在格網計算環境中異質性資源配置策略

賴冠州 *伍朝欽 楊大猶 黃振維 林士傑 吳勇均
臺中教育大學資訊科學學系 *彰化師範大學資訊工程學系
kclai@mail.ntcu.edu.tw

摘要

本研究探討在格網環境中異質性資源之配置策略；並分析在異質網路環境中，考慮傳輸花費模式時，工作排程與處理器資源配置機制之效能。本研究透過評估工作在各個不同的resource(或site)上執行時所能得到的工作完成時間，取得可最早完成工作的資源，以減少非必要性的傳輸花費，以便最早完成工作。
關鍵詞：格網計算、異質性、傳輸花費模式、資源配置。

1 Introduction

由於科技進步，使得計算能力或儲存空間之需求都呈指數型的成長。因此，越來越多的研究專注在藉由不同格網系統之發展，以便充分使用分散的計算與儲存資源[14]。格網計算已被證明是一個相當重要的研究領域，用以解決動態、各組織間合作的問題[3],[17]。換言之，格網系統所提供之計算能力與儲存空間，已遠遠超越當代之超級電腦。格網計算的概念源自於電力格網[18]。因此，格網計算的概念可延伸為：當使用者需要使用到網路上的計算資源時，只要連上網路即可取得。

然而一件工作在meta-computing環境當中的資源配置，比在任何一個單一平行化的電腦還要複雜許多。因為它存在著在多方面的高度異質性、不同的本地資源配置策略以及其他因素。

在許多的研究當中大多都有提及傳輸花費的重要性，但都尚未考慮到如何降低工作的傳輸花費。因此，在本研究當中，將針對以下幾點進行討論：

(1) 所有資源提出的工作都先送到Grid-Scheduler，再由Grid-Scheduler做工作分配。Grid-Scheduler會判斷一件工作是留在本地端處理即可，或者需要傳送到其他resource來幫忙處理。
(2) 如有需要傳送到其他resource來幫忙處理，該如何選擇resource來處理此工作。

本研究將針對以上兩點問題，來判斷工作適合在哪個計算資源中進行處理，以取得最佳的工作完成時間。

因此，本研究之目的在於設計一套processor allocation algorithm，透過條件判斷，計算出一件工作在不同resource計算的完成時間，再比較工作在各個不同resource上可以得到的工作完成時間，取得一個可以最早完成工作的resource，並將工作分配給此resource來執行。表 1為本研究所使用到的符號與其意思。

表 1 Meanings of Terms

S_i	初始工作
Res_ID	The resource id
GL	為Gridlet的縮寫
FT_{Lo}	工作在本地端評估的完成時間
ST	工作在resource的開始時間
numPE	執行一件工作所需要的PE數量
BR	為Baud_Rate的縮寫，代表resource傳送工作出去的速度

2 Related Works

Meta-computing一詞是在1992年由Smarr and Catlett兩位學者所提出[8]。Meta-computer是一種透過網路與適當的中介軟體將網路上的計算資源連結起來，就如同multi-site computing的技術一樣。用此技術，可讓使用者透過一個入口網站來遞交自己的工作，而不需要知道在入口網站的背後工作會如何的被處理。

下面將介紹幾個job scheduling與processor allocation的演算法，以及幾個過去研究中所提出的cost model。

2.1 Job Scheduling Algorithms

在此部份，介紹First-Come, First-Served (FCFS)與 List Scheduling (LS)兩種工作排程演算法。

2.1.1 FCFS

在作業系統當中的CPU排程演算法裡，最簡單、公平、且容易了解的方法是FCFS演算法。基本上，FCFS擁有一個佇列，排在佇列的第一位可以最先被服務，接著的會依照排隊的先後順序來提供服務。

2.1.2 LS

LS演算法的基本概念是：所有工作都有某些類型的優先權，再依某類型的優先權將工作重新排序，產生一個新的工作列表。有如下的判斷步驟：

(1) 從工作列表中，選擇擁有最高優先權的工作來進行排程。
(2) 從系統中選擇一個或一些可滿足這個工作的資源。

目前比較眾所皆知的LS策略有：The Earliest Starting Time first (EST), The Earliest Completion Time first (ECT), The Longest Processing Time first (LPT)等，都是依據不同的參數來決定工作的處理順序。

