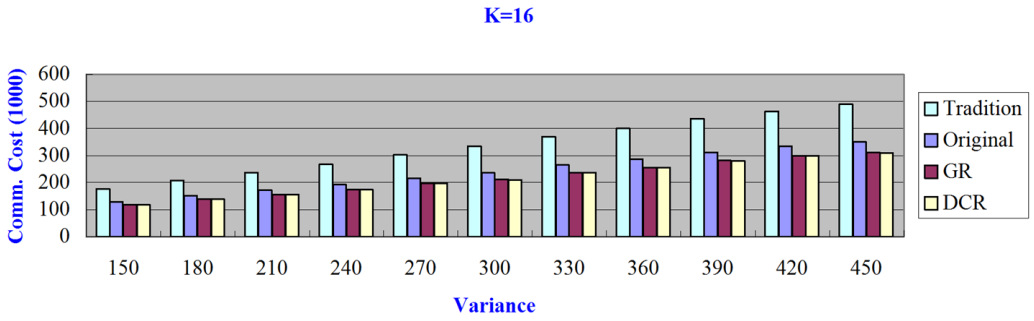


Figure 8. Communication costs comparison with C = 8, K = 16, <G(8):{4, 4, 4, 6, 6, 6, 8, 8}>.

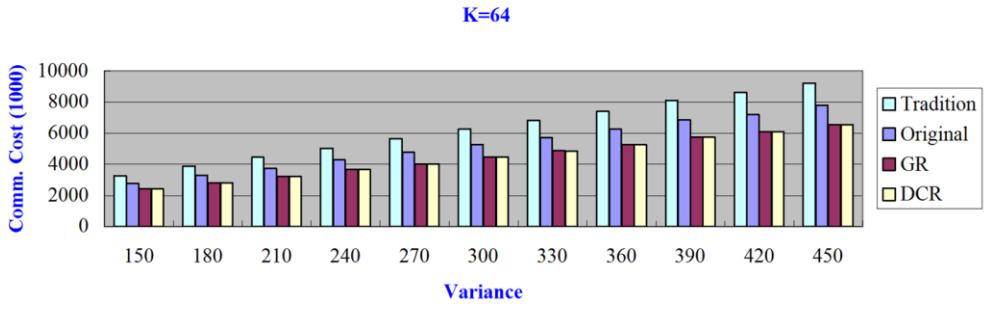


Figure 9. Communication costs comparison with C = 16, K = 64 and <G(16):{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}>

REFERENCES

[13] O. Beaumont, A. Legrand and Y. Robert, "Optimal algorithms for scheduling divisible workloads on heterogeneous systems," *Proceedings of the 12th IEEE Heterogeneous Computing Workshop*, 2003.

[14] J. Dawson and P. Strazdins, "Optimizing User-Level Communication Patterns on the Fujitsu AP3000," *Proceedings of the 1st IEEE International Workshop on Cluster Computing*, pp. 105-111, 1999.

[15] Henri E. Bal, Aske Plaat, Mirjam G. Bakker, Peter Dozy, and Rutger F.H. Hofman, "Optimizing Parallel Applications for Wide-Area Clusters," *Proceedings of the 12th International Parallel Processing Symposium IPPS'98*, pp 784-790, 1998.

[16] M. Faerman, A. Birnbaum, H. Casanova and F. Berman, "Resource Allocation for Steerable Parallel Parameter Searches," *Proceedings of GRID'02*, 2002.

[17] I. Foster and C. Kessclman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, ISBN 1-55860-475-8, 1999.

[18] James Frey, Todd Tannenbaum, M. Livny, I. Foster and S. Tuccke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," *Journal of Cluster Computing*, vol. 5, pp. 237 – 246, 2002.

[19] M. Guo and I. Nakata, "A Framework for Efficient Data Redistribution on Distributed Memory Multicomputers," *The Journal of Supercomputing*, vol.20, no.3, pp. 243-265, 2001.

[20] Jih-Woei Huang and Chih-Ping Chu, "An Efficient Communication Scheduling Method for the Processor Mapping Technique Applied Data Redistribution," *The Journal of Supercomputing*, vol. 37, no. 3, pp. 297-318, 2006

[21] Ching-Hsien Hsu, Guan-Hao Lin, Kuan-Ching Li and Chao-Tung Yang, "Localization Techniques for Cluster-Based Data Grid," *Proceedings of the 6th ICA3PP*, Melbourne, Australia, 2005

[22] Ching-Hsien Hsu, Tzu-Tai Lo and Kun-Ming Yu "Localized Communications of Data Parallel Programs on Multi-cluster Grid Systems," European Grid Conference, LNCS 3470, pp. 900 – 910, 2005.

[23] Florin Isaila and Walter F. Tichy, "Mapping Functions and Data Redistribution for Parallel Files," *Proceedings of IPDPS 2002 Workshop on Parallel and Distributed Scientific and Engineering Computing with Applications, Fort Lauderdale*, April 2002.

[24] Bahman Javadi, Mohammad K. Akbari and Jemal H. Abawajy, "Performance Analysis of Heterogeneous Multi-Cluster Systems," *Proceedings of ICPP*, 2005

[25] Bahman Javadi, J.H. Abawajy and Mohammad K. Akbari "Performance Analysis of Interconnection Networks for Multi-cluster Systems" *Proceedings of the 6th ICCS*, LNCS 3516, pp. 205 – 212, 2005.

[26] Jens Koonp and Eduard Mehofer, "Distribution assignment placement: Effective optimization of redistribution costs," *IEEE TPDS*, vol. 13, no. 6, June 2002.

[27] E. T. Kalns and L. M. Ni, "Processor mapping techniques toward efficient data redistribution," *IEEE TPDS*, vol. 6, no. 12, pp. 1234-1247, 1995.

[28] Jong Sik Lee, "Data Distribution Management Modeling and Implementation on Computational Grid," *Proceedings of the 4th GCC*, Beijing, China, 2005.

[29] Saeri Lee, Hyun-Gyoo Yook, Mi-Soon Koo and Myong-Soon Park, "Processor reordering algorithms toward efficient GEN_BLOCK redistribution," *Proceedings of the 2001 ACM symposium on Applied computing*, 2001.

[30] Y. W. Lim, P. B. Bhat and V. K. Parsanna, "Efficient algorithm for block-cyclic redistribution of arrays," *Algorithmica*, vol. 24, no. 3-4, pp. 298-330, 1999.

[31] Aske Plaat, Henri E. Bal, and Rutger F.H. Hofman, "Sensitivity of Parallel Applications to Large Differences in Bandwidth and Latency in Two-Layer Interconnects," *Proceedings of the 5th IEEE High Performance Computer Architecture HPCA'99*, pp. 244-253, 1999.

[32] Xiao Qin and Hong Jiang, "Dynamic, Reliability-driven Scheduling of Parallel Real-time Jobs in Heterogeneous Systems," *Proceedings of the 30th ICPP*, Valencia, Spain, 2001.

[33] S. Ranaweera and Dharma P. Agrawal, "Scheduling of Periodic Time Critical Applications for Pipelined Execution on Heterogeneous Systems," *Proceedings of the 30th ICPP*, Valencia, Spain, 2001.

[34] D.P. Spooner, S.A. Jarvis, J. Caoy, S. Saini and G.R. Nudd, "Local Grid Scheduling Techniques using Performance Prediction," *IEE Proc. Computers and Digital Techniques*, 150(2): 87-96, 2003.

出席國際學術會議心得報告

計畫名稱	應用P2P與Web技術發展以SOA為基礎的格網中介軟體與經濟模型
計畫編號	NSC 97-2628-E-216-006-MY3
報告人姓名	許慶賢
服務機構及職稱	中華大學資訊工程學系教授
會議名稱	**The 4th International ICST Conference on Scalable Information Systems (INFOSCALE 2009)**
會議/訪問時間地點	香港 / 2009.06.09-11
發表論文題目	**Power Consumption Optimization of MPI Programs on Multi-Core Clusters**

參加會議經過

會議時間	行程敘述
2009/06/10	(上午) 8:30 會場報到 9:00 聆聽Keynote Speech Reevaluating Amdahl's Law in the Multicore Era Xian-He Sun, Illinois Institute of Technology, Chicago, USA 10:30 發表論文、聽取其它論文發表 (下午) 1:30 聆聽Keynote Speech Metropolitan VANET: Services on the Road Minglu Li, Shanghai Jiao Tong University, China 2:00 聽取 Resource Allocation and Application相關論文發表 3:45 主持 Session (晚上) 6:30 參加晚宴

2009/06/11	(上午) 9:00 聆聽Keynote Speech Autonomic Cloud Systems Management: Challenge and Opportunities Cheng-Zhong Xu, Wayne State University, USA 10:30聽取 Information Security相關論文發表 (下午) 1:30聽取 Parallel and Distributed Computing相關論文發表 3:30聽取 RFID / Sensor Network 相關論文發表

這一次在香港所舉行的國際學術研討會議共計兩天。第一天上午由 Dr. Xian-He Sun (Illinois Institute of Technology, China) 針對 The current Multi-core architecture and memory-wall problem，作為此次研討會的開始，下午由 Dr. Minglu Li （Shanghai Jiao Tong University, China）針對 The application of mobile communication technology 給予專題演講。下午接著是一個場次進行。本人聽取 session 3 的相關論文發表，也擔任主持第一天 session 3 的論文發表。晚上在大會的地點與幾位國外學者及中國、香港教授交換心得意見。第二天，專題演講是由 Dr. Cheng-Zhong Xu (Central Michigan University, USA) 針對 "Embedded Software Development with MDA"發表演說。本人也參與的第二天全部的大會議程，這天由 Cheng-Zhong Xu （Wayne State University, USA）。這一天，也發表了這一次的論文。本人主要聽取 Multi-Core 等相關研究，同時獲悉許多新興起的研究主題，並了解目前國外大多數學者主要的研究方向，並且把握最後一天的機會與國外的教授認識，希望能夠讓他們加深對台灣研究的印象。二天下來，本人聽了許多優秀的論文發表。這些研究所涵蓋的主題包含有：無線網路技術、網路安全、Multi-Core、資料庫以及普及運算等等熱門的研究課題。此次的國際學術研討會議有許多知名學者的參與，讓每一位參加這個會議的人士都能夠得到國際上最新的技術與資訊。是一次非常成功的學術研討會。

Power Consumption Optimization of MPI Programs on Multi-Core Clusters

Ching-Hsien Hsu and Yen-Jun Chen

Abstract

While the energy crisis and the environmental pollution become important global issues, the power consumption researching brings to computer sciences world. In this generation, high speed CPU structures include multi-core CPU have been provided to bring more computational cycles yet efficiently managing power the system needs. Cluster of SMPs and Multi-core CPUs are designed to bring more computational cycles in a sole computing platform, unavoidable extra energy consumption in loading jobs is incurred.

Data exchange among nodes is essential and needed during the execution of parallel applications in cluster environments. Popular networking technologies used are Fast Ethernet or Gigabit Ethernet, which are cheaper and much slower when compared to Infiniband or 10G Ethernet. Two questions on data exchange among nodes arise in multi-core CPU cluster environments. The former one is, if data are sent between two nodes, the network latency takes longer than system bus inside of a multi-core CPU, and thus, wait-for-sending data are blocked in cache. And the latter is, if a core keeps in waiting state, the unpredicted waiting time brings to cores higher load. These two situations consume extra power and no additional contribution for increasing overall speed. In this paper, we present a novel approach to tackle the congestion problem and taking into consideration energy in general network environments, by combining hardware power saving function, maintaining the transmission unchanged while saving more energy than any general and previous cases.

1. Introduction

Reduction on power consumption of computer systems is a hot issue recently, since many CPUs and computer-related hardware has been produced and under operation everywhere. As the number of single-core CPU has reached to physical limitation on current semi-conductor technology, the computing performance has met the bottleneck. Multi-core CPUs become a simple yet efficient solution to increase performance and speed since that concept SMP in a single chip, that is, making up a small cluster to be executed inside a host. Additionally, it reduces the amount of context switching while in single-core CPUs, increases straight forwardly the overall performance. Some CPU technologies and our target will be introduced in below.

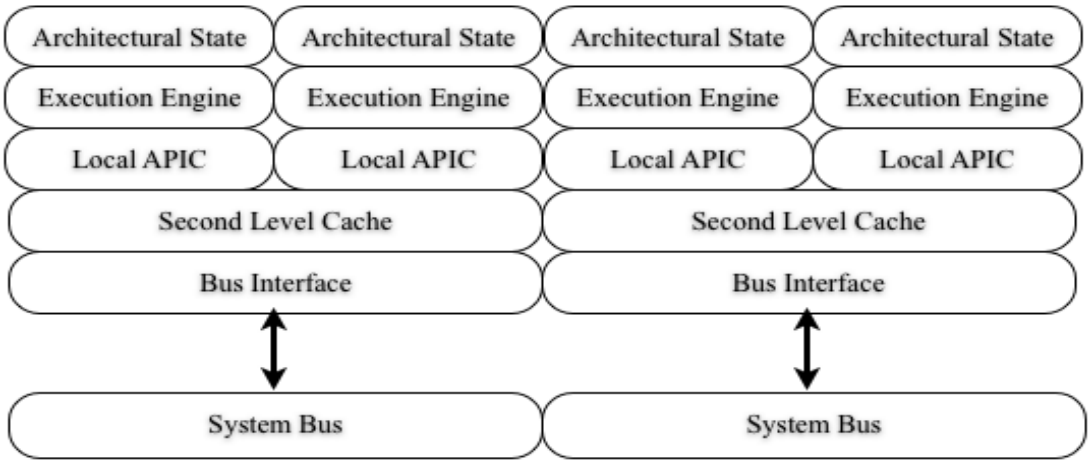


Figure 1: Intel Quad-Core CPU system structure [11]

Figure 1 illustrates the architecture of Intel quad-core CPU, which looks like a combination of two dual-core CPUs. It has four individual execution engines, where each two cores share one set of L2 cache and system bus interface, and connect to the fixed system bus. The advantages of this architecture are twofold. The former one is that each core can fully utilize L2 cache as each core needs larger memory, while the latter is that each core accesses L2 cache through individual hub [7] simplifying system bus and cache memory structures. Intel CPU provides “SpeedStep” [3] technology that helps to control CPU frequency and voltage, and it needs to change all cores’ frequency at the same.

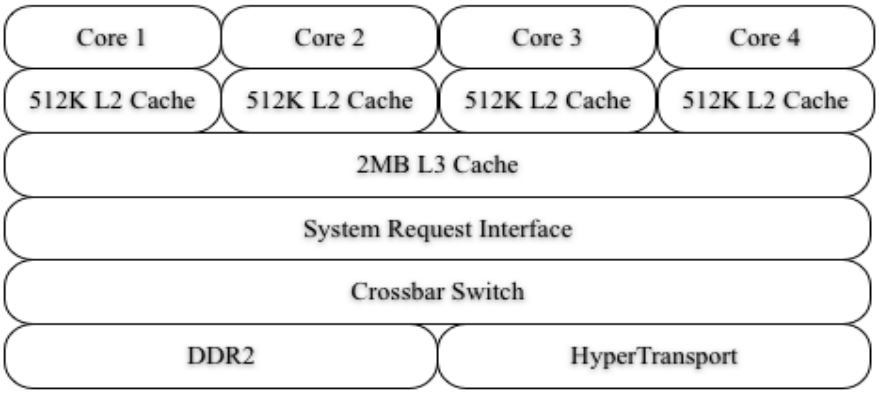


Figure 2: AMD Quad-Core CPU system structure [12]

AMD quad-core CPU, as shown in Figure 2, has individual L2 cache in each core and share L3 cache, (a special design), and then integrated to DDR2 memory controller into CPU, helping to increase memory access speed. Each core has individual channel to access system bus, and L3 cache and peripheral chips from crossbar switch. AMD provides "PowerNow!" [4] technology to adjust each core's working frequency / voltage.

A cluster platform is built up by interconnecting a number of single-core CPU, and a message passing library, such as MPI is needed for data exchange among computing nodes in this distribution computing environment. In addition, high speed network as Infiniband is needed to interconnect the computing nodes. As multi-core CPUs are introduced and built in cluster environments, the architecture of this newly proposed cluster is as presented in Figure 3. The main advantages of data exchanges between cores inside of a CPU is much faster than passing by a network and South / North bridge chip.

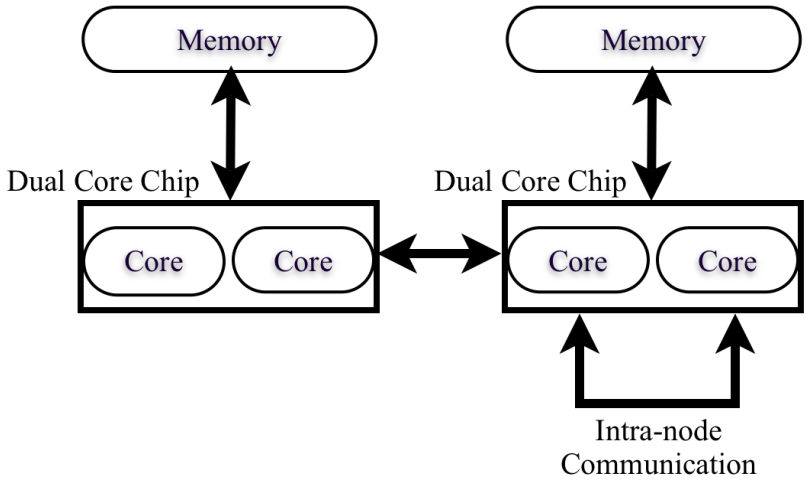


Figure 3: Multi-core based cluster structure [13]

Developed from 1999, InfiniBand [16] is a point-to-point structure, original design concept that focused on high-performance computing support, so bidirectional serial fiber interface, failover mechanism and scalable ability are the necessary functions. InfiniBand supports at least 2.5Gbit/s bandwidth in each direction in single data rate (SDR) mode, the transmitted information includes 2 Gbit useful data and 500Mbit control commands. Besides, InfiniBand supports DDR (Double Data Rate) and QDR (Quad Data Rate) transmission mode, and each mode supports 3 different speed (1x, 4x and 12x) configurations, so the maximum bandwidth is 96Gbit/s. The detail specification is as Table 1.

Table 2: Infiniband transmission mode list

	Single (SDR)	Double (DDR)	Quad (QDR)
1X	2 Gbit/s	4 Gbit/s	8 Gbit/s
4X	8 Gbit/s	16 Gbit/s	32 Gbit/s
12X	24 Gbit/s	48 Gbit/s	96 Gbit/s

Infiniband networking technology is a good and fast enough solution to connect all computing nodes of a cluster platform, but expensive. Gigabit Ethernet is cheaper solution and widely built in general network environment, though slower in transmission speed and definitely drop down data exchange performance. To

send data to a core that is inside of a different host will be needed to consume extra energy when waiting for data.

"SpeedStep" and "PowerNow!" technologies are good solutions to reduce power consumption, since they adjust CPU's frequency and voltage dynamically to save energy. The power consumption can be calculated by the function:

$$P=IV=V^2f=J/s. \quad (1)$$

where P is Watt, V is voltage, I is current, f is working frequency of CPU, J is joule and s is time in seconds. It means that lower voltage in the same current condition saves more energy. How and when to reduce voltage and frequency become an important issue, since one of main targets of clustering computing computers is to increase the performance, while slowing down CPU's frequency is conflict with performance. Considering data latency of network, and CPU load in current CPU technologies, we would like to create a low energy cost cluster platform based on general network architecture, that keeps almost the same data transmission time though lower in energy consumption when CPU in full speed.

To address the above questions, we use OpenMPI and multi-core CPU to build up a Linux based a low energy cost cluster, and implement three solutions on this environment.

- CPU power consumption reduction

  Drive CPU power saving technology to reduce working frequency when low working loading, the method reduces unavailable power consumption.

- CPU internal bus congestion reduction

  Add waiting time between each data frame before send out, the method slows down data transmission speed and reduces core working loading.

- Loading-Aware Dispatching (LAD) Algorithm

Lower loading core is indicated higher priority to receive data frame, the method increases core working efficiency.

2. Related Work

Based on the concept about reducing computing time, the job scheduling methodology as introduced in [8] and [21] was designed targeting for a faster complete data transmission; otherwise, adjust cache block size to find the fastest speed that transmits data using MPI between MPI nodes in situations as listed in [13] was studied, and similar implementation of the method using OpenMP was also observed in [14]. Another investigation focused on compiler that analyze program's semantics, and insert special hardware control command that automatically adjusts simulation board's working frequency and voltage, [10] research needs to be combined both hardware and software resources.

Some kinds of paper designed their methodologies or solutions under simulation board, or called NoC system, as shown in its structure is as below:

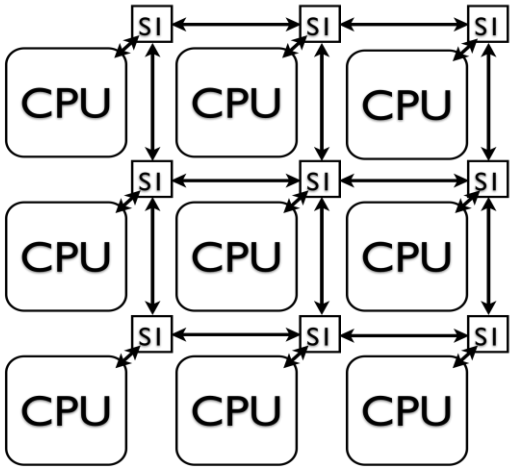


Figure 4: General NoC system structure

Base on a simulation board, researchers have designed routing path algorithm that tries to find a shortest path to transmit data in Networks-on-Chip [15], in order to reduce data transmission time between CPUs, as also to have opportunities to realistically port and implement it to a cluster environment.

Others, researches have applied Genetic Algorithms to make a dynamically and continuous improvement on power saving methodology [9]. Through a software based methodology, routing paths are modified, link speed and working voltage are monitored and modified at the same time to reduce power consumption of whole simulation board, while the voltage detection information required hardware support.

Consider higher power density and thermal hotspots happened in NoC, the paper [18] provided a compiler-based approach to balances the processor workload, these researchers partitions a NoC system to several area and dispathes jobs to them by node remapping, the strategy reduces the chances of thermal concentration at runtime situation, and brings benefit about a bit of performance increasing. The paper [20] and [24] studied the same point about thermal control.

Modern Operating Systems as Linux and Windows provides hardware power saving function as introduced in [1] and [2], where they can drive “SpeedStep” [3] and “PowerNow!” [4] utilizing special driver to control CPU voltage and frequency. Of course hardware support is necessary, since depending on the CPU loading, CPU is automatically selected with lower frequency and voltage automatically. Besides, someone add management system into OS kernel to control energy consumption directly [22].

The peripheral devices of computer, as disk subsystem is a high energy consumption hardware, the paper [17] studied how to implement disk energy optimization in compiler, these researcheers considered disk start time, idle time, spindle speed, the disk accessing frequency of program and CPU / core number of each host, made up a benchmark system and real test environment to verify physical result.

Some groups study the power saving strategy implementation in data center, as database or search engine server. Huge energy is consumed by this kind of application when they have no work. The nearer research [19] provides a hardware based solution to detect idle time power waste and designs a novel power supplier

operation method, the approach applied in enterprise-scale commercial deployments and saved 74% power consumption.

Besides, some researchers studied OS resource management and power consumption evaluation and task scheduling method as [23] and [25], this kind of study provides a direction to optimize computer operation tuning and reduces system idle time that brings by resource waiting.

3. Challenges of Power Saving in Multi-Core Based Cluster

In the previous single-core CPU based cluster environment, data distribution with CPU energy control are easier to implement by isolated CPU frequency control of each host. In multi-core based cluster, CPU internal bus architecture, bandwidth and power control structure bring differnet challanges in this issue. When we built a cluster platform that combines some key technologies as listed in Chapter 1 for experiment purposes, their advantages bring higher speed for data transmission peformance, yet only between cores inside a CPU, a CPU core is maintained with high load means the CPU speed cannot be decreased. Analysis and reasoning on these situations are discussed next.

The “SpeedStep” and “PowerNow!” were not show in Figure 1 and 2. The “SpeedStep” provides solely full CPU frequency and voltage adjustment. The design makes power control easier, though consumes extra energy. If only one core works with high load, power control mechanism cannot reduce other cores’ frequency / voltage, nor dropping down the performance of a busy core. Inefficient energy consumption brings temperature increasing, since low loading core generates the same heat as high load one, and brings the CPU’s temperature up at the same time.

AMD “PowerNow!” shows advantage in this issue, since we can reduce frequency when core works in lower loading without need to consider other cores’ situation, and heat reduction is also another benefit.

As description of Figure 1, Intel’s CPU architecture shares L2 cache to cores using individual hub, all packets between core and cache needs to pass through by it. The architecture has 2 advantages and 2 problems:

Advantages

- **Flexible cache allocation**

Every core was allowed to use whole L2 cache from cache hub, the hub provides single memory access channel for each core, and hub assigns cache memory space to requested core. The method simplifies internal cache access structure.

- **Decrease cache missing rate**

When each core has massive cache request all of a sudden, flexible cache memory allocation provides larger space to save data frame, and also decreases page swapping from main memory at the same time.