2.2 Processor Allocation Algorithms

在此部份，介紹四種處理器配置演算法。在此部份的問題是，如何將處理器分配給工作，以取得最佳化的排程長度。

2.2.1 Naive

這是一個比較簡單而且天真的演算法。它的想法是：在搜尋的範圍之內，只要滿足工作所要求的內容即可。因此，工作會由從系統中搜尋到的第一個resource開始分配PE，而這些resource沒有經過任何排序。

2.2.2 Largest Machine First (LMF)

LMF可以從它字面上的字義了解此演算法的運作過程。它主要的想法是：將格網系統中蒐集到的resource的資訊，依各個resource所擁有的閒置的處理器數量多寡，由大到小重新排序。再依工作所需要的處理器數量，從最大的resource開始進行分配。

2.2.3 Smallest Machine First (SMF)

在SMF演算法的介紹，可與LMF演算法作比較，因為其執行步驟與LMF演算法相同，所不同的地方在於兩者對處理器排序的方法。顧名思義，SMF演算法的分配順序為：從擁有最少處理器數量的resource為優先分配。

2.2.4 Minimum Effective Execution Time (MEET)

由Dr. Keqin Li所提出的MEET演算法[7]的目標是：當一個工作可進入排程時，則試著最小化它的有效執行時間。MEET演算法的處理過程為：各個resource需先依照處理器數量多寡，由大到小進行排列，找到 r 個resource可以提供足夠的處理器數量給工作進行執行，再將工作平均地分配到這個不同的resource上。此MEET演算法可避免工作切割的過小而產生過大的傳輸時間，也可將擁有較多處理器的resource保留下來。

2.3 Cost Models

在multi-site的工作排程上，已有許多的研究對communication cost model進行討論。這些models都討論著相同的問題，當一份工作被切割為許多子工作，且分配到不同的機器上執行時，如何將工作的所需要的執行時間(execution time) 從原本只需要計算時間 (computation time) t_i，因為在各個子工作之間又多了communication time，轉變為需要計算時間與傳輸、溝通時間(communication time) t_i^*，。

在文獻[7]之研究中，由於考慮到內部處理器間的傳輸溝通與外部處理器間的傳輸溝通時間，因此將公式定義如公式(1)：

$$t_{i,k}^* = t_{i,comp} + t_{i,comm}\frac{s_{i,k}}{s_i} + t_{i,comm}\left(1-\frac{s_{i,k}}{s_i}\right)\alpha \qquad (1)$$

此公式可進一步再化約成如公式(2)：

$$t_{i,k}^* = t_i\left(1+\left(1-\frac{s_{i,k}}{s_i}\right)(\alpha-1)\beta\right) \qquad (2)$$

其中 $\beta=\frac{t_{i,comm}}{t_i}$ 為工作的傳輸時間與執行時間的比率。從公式(2)中可看出，β 值較大時表示工作的執行時間幾乎都花費在傳輸溝通上；相反地，β 值較小時表示工作的執行時間幾乎都用在計算上。透過上述的公式，可得知每個子工作的執行時間 $t_{i,k}^*$，進而得知工作分配後的執行時間，也就是 $t_i^*=\max\left(t_{i,1}^*,t_{i,2}^*,\ldots,t_{i,r_i}^*\right)$。所以，可以透過公式(3)可更快速的找到工作所需的執行時間，因為當最小的子工作所需要的處理器數量越大時，表示執行時間最長的子工作所需的時間越短。

$$t_i^* = t_i\left(1+\left(1-\frac{1}{s_i}\left(\min_{1\le k\le r_i}\left(s_{i,k}\right)\right)\right)(\alpha-1)\beta\right) \qquad (3)$$

在文獻[2],[15]研究中，其前提為：所有在multi-site scheduling環境裡的工作必須先提交給格網排程者(Grid-Scheduler)，並且工作是可以跨resource執行的。