Problems

- **Cache Hub Congestion**

If huge amount of data request or sending commands happen suddenly, individual cache hub blocks data frames in cache memory or stops commands in queue. All cores and hub keep in busy state and thus consume extra energy.

- **Network Bandwidth Condition**

Lower network bandwidth makes previous situation more seriously in many nodes' cluster, since network speed cannot be as fast as internal CPU bus, if cross-node data frames appear, the delivering time is longer than intra-node data switch.

Compared with Intel, while data frame flood sends to CPU, AMD structure has no enough cache to save them, yet individual bus / memory access channel of each core provides isolated bandwidth, L2 cache built

in core reduces data flow interference. Different CPU structure provides their advantages, and weakness appears while they are compared to each other.

In a general situation, each computing node executed under a given core / host randomly indicated by cluster software, signifies that programmer cannot obtain additional core loading from node's code section. Following our purpose, finding system information about thread / node location works, but it is a hard method since the program would spend large amount of time in device I/O, includes open system state file, analysis information and obtaining node's location. Another alternative method is easier, where we make cluster platform that fixes node location in indicated core or host, and the function helps to get core loading from node's code. OpenMPI is selected for this issue.

4. The Proposed Approach

Upon with CPU specification, CPU power control interface and network structure, we provide a novel data dispatching strategy to solve the previous challanges in Chapter 3, it combines data flow limitation, core frequency controlling, and accords core working load to transmit data frame, detail operation is as below.

It is not a good method to keep performance. In fact, we add 1μs delay between two packets, in a real environment, and the total transmission time is added as:

$$T = N \times D \qquad (2)$$

where T is total time, N is total number of packets and D is delay time between packets. We found that the total time has just been added less than one to four seconds in average, when is transmitted 100K data frames across two hosts that are connected via Gigabit Ethernet. Additionally, the advantage is that the loading of a central node that sends data to other nodes is decreased by almost 50%. On the other hand, data receiving core load is decreased by 15% in average when we added 10μs delay in these nodes, follow Function 2, the amoung of increased delay time should be 1s, yet in experiment result, total transmission time is increased by

less than 0.5s. These experiment results means the core work loading brings up by massive data frame, not by CPU bound process. This method reduces core work loading and helps below method to operate.

Although the challenge presented in section 3.1 exists, as for power saving issue, we use AMD system and "PowerNow!" to slow down lowering loading core frequency. The given CPU supports two steps frequency, and therefore they work in different voltage and current. Thus we focus on frequency adjustment, and calculating power consumption of each core as below:

$$P = V_{max} \times I_{max} \times T \quad (3)$$

where $V_{\max}$ and $I_{\max}$ are found from AMD CPU technology specification [6], and T is program execution time. Since "Time" joins the function, the unit of P is Joule.

There is a CPU frequency controlling software: CPUFreq. It provides simple commands to change CPU work state and 4 default operation modes:

- Performance mode: CPU works in highest frequency always
- Powersave mode: CPU works in lowest frequency always
- OnDemand mode: CPU frequency is adjusted following CPU work loading
- UserSpace mode: User is permitted to change CPU frequency manually follow CPU specification

We have used UserSpace mode and got the best CPU work loading threshold range to change CPU frequency: 75%~80%, if CPU work loading lower than this, we reduce frequency; if higher, we increase frequency. But actually, the default threshold of OnDemand mode is 80%, so we use OnDemand mode to control CPU frequency when our data dispatching method is executed.

Following the previous results, working with OnDemand mode of CPUFreq, we provide a Loading-Aware Dispatching method (LAD). Based on the AMD "PowerNow!" hardware structure, and keeping the same load on all cores is necessary for efficient energy consumption, thus sending data from central node to lowest loading node makes sense. If the load can be reduced on a core, then reducing CPU frequency is permitted for saving energy.

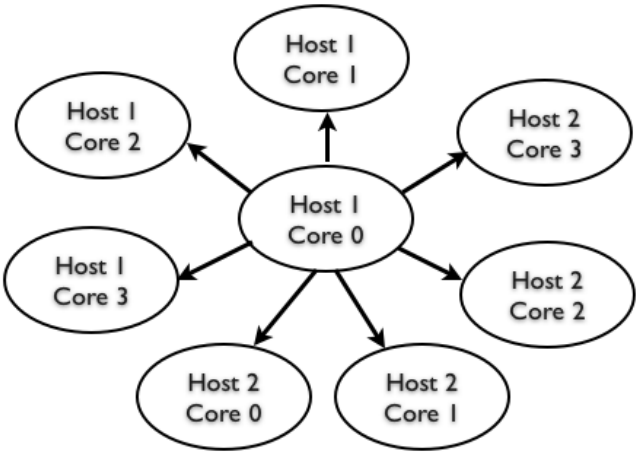


Figure 5: LAD Algorithm structure diagram

Still in LAD algorithm, as indicated in Figure 4, data frames are sent sequentially from Host 1-Core 0 to other cores. This method is often used to distribute wait-for-calculate data blocks in complex math parallel calculations. MPI provides broadcast command to distribute data and reduce command to receive result. In order to changing data frame transmission path dynamically, we use point-to-point command to switch data, since this type of command can indicate sending and receiving node.

The detail of operation flow is as below:

- Step 1: Detect core loading
- Step 2: Find lowest loading core
- Step 3: Send several data frames to the lowest loading core
- Step 4: Repeat previous two step until all data frames are transmitted over

The data distribution algorithm is given as below.

Loading-Aware Dispatching (LAD)Algorithm

```
1.  generating wait-for-send data frame
2.  if (node 0)
3.  {
4.      //send data follow sorting result
5.      while(!DataSendingFinish)
6.      {
7.          //detect nodes' loading from system information and save in TargetNode
8.          OpenCPUState;
9.          CalculateCPULoading;
```

```
10.          //sort TargetNode from low to high
11.          CPULoadingSorting;
12.          //send 1000 data frame
13.          for(i=1; i<1000; i++)
14.              SendData(TargetNode[i]);
15.          if(whole data transmitted)
16.              DataSendingFinish=true;
17.      }
18.      //send finish message to receiving nodes
19.      for(i=1; i<NodeNumber; i++)
20.          SendData(i);
21.  }
22.  if (other nodes)
23.  {
24.      //receive data from node 0
25.      ReceiveData(0);
26.      usleep();
27.  }
```

5. Performance Evaluation and Analysis

In this chapter, experimental environment and results of LAD algorithm is presented. The cluster platform includes two computing nodes and connected via Gigabit Ethernet, and each node is installed with Ubuntu Linux 8.10 / kernel 2.6.27-9, OpenMPI message passing library is selected for thread execution affinity function, the hardware specification is listed as next.

Table 3: Host specification

CPU	AMD Phenom X4 9650 Quad-Core 2.3GHz
Layer 1 Cache	64K Instruction Cache and 64K Data Cache Per Core
Layer 2 Cache	512K Per Core
Layer 3 Cache	Share 2M for 4 Cores
Main Memory	DDR2-800 4GB

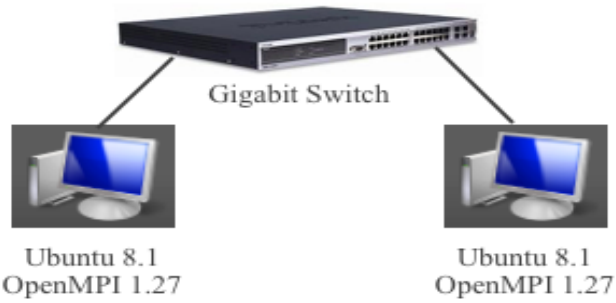


Figure 6: Test environment

Data frame size

Three different sizes of data frames are transmitted between nodes: one byte, 1460 bytes and 8000 bytes. One byte frame is not only the smallest one in MPI data frame, but also in network, for complete data transmission in shortest time, source node generates huge amount of one byte frame, these packets congest CPU internal bus and network.

1518 bytes frame is the largest one in network, but considering that network header should be inserted into network packet, we select 1460 bytes frame for testing, and then, this size of packet brings largest amount of data in a single network packet, and trigger fewest network driver interrupt to CPU. Finally 8000 bytes frame is set for large data frame testing, since it needs to be separated to several other packets by network driver for transmission, but not necessary to be separated in intra-node, and thus need the longest time for data transmission.

While the experiment is executed, we send 100K data frames between two nodes, and calculate the power consumption.

CPU frequency and packet delay

Each experiment result figures and tables that follows next has four blocks. The first one is executed in Performance Mode (PM, CPU works in 2.3GHz), the second one is PowerSave Mode (PS, 1.15GHz), the

third one is OnDemand Mode (OD, slows down frequency while CPU loading lower than 80%), and last one is LAD algorithm that works with OnDemand Mode.

Besides, each block has four delay time configurations, the first one contains no delay between each data frame, the second delays 5µs, the third one delays 10µs, and last one delay 20µs. Still in figures that follows next, TD stands for Transmission Delay, Transmission Time as TT, and PC for Power Consumption.

Rank number

The "Rank Number" in each figures and tables mean the number of nodes / cores join data dispatching. For example, rank 2 means rank 0 dispatchs data to rank one, and rank 4 means rank 0 dispatchs data to rank one, 2, and 3. Since each host has four cores, the rank number 2~4 are internal node data transmission, and rank 5~8 are cross node data transmission.

Although only four cores join work in rank number 2~4, other cores consume energy at the same time, and we still need to add the energy consumed.

One byte frame

Table 3 and Figure 5 show the TT for one byte frame, and Figure 6 the PC. Comparing PM, PS and OD mode, we find that TD increases the TT over 3 seconds in rank 2~4 in every frequency level, but increases less than one second in 5~8. Table 4 and Figure 6 displayed one byte frame PC. Clearly, the PS mode spends the longest time to transmit data, though consumes the lowest energy. OD mode has none remarkable performance in power saving in rank 7~8, but it uses average 100J less than PM mode in rank 2~6, and keeps TT increasing less than 0.4s in cross-node situation. LAD algorithm displays advantage in no delay situation, less than 1s TT increasing yet consumes almost the same energy in rank 7~8. In other situations, LAD spends maximum 4s longer than OD mode, and saves 400J.

Table 4: Detail results of time effect of TD on TT (Frame = 1 Byte)

Mode & TD \ Rank Number		2	3	4	5	6	7	8
PM mode	0	0.160	0.333	0.501	6.262	10.646	15.072	18.630
	5	0.928	1.152	1.286	6.813	11.276	15.867	19.384
	10	2.292	2.271	1.775	6.655	11.076	15.419	19.238
	20	3.251	3.229	3.216	6.924	11.083	15.603	19.032
PS mode	0	0.285	0.576	0.909	9.984	16.537	23.151	28.976
	5	1.326	1.689	1.935	10.599	17.311	23.679	29.518
	10	2.637	2.174	2.429	10.580	17.848	24.157	29.598
	20	3.612	6.651	3.850	10.767	17.470	24.165	29.651
OD mode	0	0.216	0.372	0.531	6.625	11.388	16.025	18.664
	5	1.330	1.625	1.824	7.143	11.973	16.863	19.503
	10	2.630	2.126	2.256	6.898	11.693	16.456	19.456
	20	3.489	3.683	3.756	7.343	11.723	16.615	19.161
LAD	0	0.288	0.577	0.918	7.182	12.018	16.699	19.524
	5	1.367	1.423	1.623	8.716	14.587	20.181	20.704
	10	2.659	1.960	2.028	8.718	14.508	20.221	21.355
	20	3.598	3.813	3.716	9.254	15.129	20.253	22.943

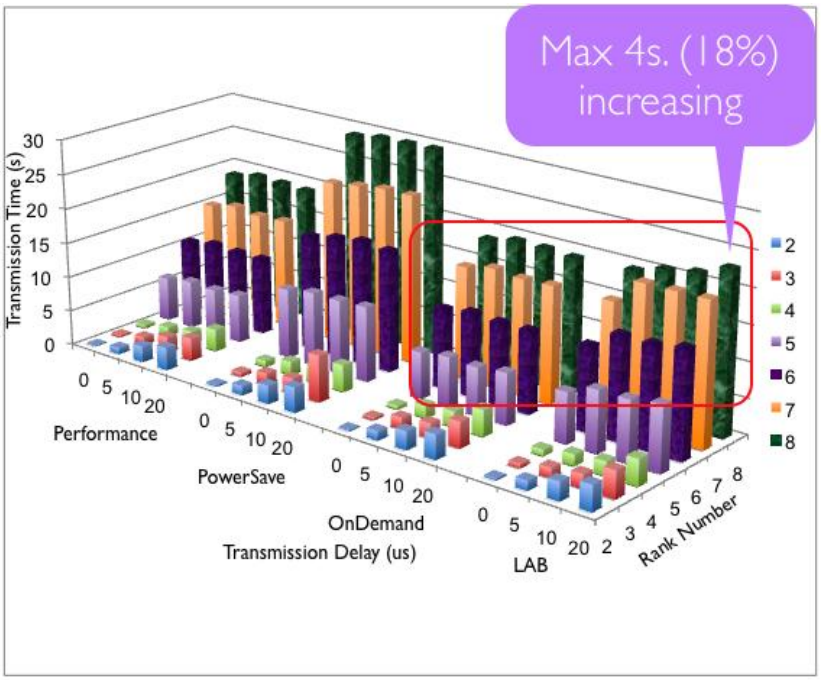


Figure 7: Time effect of TD on TT (Frame = 1 Byte)

Table 5: Detail results of power effect of TD on PC (Frame = one Byte)

Mode & TD \ Rank Number		2	3	4	5	6	7	8
PM mode	0	25.400	52.864	79.535	994.105	1690.074	2392.710	2957.550
	5	147.322	182.882	204.155	1081.577	1790.088	2518.918	3077.249
	10	363.860	360.526	281.785	1056.495	1758.337	2447.797	3054.071
	20	516.103	512.610	510.546	1099.199	1759.448	2477.007	3021.368
PS mode	0	20.349	41.126	64.903	712.858	1180.742	1652.981	2068.886
	5	94.676	120.595	138.159	756.769	1236.005	1690.681	2107.585
	10	188.282	155.224	173.431	755.412	1274.347	1724.810	2113.297
	20	257.897	474.881	274.890	768.764	1247.358	1725.381	2117.081
OD mode	0	15.422	26.561	43.711	807.412	1516.140	2220.892	2926.660
	5	105.881	133.768	161.069	767.735	1733.671	2433.350	3063.280
	10	216.499	175.010	182.916	869.106	1578.218	2407.817	3072.742
	20	232.068	220.862	300.934	799.179	1557.660	2429.356	3029.503
LAD	0	20.563	41.198	95.615	776.907	1475.156	2291.333	2901.684
	5	112.530	106.221	126.801	957.703	1557.115	2203.301	2980.904
	10	207.967	161.345	166.637	870.518	1631.205	2207.031	2976.349
	20	213.865	302.963	294.978	676.686	1370.538	2073.595	2787.763

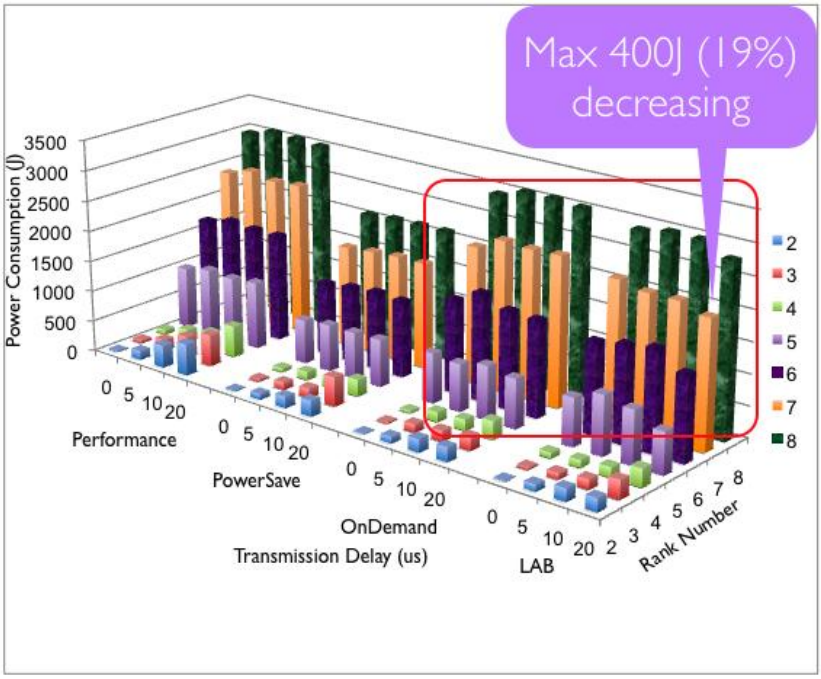


Figure 8: Power effect of TD on PC (Frame = one Byte)

1460 byte frame

Table 5 and Figure 7 show 1460 bytes frame TT. By comparing PM mode and OD mode, the completed time is longer than one byte frame in all situations. In Figure 8, OD mode uses in average over 200J less than PM mode. Our LAD algorithm made uses of 24~25s to complete data transmission as OD mode, yet consumes less than OD mode 200~600J in 8 ranks. In other situations, LAD keeps nearly the same performance, spending 3s longer than OD mode and consuming 200~600J less than OD mode.

Table 6: Detail results of time effect of TD on TT (Frame = 1460 Byte)

Mode & TD \ Rank Number		2	3	4	5	6	7	8
PM mode	0	0.353	0.525	0.721	8.969	12.421	17.518	25.286
	5	0.996	1.188	1.321	11.687	13.115	18.323	24.394
	10	2.481	2.267	1.818	9.811	12.892	17.752	23.960
	20	3.330	3.281	3.245	14.031	12.760	17.835	24.511
PS mode	0	0.621	0.913	1.254	11.391	18.925	25.933	31.738
	5	1.448	1.825	2.004	10.599	19.379	26.427	32.430
	10	2.947	2.252	2.442	12.100	19.803	26.545	32.802
	20	3.708	3.749	3.941	11.890	19.405	26.641	32.827
OD mode	0	0.408	0.548	0.738	7.769	13.033	18.427	24.356
	5	1.394	1.707	1.931	8.435	13.749	19.017	25.097
	10	2.818	2.221	2.285	8.329	13.512	18.971	24.542
	20	3.723	3.720	3.874	8.352	13.584	18.841	24.547
LAD	0	0.630	0.940	1.271	10.855	16.063	21.985	24.732
	5	1.403	1.500	1.646	10.192	16.104	21.200	24.741
	10	2.861	1.993	2.080	9.852	16.307	21.228	25.182
	20	3.742	3.871	3.611	10.482	17.143	21.356	25.566

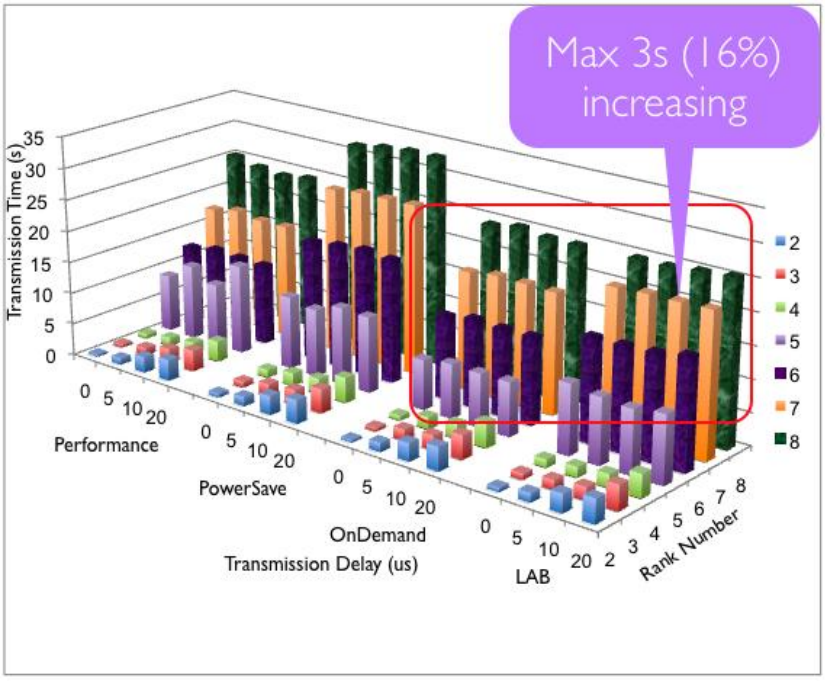


Figure 9: Time effect of TD on TT (Frame = 1460 Byte)

Table 7: Detail results of power effect of TD on PC (Frame = 1460 Byte)

Mode & TD \ Rank Number		2	3	4	5	6	7	8
PM mode	0	56.039	83.345	114.460	1423.847	1971.859	2781.018	4014.203
	5	158.117	188.597	209.711	1855.335	2082.032	2908.813	3872.596
	10	393.864	359.891	288.611	1557.516	2046.631	2818.166	3803.698
	20	528.644	520.865	515.150	2227.449	2025.676	2831.342	3891.170
PS mode	0	44.339	65.188	89.536	813.317	1351.245	1851.616	2266.093
	5	103.387	130.305	143.086	756.769	1383.661	1886.888	2315.502
	10	210.416	160.793	174.359	863.940	1413.934	1895.313	2342.063
	20	264.751	267.679	281.387	848.946	1385.517	1902.167	2343.848
OD mode	0	33.586	39.127	52.693	945.259	1645.667	2710.100	3839.306
	5	110.451	132.799	158.958	1032.840	1849.698	2663.291	3900.304
	10	223.043	182.830	195.905	1049.544	1827.601	2729.659	3861.452
	20	306.473	306.226	309.360	1022.164	1827.937	2739.376	3834.217
LAD	0	44.982	87.644	123.505	1028.737	1705.181	2431.432	3650.212
	5	104.575	118.019	124.578	1044.754	1715.120	2455.331	3608.556
	10	235.515	153.219	171.224	976.402	1847.987	2431.883	3525.145
	20	308.037	307.738	290.581	890.359	1654.873	2355.386	3209.088

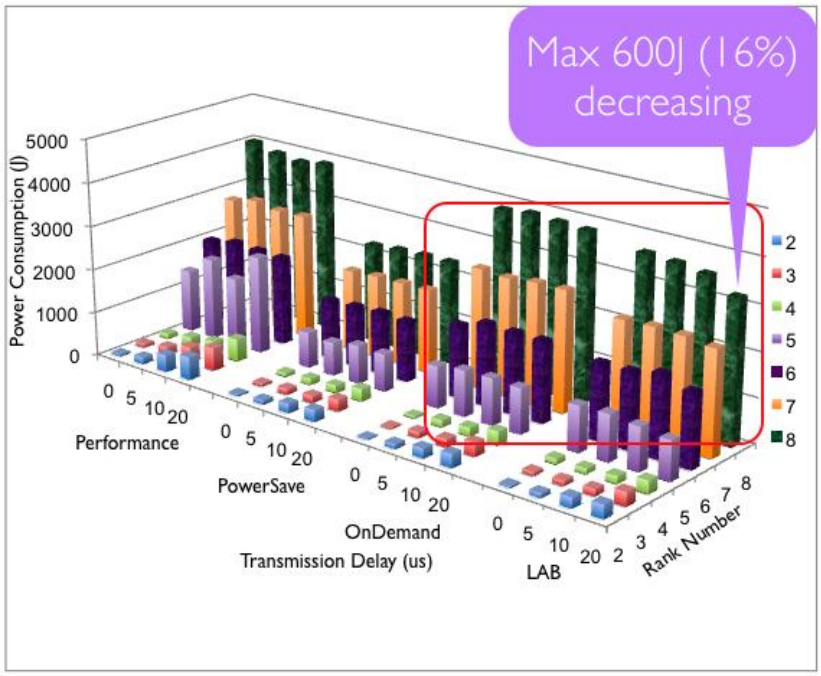


Figure 10: Power effect of TD on PC (Frame = 1460 Byte)

8000 byte frame

Although 8000 byte frame is the longest one, PS mode TT keeps 6s longer than other frames' size, as in Figure 9. Comparing OD and PM Mode, OD mode spends less than 1s longer than PM Mode, yet saves 200~400J in other cases. Comparing LAD algorithm and OD mode, LAD algorithm still keeps its advantages in the longest frame size, spends almost the same TT in 8 ranks and average 2~3s longer in other cross-node situations, consuming 100~ 400J less than OD mode.