(1) 所有resource內所產生的工作，經由site內部的工作佇列提交到Grid-Scheduler。

(2) Grid-Scheduler在蒐集到所有的工作之後，開始進行判斷：是否有任何一個site能獨自提供足夠的資源來完成此工作。如果沒有辦法，則工作進行切割，將子工作分散到不同resource的機器上進行執行，且所有子工作需同時開始。

由於工作可能在切割過後分配到不同site上作執行。因此，需考慮到overhead，這個overhead是由於在較慢的網路環境中進行傳輸的動作所造成的。其cost model以公式(4)來表示：

$$t_i^* = t_i(1+p) \qquad (4)$$

在文獻[5]中提出了四種cost model的公式，所有在multi-site scheduling環境下的工作，如果有切割成不同的子工作，並分散到不同的機器上去作執行時，則需以這四種公式計算出工作所需的執行時間，如表2所示。

表2公式轉換對照

原始公式	轉換後公式
$(1+p)^f$	$t_i^* = t_i(1+p)^{r_i}$
$\max_i\left[(1+p)^{r_i}\right]$	$t_i^* = t_i \max_{1\le k\le r_i}\left((1+p)^{s_{i,k}}\right)$
$(1+p)*f$	$t_i^* = t_i(1+p)r_i$
$\max_i\left[(1+p)*r_i\right]$	$t_i^* = t_i \max_{1\le k\le r_i}\left((1+p)s_{i,k}\right)$

在文獻[4]研究中，使用了同步與非同步兩種類別的工作來進行實驗。在此研究中的排程者需具備著能依據所使用的機器及所分配到的處理器來對工作的預期runtime作預估，因為在一個速度較慢的WAN網路環境中，機器的數量會影響到工作的處理時間。在同步的工作中，暗示著其存在著某種層級的平行化，因此在同步與非同步工作之間最大的差別是，這兩個類別的工作在溝通、傳輸效能與預期run time之間的相依性問題。因此，在此研究中提出了下面公式(5)：

$$t_i^* = \frac{Seq_i}{S\left(\sum_{k=1}^{r_i} s_{i,k} P_k\right)}\left(1+\beta(\alpha-1)r_i\right) \quad (5)$$

3 Proposed Algorithm

3.1 Problem Definition

在本研究的meta-computer的環境中，不同resource內擁有不同的node數量之外，另外再將每個node的架構設定為異質性，也就是使其processor的數量不同及processor的MIPS rating不一，而其異質性則以隨機的方式產生。

在本研究的實驗架構中，每個resource擁有自己的scheduler，在每個resource之上以一個meta-computer來形成此環境，且在meta-computer中有個負責接收所有resource所提出的工作要求的Grid-Scheduler(也就是meta scheduler)，由接收所有工作要求並進行工作分配的計算與協調。在經過Grid-Scheduler計算之後，會將工作分配到最適合的resource執行，分配後的工作會由resource的scheduler接收，並且依照自己resource內的policy進行工作處理。因此，在實驗環境中，Grid-Scheduler扮演著一個集中式的排程者角色，來進行工作在分散式系統中的工作分配之計算與協調。

在真實環境中存在著許多不同類型的resource與job，在job找到適合的resource前，必須對格網系統中的資源進行初探性的選擇，在此忽略此種在資源與工作之間preselection的處理，而聚焦於排程的結果。

3.2 Proposed Algorithm

首先以 S_i 表示此次要處理的工作，演算法在一開始先判斷執行工作的 Site ID。若執行工作的 Site ID 等於提出工作(Gridlet)要求的 Site ID 時，則此工作被判斷為屬於由 local site 所提出。因此，此工作在 local site 上執行所能得到的完成時間為：工作在 local site 上可以開始的時間加上工作的計算時間(工作所需要的 time units，乘上執行工作需要的 PE 數量，接著再除以 local site 的 total time units。

若判斷執行工作的 Site ID 不等於提出工作要求的 Site ID 時，則此工作被判斷為屬於由 remote site 所提出。因此，需進入下一層的條件(傳輸速率)判斷：