Table 8: Detail results of time effect of TD on TT (Frame = 8000 Byte)

Mode & TD \ Rank Number		2	3	4	5	6	7	8
PM mode	0	1.220	1.409	1.597	11.158	20.343	26.952	31.241
	5	1.484	1.710	1.783	13.364	21.986	27.664	34.053
	10	1.993	2.171	2.260	11.857	21.455	27.397	33.398
	20	3.824	3.753	3.732	11.247	21.604	27.513	33.178

PS mode	0	2.333	2.619	2.812	16.480	27.429	34.139	38.755
	5	2.240	2.684	2.964	16.716	27.728	35.219	39.884
	10	2.774	3.088	3.245	17.336	19.803	35.387	41.127
	20	4.685	4.678	4.244	16.700	27.613	35.219	39.930
OD mode	0	1.274	1.464	1.610	10.648	22.022	27.752	31.210
	5	2.045	2.407	2.226	14.377	21.778	27.739	34.744
	10	2.546	2.810	2.769	13.856	22.079	27.932	34.379
	20	4.338	4.380	4.448	14.037	21.957	27.603	34.878
LAD	0	1.917	2.125	2.200	12.242	23.169	28.553	33.991
	5	2.234	2.399	2.579	13.487	22.152	29.627	34.864
	10	2.672	2.779	2.740	13.018	24.838	30.845	34.518
	20	4.456	4.663	4.163	12.754	22.897	29.118	36.666

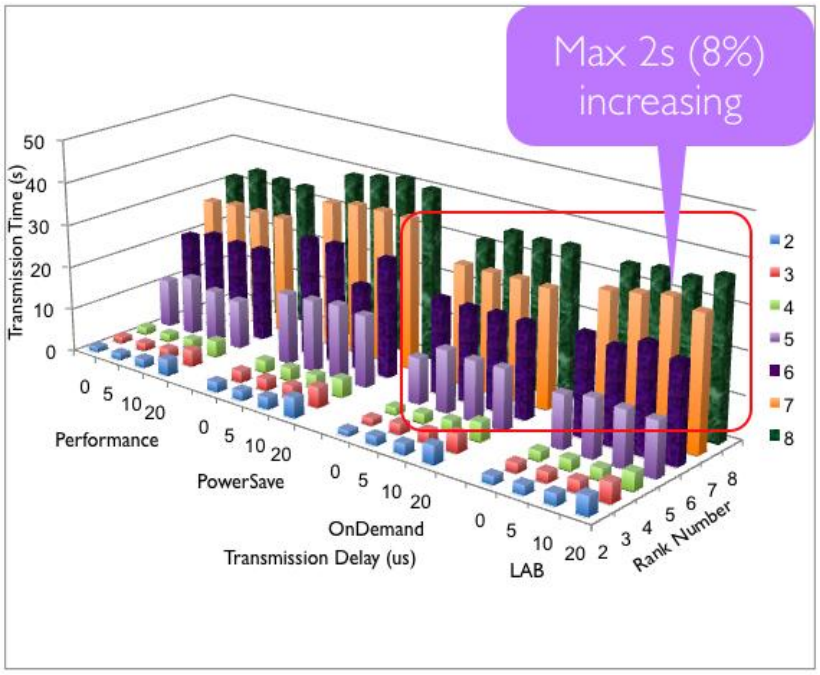


Figure 11: Time effect of TD on TT (Frame = 8000 Byte)

Table 9: Detail results of power effect of TD on PC (Frame = 8000 Byte)

Mode & TD \ Rank Number		2	3	4	5	6	7	8
PM mode	0	193.677	223.682	253.527	1771.355	3229.492	4278.684	4959.571
	5	235.588	271.466	283.055	2121.562	3490.321	4391.715	5405.982
	10	316.393	344.651	358.780	1882.322	3406.024	4349.329	5301.999
	20	607.068	595.796	592.462	1785.484	3429.678	4367.744	5267.074
PS mode	0	166.576	186.997	200.777	1176.672	1958.431	2437.525	2767.107
	5	159.936	191.638	211.630	1193.522	1979.779	2514.637	2847.718
	10	198.064	220.483	231.693	1237.790	1413.934	2526.632	2936.468
	20	334.509	334.009	303.022	1192.380	1971.568	2514.637	2851.002
OD mode	0	107.866	147.418	185.271	1320.356	2877.695	4069.769	4903.488
	5	156.932	193.698	202.611	1652.663	2925.864	4059.867	5447.829
	10	203.622	231.316	238.859	1706.812	2862.416	4076.114	5432.837
	20	353.408	367.326	361.262	1664.418	3014.893	4030.163	5487.617
LAD	0	167.818	201.826	224.777	1355.342	2522.337	3977.022	4695.212
	5	183.581	197.163	205.979	1436.942	2499.579	3962.008	4708.029
	10	212.619	220.259	225.554	1301.254	2938.202	4316.622	5198.511
	20	284.493	376.613	329.994	1265.924	2629.309	4060.896	5061.468

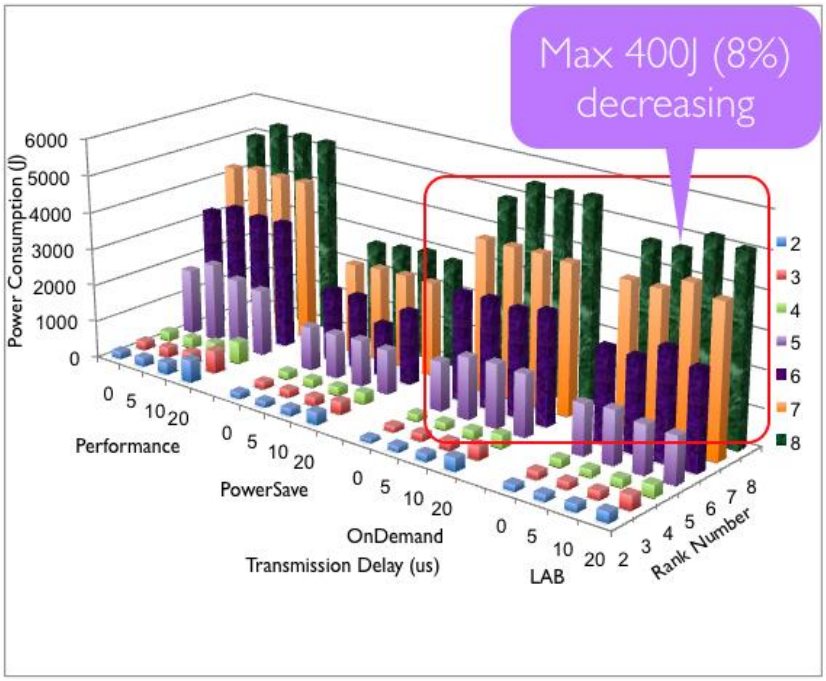


Figure 12: Power effect of TD on PC (Frame = 8000 Byte)

Remarks

In this proposed research, LAD algorithm keeps in average 4s TT increasing, yet saves 200~600J that compares with OD mode in cross-node situation. Limited by only 2 steps experimental cases of CPU frequencies (2.3GHz and 1.15GHz), we cannot keep CPU loading in a smooth curve. In desktop and server CPU, they do not keep in high loading work longer time, while they complete a concurrent job and next one does not be started. Power saving technology helps to decrease host energy consumption, and decreasing energy cost and carbon dioxide emissions can be reduced.

6. Conclusions

One byte data frame is the smallest one, and it has 5 seconds transmission time shorter than 1460 bytes frame and 14 seconds shorter than 8000 bytes frame in cross node situation. That means two kinds of application which have no huge data need to be transmitted are suitable to use small data frame.

- Mathematical calculation

● Operation command sending in any application

Small data frame helps to reduce transmission time and energy consumption, more core calculation cycles can be released to do CPU bound jobs.

Besides, there are two kinds of application suitable to use large data frame.

● Database server that is sending data back

● Distributed file transmission

Larger data frame reduces frame generated time and transmits more data in single frame because larger content space.

There are many directions to continue this investigation, to develop methods to save energy. If hardware and software provides functions about voltage or speed control, motherboard or any other type of peripheral device, then a hardware driver, power-aware job scheduling and data distribution algorithms can be combined and implemented, targeting in the construction of a low energy cost cluster computing platform in future.

Reference

1. “Power Management Guide”, http://www.gentoo.com/doc/en/power-management-guide.xml
2. “Enabling CPU Frequency Scaling”, http://ubuntu.wordpress.com/2005/11/04/enabling-cpu-freq uency -scaling/
3. “Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor”, ftp://download.intel.co m/design/network/papers/30117401.pdf
4. “AMD PowerNow! Technology Platform Design Guide for Embedded Processors”, http://www.am d.com/epd/processors/6.32bitproc/8.amdk6fami/x24267/24267a.pdf
5. “AMD / Intel CPU voltage control driver down load”, http://www.linux-phc.org/viewtopic.php? f= 13 &t= 2

6. “AMD Family 10h Desktop Processor Power and Thermal Data Sheet”, http://www.amd.com/us-en /assets/content_type/white_papers_and_tech_docs/GH_43375_10h_DT_PTDS_PUB_3.14.pdf

7. “AMD Opteron Processor with Direct Connect Architecture”, http://enterprise.amd.com/downloads/4 P_Power_PID_4149 8.pdf

8. Chao-Yang Lan, Ching-Hsien Hsu and Shih-Chang Chen, “Scheduling Contention-Free Irregular R edistributions in Parallelizing Compilers”, The Journal of Supercomputing, Volume 40, Issue 3, (J une 2007), Pages: 229-247

9. Dongkun Shin and Jihong Kim, “Power-Aware Communication Optimization for Networks-on-Chi ps with Voltage Scalable Links”, Proceeding of the International Conference on Hardware/Softwar e Codesign and System Synthesis, 2004, Pages: 170-175

10. Guangyu Chen, Feihui Li and Mahmut Kandemir, “Reducing Energy Consumption of On-Chip Networks Through a Hybrid Compiler-Runtime Approach”, 16th International Conference on Par allel Architecture and Compilation Techniques (PA CT 2007), Pages: 163-174

11. “Intel 64 And IA-32 Architectures Software Developers Manual, Volume 1”, http://download.in t el.com/design/processor/manuals/253665.pdf

12. “Key Architectural Features of AMD Phenom X4 Quad-Core Processors”, http://www.amd.com/u s-en/Processors/ProductInformation/0,,30_118_15331_15332%5E15334,00.html

13. Lei Chia, Albert Hartono, Dhabaleswar K. Panda, “Designing High Performance and Scalable M PI Inter-node Communication Support for Clusters”, 2006 IEEE International Conference on Clu ster Computing, 25-28 Sept. 2006, Pages: 1-10

14. Ranjit Noronha and D.K. Panda, “Improving Scalability of OpenMP Applications on Multi-core Systems Using Large Page Support”, 2007 IEEE International Parallel and Distributed Processin g Symposium, 26-30 March 2007, Pages: 1-8

15. Umit Y. Ogras, Radu Marculescu, Hyung Gyu Lee and Na Ehyuck Chang, “Communication Ar chitecture Optimization: Making the Shortest Path Shorter in Regular Networks-on-Chip”, 2006 Proceedings of the conference on Design, Automation and Test in Europe, Munich, Germany, March 2006, Volume 1, Pages: 712-717

16. “InfiniBand Introduction”, http://en.wikipedia.org/wiki/InfiniBand

17. Seung Woo Son, Guangyu Chen, Ozcan Ozturk Mahmut Kandemir and Alok Choudhary, “Com piler-Directed Energy Optimization for Parallel-Disk-Based Systems” IEEE Transactions on Parall el and Distributed Systems, September 2007, Volume. 18, No. 9, Pages: 1241-1257

18. Sri Hari Krishna Narayanan, Mahmut Kandemir and Ozcan Ozturk, “Compiler-Directed Power D ensity Reduction in NoC-Based Multi-Core Designs”, Proceedings of the 7th International Synpo sium on Quality Electronic Desing, 2006, Pages: 570-575

19. David Meisner, Brian T. Gold and Thomas F. Wenisch, “PowerNap: eliminating server idle po wer”, Proceeding of the 14th International Conference on Architectural Support for Programming Languages and Operation Systems, 2009, Pages: 205-216

20. Michael B. Healy, Hsien-Hsin S. Lee, Gabriel H. Loh and Sung Kyu Lim, “Thermal Optimizati on in Multi-Granularity Multi-core Floorplanning” Proceedings of the 2009 Conference on Asia and South Pacific Design Automation, 2009, Pages: 43-48

21. M. Aater Suleman, Onur Mutlu, moinuddin K. Qureshi and Yale N. Patt, “Accelerating Critical Section Execution with Asymmetric Multi-core Architecture”, Proceeding of the 14th Inteernati onal Conferenceon Architectural Support for Programming Languages and Operating Systems, 20 09, Pages: 253-264

22. David C. Snowdon, Etienne Le Sueur, Stefan M. Petters and Gemot Heiser, “Koala: A Platform for OS-Level Power Management”, Proceedings of the Fourth ACM European Conference on Computer Systems, 2009, Pages: 289-302

23. Alexander S. van Amesfoort, Ana lucia Varbanescu, Henk J. Sips and Rob V. van Nieuwpoort, “Evaluating Multi-core Platforms for HPC Data-Intensive kernels”, Proceedings of the 6th AC M Conference on Computing Frontiers, 2009, Pages: 207-216

24. XianGrong Zhou, ChenJie Yu and Peter Petrov, “Temperature-Aware Register Reallocation for Register File Power-Density Minimization”, ACM Transactions on Design Automation of Electro nic Systems, March 2009, Volume 14, Issue 2, No. 26.

25. Radha Guha, Nader Bagherzadeh and Pai Chou, “Resource Management and Task Partitioning a nd Scheduling on a Run-Time Reconfigurable Embedded System”, Computers and Electrical En gineering, March 2009, Volume 35, Issue 2, Pages: 258-285

行政院所屬各機關人員出國報告書提要

撰寫時間： 98 年 5 月 30 日

姓名	許慶賢	服務機關名稱	中華大學資工系	連絡電話、電子信箱	03-5186410 chh@chu.edu.tw
出生日期	62 年 2 月 23 日		職稱	副教授	
出席國際會議名稱	The 3rd ChinaGrid Annual Conference (ChinaGrid)				
到達國家及地點	Kunming, China		出國期間	自 98 年 05 月 24 日 迄 98 年 05 月 29 日	
內容提要	這一次在昆明所舉行的國際學術研討會議共計四天。第一天上午本人抵達會場辦理報到，第一天，除了主持一場 invited session 的論文發表。同時，自己也在上午的場次發表了這次被大會接受的論文。第二天，聽取了 Prof. Kai Hwang 有關於 Massively Distributed Systems: From Grids and P2P to Clouds 精闢的演說。第二天許多重要的研究進行論文發表。本人餐與 Architecture and Infrastructure、Grid computing、以及 P2P computing 相關場次聽取報告。晚上本人亦參加酒會，並且與幾位國外學者及中國、香港教授交換意見，合影留念。第三天本人在上午聽取了 Data and Information Management 相關研究，同時獲悉許多新興起的研究主題，並了解目前國外大多數學者主要的研究方向，並且把握機會與國外的教授認識，希望能夠讓他們加深對台灣研究的印象。第四天，本人則是選擇 Service Oriented Computing 以及 Network Storage 相關研究聆聽論文發表。四天下來，本人聽了許多優秀的論文發表。這些研究所涵蓋的主題包含有：網格系統技術、工作排程、網格計算、網格資料庫以及無線網路等等熱門的研究課題。此次的國際學術研討會議有許多知名學者的參與，讓每一位參加這個會議的人士都能夠得到國際上最新的技術與資訊。是一次非常成功的學術研討會。參加本次的國際學術研討會議，感受良多。讓本人見識到許多國際知名的研究學者以及專業人才，得以與之交流。讓本人與其他教授面對面暢談所學領域的種種問題。看了眾多研究成果以及聽了數篇專題演講，最後，本人認為，會議所安排的會場以及邀請的講席等，都相當的不錯，會議舉辦得很成功，值得我們學習。				
出席人所屬機關審核意見					
層轉機關審核意見					
研考會處理意見					

Towards Improving QoS-Guided Scheduling in Grids

Ching-Hsien Hsu[1], Justin Zhan[2], Wai-Chi Fang[3] and Jianhua Ma[4]

[1]*Department of Computer Science and Information Engineering, Chung Hua University, Taiwan*
chh@chu.edu.tw

[2]*Heinz School, Carnegie Mellon University, USA*
justinzh@andrew.cmu.edu

[3]*Department of Electronics Engineering, National Chiao Tung University, Taiwan*
wfang@mail.nctu.edu.tw

[4]*Digital Media Department, Hosei University, Japan*
jianhua@hosei.ac.jp

Abstract

With the emergence of grid technologies, the problem of scheduling tasks in heterogeneous systems has been arousing attention. In this paper, we present two optimization schemes, Makespan Optimization Rescheduling (MOR) and Resource Optimization Rescheduling (ROR), which are based on the QoS Min-Min scheduling technique, for reducing the makespan of a schedule and the need of total resource amount. The main idea of the proposed techniques is to reduce overall execution time without increasing resource need; or reduce resource need without increasing overall execution time. To evaluate the effectiveness of the proposed techniques, we have implemented both techniques along with the QoS Min-Min scheduling algorithm. The experimental results show that the MOR and ROR optimization schemes provide noticeable improvements.

1. Introduction

With the emergence of IT technologies, the need of computing and storage are rapidly increased. To invest more and more equipments is not an economic method for an organization to satisfy the even growing computational and storage need. As a result, grid has become a widely accepted paradigm for high performance computing.

To realize the concept virtual organization, in [13], the grid is also defined as "A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements". As the grid system aims to satisfy users' requirements with limit resources, scheduling grid resources plays an important factor to improve the overall performance of a grid.

In general, grid scheduling can be classified in two categories: the performance guided schedulers and the economy guided schedulers [16]. Objective of the performance guided scheduling is to minimize turnaround time (or makespan) of grid applications. On the other hand, in economy guided scheduling, to minimize the cost of resource is the main objective. However, both of the scheduling problems are NP-complete, which has also instigated many heuristic solutions [1, 6, 10, 14] to resolve. As mentioned in [23], a complete grid scheduling framework comprises application model, resource model, performance model, and scheduling policy. The scheduling policy can further decomposed into three phases, the resource discovery and selection phase, the job scheduling phase and the job monitoring and migration phase, where the second phase is the focus of this study.

Although many research works have been devoted in scheduling grid applications on heterogeneous system, to deal with QOS scheduling in grid is quite complicated due to more constrain factors in job scheduling, such as the need of large storage, big size memory, specific I/O devices or real-time services, requested by the tasks to be completed. In this paper, we present two QoS based rescheduling schemes aim to improve the makespan of scheduling batch jobs in grid. In addition, based on the QoS guided scheduling scheme, the proposed rescheduling technique can also reduce the amount of resource need without increasing the makespan of grid jobs. The main contribution of this work are twofold, one can shorten the turnaround time of grid applications without increasing the need of grid resources; the other one can minimize the need of grid resources without increasing the turnaround time of grid applications,

compared with the traditional QoS guided scheduling method. To evaluate the performance of the proposed techniques, we have implemented our rescheduling approaches along with the QoS Min-Min scheduling algorithm [9] and the non-QoS based Min-Min scheduling algorithm. The experimental results show that the proposed techniques are effective in heterogeneous systems under different circumstances. The improvement is also significant in economic grid model [3].

The rest of this paper is organized as follows. Section 2 briefly describes related research in grid computing and job scheduling. Section 3 clarifies our research model by illustrating the traditional Min-min model and the QoS guided Min-min model. In Section 4, two optimization schemes for reducing the total execution time of an application and reducing resource need are presented, where two rescheduling approaches are illustrated in detail. We conduct performance evaluation and discuss experiment results in Section 5. Finally, concluding remarks and future work are given in Section 6.

2. Related Work

Grid scheduling can be classified into traditional grid scheduling and QoS guided scheduling or economic based grid scheduling. The former emphasizes the performance of systems of applications, such as system throughput, jobs' completion time or response time. Swany *et al*. provides an approach to improving throughput for grid applications with network logistics by building a tree of "best" paths through the graph and has running time of O(Nlog$N$) for implementations that keep the edges sorted [15]. Such approach is referred as the Minimax Path (MMP) and employs a greedy, tree-building algorithm that produces optimal results [20]. Besides data-parallel applications requiring high performance in grid systems, there is a Dynamic Service Architecture (DSA) based on static compositions and optimizations, but also allows for high performance and flexibility, by use of a lookahead scheduling mechanism [4]. To minimizing the processing time of extensive processing loads originating from various sources, the approaches divisible load model [5] and single level tree network with two root processors with divisible load are proposed [12]. In addition to the job matching algorithm, the resource selection algorithm is at the core of the job scheduling decision module and must have the ability to integrate multi-site computation power. The CGRS algorithm based on the distributed computing grid model and the grid scheduling model integrates a new density-based internet clustering algorithm into the decoupled scheduling approach of the GrADS and decreases its time complexity [24]. The scheduling of parallel jobs in a heterogeneous multi-site environment, where each site has a homogeneous cluster of processors, but processors at different sites has different speeds, is presented in [18]. Scheduling strategy is not only in batch but also can be in real-time. The SAREG approach paves the way to the design of security-aware real-time scheduling algorithms for Grid computing environments [21].

For QoS guided grid scheduling, apparently, applications in grids need various resources to run its completion. In [17], an architecture named public computing utility (PCU) is proposed uses virtual machine (VMs) to implement "time-sharing" over the resources and augments finite number of private resources to public resources to obtain higher level of quality of services. However, the QoS demands maybe include various packet-type and class in executing job. As a result, a scheduling algorithm that can support multiple QoS classes is needed. Based on this demand, a multi-QoS scheduling algorithm is proposed to improve the scheduling fairness and users' demand [11]. He *et al.* [7] also presented a hybrid approach for scheduling moldable jobs with QoS demands. In [9], a novel framework for policy based scheduling in resource allocation of grid computing is also presented. The scheduling strategy can control the request assignment to grid resources by adjusting usage accounts or request priorities. Resource management is achieved by assigning usage quotas to intended users. The scheduling method also supports reservation based grid resource allocation and quality of service feature. Sometimes the scheduler is not only to match the job to which resource, but also needs to find the optimized transfer path based on the cost in network. In [19], a distributed QoS network scheduler (DQNS) is presented to adapt to the ever-changing network conditions and aims to serve the path requests based on a cost function.

3. Research Architecture

Our research model considers the static scheduling of batch jobs in grids. As this work is an extension and optimization of the QoS guided scheduling that is based on Min-Min scheduling algorithm [9], we briefly describe the Min-Min scheduling model and the QoS guided Min-Min algorithm. To simplify the presentation, we first clarify the following terminologies and assumptions.

- *QoS Machine* (M_Q) – machines can provide special services.
- *QoS Task* (T_Q) – tasks can be run completion only on QoS machine.
- *Normal Machine* (M_N) – machines can only run normal tasks.
- *Normal Task* (T_N) – tasks can be run completion on both QoS machine and normal machine.

- A chunk of tasks will be scheduled to run completion based on all available machines in a batch system.
- A task will be executed from the beginning to completion without interrupt.
- The completion time of task t_i to be executed on machine m_j is defined as

$$CT_{ij} = dt_{ij} + et_{ij} \quad (1)$$

Where et_{ij} denotes the estimated execution time of task t_i executed on machine m_j; dt_{ij} is the delay time of task t_i on machine m_j.

The Min-Min algorithm is shown in Figure 1.

```
Algorithm_Min-Min()
{
  while there are jobs to schedule
    for all job i to schedule
      for all machine j
        Compute CT_{i,j} = CT(job i, machine j)
      end for
      Compute minimum CT_{i,j}
    end for
    Select best metric match m
    Compute minimum CT_{m,n}
    Schedule job m on machine n
  end while
} End_of_ Min-Min
```

Figure 1. The Min-Min Algorithm

Analysis: If there are m jobs to be scheduled in n machines, the time complexity of Min-Min algorithm is $O(m^2n)$. The Min-Min algorithm does not take into account the QoS issue in the scheduling. In some situation, it is possible that normal tasks occupied machine that has special services (referred as QoS machine). This may increase the delay of QoS tasks or result idle of normal machines.

The QoS guided scheduling is proposed to resolve the above defect in the Min-Min algorithm. In QoS guided model, the scheduling is divided into two classes, the QoS class and the non-QoS class. In each class, the Min-Min algorithm is employed. As the QoS tasks have higher priority than normal tasks in QoS guided scheduling, the QoS tasks are prior to be allocated on QoS machines. The normal tasks are then scheduled to all machines in Min-Min manner. Figure 2 outlines the method of QoS guided scheduling model with the Min-Min scheme.

Analysis: If there are m jobs to be scheduled in n machines, the time complexity of QoS guided scheduling algorithm is $O(m^2n)$.

Figure 3 shows an example demonstrating the Min-Min and QoS Min-Min scheduling schemes. The asterisk * means that tasks/machines with QoS demand/ability, and the X means that QoS tasks couldn't be executed on that machine. Obviously, the QoS guided scheduling algorithm gets the better performance than the Min-Min algorithm in term of makespan. Nevertheless, the QoS guided model is not optimal in both makespan and resource cost. We will describe the rescheduling optimization in next section.

```
Algorithm_QOS-Min-Min()
{
  for all tasks ti in meta-task Mv (in an arbitrary order)
    for all hosts m_j (in a fixed arbitrary order)
      CT_ij = et_ij + dt_j
    end for
  end for
  do until all tasks with QoS request in Mv are mapped
    for each task with high QoS in Mv,
      find a host in the QoS qualified host set that obtains
      the earliest completion time
    end for
    find task t_k with the minimum earliest completion time
    assign task t_k to host m_l that gives the earliest completion
    time
    delete task t_k from Mv
    update d_tl
    update CT_il for all i
  end do
  do until all tasks with non-QoS request in Mv are mapped
    for each task in Mv
      find the earliest completion time and the
      corresponding host
    end for
    find the task t_k with the minimum earliest completion time
    assign task t_k to host m_l that gives the earliest completion
    time
    delete task t_k from Mv
    update d_tl
    update CT_il for all i
  end do
} End_of_ QOS-Min-Min
```

Figure 2. The QoS Guided Algorithm

4. Rescheduling Optimization

Grid scheduling works as the mapping of individual tasks to computer resources, with respecting service level agreements (SLAs) [2]. In order to achieve the optimized performance, how to mapping heterogeneous tasks to the best fit resource is an important factor. The Min-Min algorithm and the QoS guided method aims at scheduling jobs to achieve better makespan. However, there are still having rooms to make improvements. In this section, we present two optimization schemes based on the QoS guided Min-Min approach.

4.1 Makespan Optimization Rescheduling (*MOR*)

The first one is *Makespan Optimization Rescheduling* (*MOR*), which focuses on improving the makespan to achieve better performance than the QoS guided scheduling algorithm. Assume the makespan achieved by the QoS guided approach in different machines are CT_1, CT_2, …, CT_m, with CT_k = max { CT_1, CT_2, …, CT_m }, where m is the number of machines and $1 \le k \le m$. By subtracting $CT_k - CT_i$, where $1 \le i \le m$ and $i \ne k$, we can have m-1 available time fragments. According to the size of these available time fragments and the size of tasks in machine M_k, the *MOR* dispatches suitable tasks from machine M_k to any other machine that has available and large enough time fragments. Such optimization is repeated until there is no task can be moved.

	*M1	M2	M3
T1	7	4	7
T2	3	3	5
T3	9	5	7
*T4	5	X	X
T5	9	8	6
*T6	5	X	X

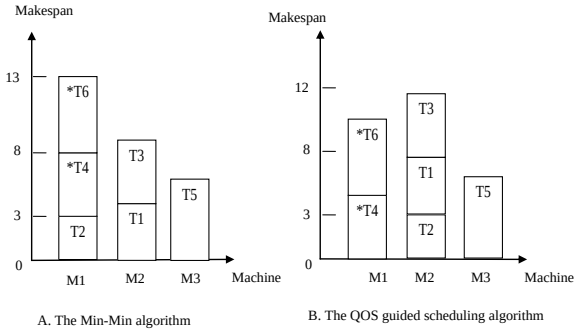
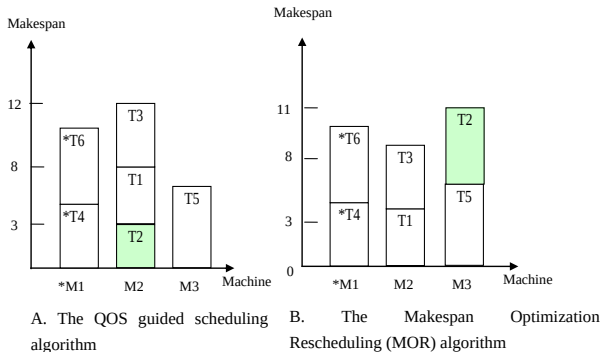


Figure 3. Min-Min and QoS Guided Min-Min

	*M1	M2	M3
T1	7	4	7
T2	3	3	5
T3	9	5	7
*T4	5	X	X
T5	9	8	6
*T6	5	X	X



12

Figure 4. Example of *MOR*

Recall the example given in Figure 3, Figure 4 shows the optimization of the *MOR* approach. The left side of Figure 4 demonstrates that the QoS guided scheme gives a schedule with makespan = 12, wheremachine M2 presents maximum *CT* (completion time), which is assembled by tasks T2, T1 and T3. Since the *CT* of machine 'M3' is 6, so 'M3' has an available time fragment (6). Checking all tasks in machine M2, only T2 is small enough to be allocated in the available time fragment in M3. Therefore, task M2 is moved to M3, resulting machine 'M3' has completion time *CT*=11, which is better than the QoS guided scheme.

As mentioned above, the *MOR* is based on the QoS guided scheduling algorithm. If there are m tasks to be scheduled in n machines, the time complexity of *MOR* is $O(m^2n)$. Figure 5 outlines a pseudo of the *MOR* scheme.

```
Algorithm_MOR()
{
  for CT_j in all machines
      find out the machine with maximum makespan CT_max and
      set it to be the standard
  end for
  do until no job can be rescheduled
      for job i in the found machine with CT_max
          for all machine j
              according to the job's QOS demand, find the
              adaptive machine j
              if (the execute time of job i in machine j + the
              CT_j < makespan)
                  rescheduling the job i to machine j
                  update the CT_j and CT_max
                  exit for
              end if
          next for
          if the job i can be reschedule
              find out the new machine with maximum CT_max
              exit for
          end if
      next for
  end do
} End_of_ MOR
```

Figure 5. The *MOR* Algorithm

4.2 Resource Optimization Rescheduling (*ROR*)

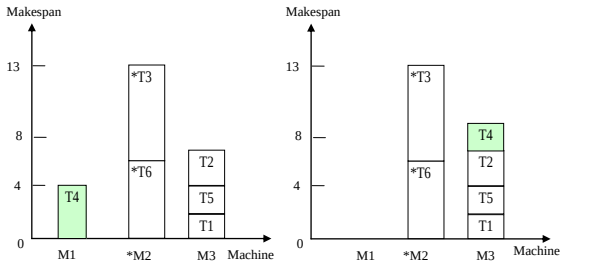
Following the assumptions described in *MOR*, the main idea of the *ROR* scheme is to re-dispatch tasks from the machine with minimum number of tasks to other machines, expecting a decrease of resource need. Consequently, if we can dispatch all tasks from machine M_x to other machines, the total amount of resource need will be decreased.

Figure 6 gives another example of QoS scheduling, where the QoS guided scheduling presents makespan = 13. According to the clarification of *ROR*, machine 'M1' has the fewest amount of tasks. We can dispatch the task 'T4' to machine 'M3' with the following constraint

$$CT_{ij} + CT_j <= CT_{max} \qquad (2)$$

The above constraint means that the rescheduling can be performed only if the movement of tasks does not increase the overall makespan. In this example, $CT_{43} = 2$, $CT_3=7$ and $CT_{max}=CT_2=13$. Because the makespan of M3 (CT_3) will be increased from 7 to 9, which is smaller than the CT_{max}, therefore, the task migration can be performed. As the only task in M1 is moved to M3, the amount of resource need is also decreased comparing with the QoS guided scheduling.

	M1	*M2	M3
T1	3	4	2
T2	6	6	3
*T3	X	7	X
T4	4	6	2
T5	5	7	2
*T6	X	6	X



A. The QOS guided scheduling

B. The Resource Optimization Rescheduling (ROR) Algorithm

Figure 6. Example of *ROR*

The *ROR* is an optimization scheme which aims to minimize resource cost. If there are m tasks to be scheduled in n machines, the time complexity of *ROR* is also $O(m^2n)$. Figure 7 depicts a high level description of the *ROR* optimization scheme.

```
Algorithm_MOR()
{
  for m in all machines
      find out the machine m with minimum count of jobs
  end for
  do until no job can be rescheduled
      for job i in the found machine with minimum count of jobs
          for all machine j
              according to the job's QOS demand, find the
              adaptive machine j
              if (the execute time of job i in machine j + the
              CT_j <= makespan CT_max)
                  rescheduling the job i to machine j
                  update the CT_j
                  update the count of jobs in machine m and
                  machine j
                  exit for
              end if
          next for
      next for
  end do
} End_of_ MOR
```

Figure 7. The *ROR* Algorithm

5. Performance Evaluation

5.1 Parameters and Metrics

To evaluate the performance of the proposed techniques, we have implemented the Min-Min scheduling algorithm and the QoS guided Min-Min

scheme. The experiment model consists of heterogeneous machines and tasks. Both of the Machines and tasks are classified into QoS type and non-QoS type. Table 1 summarizes six parameters and two comparison metrics used in the experiments. The number of tasks is ranged from 200 to 600. The number of machines is ranged from 50 to 130. The percentage of QoS machines and tasks are set between 15% and 75%. Heterogeneity of tasks are defined as H_t (for non-QoS task) and H_Q (for QoS task), which is used in generating random tasks. For example, the execution time of a non-QoS task is randomly generated from the interval [10, $H_t \times 10^2$] and execution time of a QoS task is randomly generated from the interval [10^2, $H_Q \times 10^3$] to reflect the real application world. All of the parameters used in the experiments are generated randomly with a uniform distribution. The results demonstrated in this section are the average values of running 100 random test samples.

Table 1: Parameters and Comparison Metrics

Task number (N_T)	{200, 300, 400, 500, 600}
Resource number (N_R)	{50, 70, 90, 110, 130}
Percentage of QOS resources (Q_R%)	{15%, 30%, 45%, 60%, 75%}
Percentage of QOS tasks (Q_T%)	{15%, 30%, 45%, 60%, 75%}
Heterogeneity of non-QOS tasks (H_T)	{1, 3, 5, 7, 9}
Heterogeneity of QOS tasks (H_Q)	{3, 5, 7, 9, 11}
Makespan	The completion time of a set of tasks
Resource Used (R_U)	Number of machines used for executing a set of tasks

5.2 Experimental Results of *MOR*

Table 2 compares the performance of the *MOR*, Min-Min algorithm and the QoS guided Min-Min scheme in term of makespan. There are six tests that are conducted with different parameters. In each test, the configurations are outlined beside the table caption from (a) to (f). Table (a) changes the number of tasks to analyze the performance results. Increasing the number of tasks, improvement of *MOR* is limited. An average improvement ratio is from 6% to 14%. Table (b) changes the number of machines. It is obvious that the *MOR* has significant improvement in larger grid systems, i.e., large amount of machines. The average improvement rate is 7% to 15%. Table (c) discusses the influence of changing percentages of QoS machines. Intuitionally, the *MOR* performs best with 45% QoS machines. However, this observation is not always true. By analyzing the four best ones in (a) to (d), we observe that the four tests (a) N_T=200 (N_R=50, Q_R=30%, Q_T=20%) (b) N_R=130 (N_T=500, Q_R=30%, Q_T=20%) (c) Q_R=45% (N_T=300, N_R=50, Q_T=20%) and (d) Q_T=15% (N_T=300, N_R=50, Q_R=40%) have best improvements. All of the four configurations conform to the following relation,

$$0.4 \times (N_T \times Q_T) = N_R \times Q_R \quad (3)$$

This observation indicates that the improvement of *MOR* is significant when the number of QoS tasks is 2.5 times to the number of QoS machines. Tables (e) and (f) change heterogeneity of tasks. We observed that heterogeneity of tasks is not critical to the improvement rate of the *MOR* technique, which achieves 7% improvements under different heterogeneity of tasks.

Table 2: Comparison of Makespan

(a) (N_R=50, Q_R=30%, Q_T=20%, H_T=1, H_Q=1)

Task Number (N_T)	200	300	400	500	600
Min-Min	978.2	1299.7	1631.8	1954.6	2287.8
QOS Guided Min-Min	694.6	917.8	1119.4	1359.9	1560.1
MOR	597.3	815.5	1017.7	1254.8	1458.3
Improved Ratio	14.01%	11.15%	9.08%	7.73%	6.53%

(b) (N_T=500, Q_R=30%, Q_T=20%, H_T=1, H_Q=1)

Resource Number (N_R)	50	70	90	110	130
Min-Min	1931.5	1432.2	1102.1	985.3	874.2
QOS Guided Min-Min	1355.7	938.6	724.4	590.6	508.7
MOR	1252.6	840.8	633.7	506.2	429.4
Improved Ratio	7.60%	10.42%	12.52%	14.30%	15.58%

(c) (N_T=300, N_R=50, Q_T=20%, H_T=1, H_Q=1)

Q_R%	15%	30%	45%	60%	75%
Min-Min	2470.8	1319.4	888.2	777.6	650.1
QOS Guided Min-Min	1875.9	913.6	596.1	463.8	376.4
MOR	1767.3	810.4	503.5	394.3	339.0
Improved Ratio	5.79%	11.30%	15.54%	14.99%	9.94%

(d) (N_T=300, N_R=50, Q_R=40%, H_T=1, H_Q=1)

Q_T%	15%	30%	45%	60%	75%
Min-Min	879.9	1380.2	1801.8	2217.0	2610.1
QOS Guided Min-Min	558.4	915.9	1245.2	1580.3	1900.6
MOR	474.2	817.1	1145.1	1478.5	1800.1
Improved Ratio	15.07%	10.79%	8.04%	6.44%	5.29%

(e) (N_T=500, N_R=50, Q_R=30%, Q_T=20%, H_Q=1)

H_T	1	3	5	7	9
Min-Min	1891.9	1945.1	1944.6	1926.1	1940.1
QOS Guided Min-Min	1356.0	1346.4	1346.4	1354.9	1357.3
MOR	1251.7	1241.4	1244.3	1252.0	1254.2
Improved Ratio	7.69%	7.80%	7.58%	7.59%	7.59%

(f) (N_T=500, N_R=50, Q_R=30%, Q_T=20%, H_T=1)

H_Q	3	5	7	9	11
Min-Min	1392.4	1553.9	1724.9	1871.7	2037.8
QOS Guided Min-Min	867.5	1007.8	1148.2	1273.2	1423.1
MOR	822.4	936.2	1056.7	1174.3	1316.7
Improved Ratio	5.20%	7.11%	7.97%	7.77%	7.48%

5.3 Experimental Results of *ROR*

Table 3 analyzes the effectiveness of the *ROR* technique under different circumstances.

Table 3: Comparison of Resource Used

(a) (N_R=100, Q_R=30%, Q_T=20%, H_T=1, H_Q=1)

Task Number (N_T)	200	300	400	500	600
QOS Guided Min-Min	100	100	100	100	100
ROR	39.81	44.18	46.97	49.59	51.17
Improved Ratio	60.19%	55.82%	53.03%	50.41%	48.83%

(b) (N_T=500, Q_R=30%, Q_T=20%, H_T=1, H_Q=1)

Resource Number (N_R)	50	70	90	110	130
QOS Guided Min-Min	50	70	90	110	130
ROR	26.04	35.21	43.65	50.79	58.15
Improved Ratio	47.92%	49.70%	51.50%	53.83%	55.27%

(c) (N_T=500, N_R=50, Q_T=20%, H_T=1, H_Q=1)

Q_R%	15%	30%	45%	60%	75%
QOS Guided Min-Min	50	50	50	50	50
ROR	14.61	25.94	35.12	40.18	46.5
Improved Ratio	70.78%	48.12%	29.76%	19.64%	7.00%

(d) (N_T=500, N_R=100, Q_R=40%, H_T=1, H_Q=1)

Q_T%	15%	30%	45%	60%	75%
QOS Guided Min-Min	100	100	100	100	100
ROR	57.74	52.9	48.54	44.71	41.49
Improved Ratio	42.26%	47.10%	51.46%	55.29%	58.51%

(e) (N_T=500, N_R=100, Q_R=30%, Q_T=20%, H_Q=1)

H_T	1	3	5	7	9
QOS Guided Min-Min	100	100	100	100	100
ROR	47.86	47.51	47.62	47.61	47.28
Improved Ratio	52.14%	52.49%	52.38%	52.39%	52.72%

(f) (N_T=500, N_R=100, Q_R=30%, Q_T=20%, H_T=1)

H_Q	3	5	7	9	11
QOS Guided Min-Min	100	100	100	100	100
ROR	54.61	52.01	50.64	48.18	46.53
Improved Ratio	45.39%	47.99%	49.36%	51.82%	53.47%

Similar to those of Table 2, Table (a) changes the number of tasks to verify the reduction of resource that needs to be achieved by the *ROR* technique. We noticed that the *ROR* has significant improvement in minimizing grid resources. Comparing with the QoS guided Min-Min scheduling algorithm, the *ROR* achieves 50% ~ 60% improvements without increasing overall makespan of a chunk of grid tasks. Table (b) changes the number of machines. The *ROR* retains 50% improvement ratio. Table (c) adjusts percentages of QoS machine. Because this test has 20% QoS tasks, the *ROR* performs best at 15% QoS machines. This observation implies that the *ROR* has significant improvement when QoS tasks and QoS machines are with the same percentage. Table (d) sets 40% QoS machine and changes the percentages of QoS tasks. Following the above analysis, the *ROR* technique achieves more than 50% improvements when QoS tasks are with 45%, 60% and 75%. Tables (e) and (f) change the heterogeneity of tasks. Similar to the results of section 5.2, the heterogeneity of tasks is not critical to the improvement rate of the *ROR* technique. Overall speaking, the *ROR* technique presents 50% improvements in minimizing total resource need compare with the QoS guided Min-Min scheduling algorithm.

6. Conclusions

In this paper we have presented two optimization schemes aiming to reduce the overall completion time (makespan) of a chunk of grid tasks and minimize the total resource cost. The proposed techniques are based on the QoS guided Min-Min scheduling algorithm. The optimization achieved by this work is twofold; firstly, without increasing resource costs, the overall task execution time could be reduced by the *MOR* scheme with 7%~15% improvements. Second, without increasing task completion time, the overall resource cost could be reduced by the *ROR* scheme with 50% reduction on average, which is a significant improvement to the state of the art scheduling technique. The proposed *MOR* and *ROR* techniques have characteristics of low complexity, high effectiveness in large-scale grid systems with QoS services.

References

[1] A. Abraham, R. Buyya, and B. Nath, "Nature's Heuristics for Scheduling Jobs on Computational Grids", Proc. 8th IEEE International Conference on Advanced Computing and Communications (ADCOM-2000), pp.45-52, 2000.

[2] A. Andrieux, D. Berry, J. Garibaldi, S. Jarvis, J. MacLaren, D. Ouelhadj, D. Snelling, "Open Issues in Grid Scheduling", National e-Science Centre and the Inter-disciplinary Scheduling Network Technical Paper, UKeS-2004-03.

[3] R. Buyya, D. Abramson, Jonathan Giddy, Heinz Stockinger, "Economic Models for Resource Management and Scheduling

in Grid Computing", Journal of Concurrency: Practice and Experience, vol. 14, pp. 13-15, 2002.
[4] Jesper Andersson, Morgan Ericsson, Welf Löwe, and Wolf Zimmermann, "Lookahead Scheduling for Reconfigurable GRID Systems", 10th International Europar'04: Parallel Processing, vol. 3149, pp. 263-270, 2004.
[5] D Yu, Th G Robertazzi, "Divisible Load Scheduling for Grid Computing", 15th IASTED Int'l. Conference on Parallel and Distributed Computing and Systems, Vol. 1, pp. 1-6, 2003
[6] Fangpeng Dong and Selim G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", Technical Report No. 2006-504, 2006.
[7] Ligang He, Stephen A. Jarvis, Daniel P. Spooner, Xinuo Chen, Graham R. Nudd, "Hybrid Performance-oriented Scheduling of Moldable Jobs with QoS Demands in Multiclusters and Grids", Grid and Cooperative Computing (GCC 2004), vol. 3251, pp. 217–224, 2004.
[8] Xiaoshan He, Xian-He Sun, Gregor Von Laszewski, "A QoS Guided Scheduling Algorithm for Grid Computing", Journal of Computer Science and Technology, vol.18, pp.442-451, 2003.
[9] Jang-uk In, Paul Avery, Richard Cavanaugh, Sanjay Ranka, "Policy Based Scheduling for Simple Quality of Service in Grid Computing", IPDPS 2004, pp. 23, 2004.
[10] J. Schopf. "Ten Actions when Superscheduling: A Grid Scheduling Architecture", Scheduling Architecture Workshop, 7th Global Grid Forum, 2003.
[11] Junsu Kim, Sung Ho Moon, and Dan Keun Sung, "Multi-QoS Scheduling Algorithm for Class Fairness in High Speed Downlink Packet Access", Proceedings of IEEE Personal, Indoor and Mobile Radio Communications Conference (PIMRC 2005), vol. 3, pp. 1813-1817, 2005
[12] M.A. Moges and T.G. Robertazzi, "Grid Scheduling Divisible Loads from Multiple Sources via Linear Programming", 16th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS), pp. 423-428, 2004.
[13] M. Baker, R. Buyya and D. Laforenza, "Grids and Grid Technologies for Wide-area Distributed Computing", in Journal of Software-Practice & Experience, Vol. 32, No.15, pp. 1437-1466, 2002.
[14] Jennifer M. Schopf, "A General Architecture for Scheduling on the Grid", Technical Report ANL/MCS, pp. 1000-1002, 2002.
[15] M. Swany, "Improving Throughput for Grid Applications with Network Logistics", *Proc*. IEEE/ACM Conference on High Performance Computing and Networking, 2004.
[16] R. Moreno and A.B. Alonso, "Job Scheduling and Resource Management Techniques in Economic Grid Environments", LNCS 2970, pp. 25-32, 2004.
[17] Shah Asaduzzaman and Muthucumaru Maheswaran, "Heuristics for Scheduling Virtual Machines for Improving QoS in Public Computing Utilities", *Proc.* 9th International Conference on Computer and Information Technology (ICCIT'06), 2006.
[18] Gerald Sabin, Rajkumar Kettimuthu, Arun Rajan and P Sadayappan, "Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment", in the Proc. of the 9th International Workshop on Job Scheduling Strategies for Parallel Processing, LNCS 2862, pp. 87-104 , June 2003.
[19] Sriram Ramanujam, Mitchell D. Theys, "Adaptive Scheduling based on Quality of Service in Distributed Environments", *PDPTA'05*, pp. 671-677, 2005.
[20] T. H. Cormen, C. L. Leiserson, and R. L. Rivest, "Introduction to Algorithms", First edition, MIT Press and McGraw-Hill, ISBN 0-262-03141-8, 1990.
[21] Tao Xie and Xiao Qin, "Enhancing Security of Real-Time Applications on Grids through Dynamic Scheduling", *Proc.* the 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'05), pp. 146-158, 2005.
[22] Haobo Yu, Andreas Gerstlauer, Daniel Gajski, "RTOS Scheduling in Transaction Level Models", in Proc. of the 1st IEEE/ACM/IFIP international conference on Hardware/software Codesign & System Synpaper, pp. 31-36, 2003.
[23] Y. Zhu, "A Survey on Grid Scheduling Systems", LNCS 4505, pp. 419-427, 2007.
[24] Weizhe Zhang, Hongli Zhang, Hui He, Mingzeng Hu, "Multisite Task Scheduling on Distributed Computing Grid", LNCS 3033, pp. 57–64, 2004.

國科會補助計畫衍生研發成果推廣資料表

日期:2011/10/29

國科會補助計畫	計畫名稱: 應用於格網與可擴充平行系統之通訊最佳化, 廣播與品質服務排程技術之研究
	計畫主持人: 許慶賢
	計畫編號: 97-2221-E-216-011-MY3　　學門領域: 平行與分散處理

無研發成果推廣資料

97 年度專題研究計畫研究成果彙整表

計畫主持人：許慶賢	計畫編號：97-2221-E-216-011-MY3
計畫名稱：應用於格網與可擴充平行系統之通訊最佳化, 廣播與品質服務排程技術之研究	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	4	4	100%		
		研討會論文	2	2	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	5	5	100%	人次	
		博士生	2	2	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	4	4	100%	篇	所發表之期刊皆為 SCI 等級
		研究報告/技術報告	0	0	100%		
		研討會論文	6	6	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	- 主辦 2009 台俄雙邊研討會 - 平行多核心計算工具與應用 - 獲邀 ICEGT 2011 給予專題演講 - Best Paper Award, Computer Society of the Republic of China, 2010 - Best Paper Award, 2009 National Computer Symposium, 2009 - 協助建構 Taiwan UniGrid 平台。發展 UniBox 協助近 20 所國內大學加入 UniGrid 平台。成功開發 Data Replication and Management 子系統。

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：■已發表 □未發表之文稿 □撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：(以 100 字為限)

- The Journal of Supercomputing (2010)
- International Journal of Ad-Hoc and Ubiquitous Computing(IJAHUC) (2010)
- International Journal of Communication Systems (IJCS) (2009)

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）(以 500 字為限)

多叢集系統下之通訊最佳化技術，將格網應用程式的運算資料分配到計算節點，與計算節點屬於同一個叢集系統，或距離較近的叢集系統，可以降低處理器之間的通訊成本。由於多叢集系統大多是由異質性、動態網路連結而成，我們對此提出適應於異質性與動態環境的排程技術。此外，我們也探討異質性網路的訊息廣播技術，根據一般常用的單埠(one port)通訊模式，我們將提出多種適應於樹狀結構或一般網路架構下的訊息廣播技術，該研究克服動態網路的複雜問題。最後一個部份，我們探討多個以品質服務為基礎的資源與工作排程技術，包括資源成本(Economic Guided)與時間成本(Performance Guided)最佳化的技術。結合格網經濟模型，這一個部份的研究成果，可以導入實際的格網系統。針對雲計算與服務、敝人也在 International Journal of Ad-Hoc and Ubiquitous Computing (IJAHUC) 客座編輯一特刊 (guest editors: Ching-Hsien Hsu, Chung-Ming Huang and Jiannong Cao)，針對相關主題做進一步的探討。