(1) 當執行工作的 Site 之傳輸速率小於或等於提出工作要求的 Site 時，則此工作在 remote site 上執行所能得到的完成時間為：工作在 remote site 上可以開始的時間加上工作的計算時間(工作所需要的 time units，乘上執行工作需要的 PE 數量，接著再除以 remote site 的 total time units)，再加上工作的傳輸時間(工作的檔案大小，乘上執行工作需要的 PE 數量，接著再除以執行工作的 site 之傳輸速率)，即可取得工作的 finish time。

(2) 反之，當執行工作的 Site 之傳輸速率大於提出工作要求的 Site 時，則此工作在 remote site 上執行所能得到的完成時間為：工作在 remote site 上可以開始的時間加上工作的計算時間(工作所需要的 time units，乘上執行工作需要的 PE 數量，接著再除以 remote site 的 total time units)，再加上工作的傳輸時間(工作的檔案大小，乘上執行工作需要的 PE 數量，接著再除以提交工作要求的 site 之傳輸速率)，即可取得工作的 finish time。

經由以上幾點判斷與公式的計算，可取得工作在各個 site 上評估後的 finish time，再從各個 finish time 中挑選出最早的(也就是最小的)完成時間，取得此最早完成時間之 site 的 id，並將工作分配給此 site 計算。本研究之演算法列於圖 1。

Input: S_i and $\left(P_1^{'}, P_2^{'}, \ldots, P_m^{'}\right)$.
Output: $\left(s_i, P_j, P_{j_s}, P_{j_f}\right)$.

// compute the finish time on every resources for a Gridlet(a job)
if (Res's_ID == GL's_Res_ID) {

$$FT_{Lo} = ST + \frac{GL's\_len \times numPE}{Res's\_Total\_Job\_Time\_Units};$$

} // local resource (no communication time)
else (Res's_ID != GL's_Res_ID) {
  if (ExecRes's_BR <= GL's_Res_BR) {

$$FT_{Re\_ExecR} = ST + \frac{GL's\_len \times numPE}{Res's\_Total\_Job\_Time\_Units} + \frac{GL's\_file\_size \times numPE}{Res's\_BR};$$

  } else {

$$FT_{Re\_JobR} = ST + \frac{GL's\_len \times numPE}{Res s\_Total\_Job\_Time\_Units} + \frac{GL's\_file\_size \times numPE}{GL's\_Res\_BR};$$

  }
end if;
} // remote resource
end if;

// compare the finish time of every resources' to get a earliest finish time
// of the Gridlet
double earliest_FT = FT[0];
for (Res's_ID = 1 ; Res's_ID <= Total_Resource.length ; Res's_ID++) {
  if (FT[Res's_ID] < earliest_FT) {
    earliest_FT = FT[Res's_ID];
  }
  end if;
}
end for;
return $\left(s_i, P_j, P_{j_s}, P_{j_f}\right)$.

圖 1 Proposed Algorithm

3.3 A Simple Example

在這個部份，以一個簡單的例子解釋所提出的演算法。表 3為resource的資訊；表 4為使用者所提出的工作資訊。在此以FCFS為工作排程演算法與本研究所提出的演算法為處理器配置算法，並將工作由大至小排序後進行處理。其結果列於表 5。

表 3 Resources' Information in the Grid System

	Res's Name	Machine's Name	MIPS/s of a PE	Number of PE	Res's Total MIPS/s	BR
Meta-computer	Site_1	Machine_1	2	6	59	5
		Machine_2	9	5		
		Machine_3	1	2		
	Site_2	Machine_1	3	9	27	8
	Site_3	Machine_1	6	7	102	3
		Machine_2	10	6		
	Site_4	Machine_1	9	10	100	10

表 4 Information of Jobs Submitted by Users

User's Locality	id	length	file_size	output_size	numPE	Total length
Site_1	0	6	5	1	2	12
Site_2	1	8	7	10	4	32
Site_3	2	3	3	8	7	21
Site_4	3	7	10	10	2	14
Site_1	4	3	9	10	6	18
Site_2	5	8	3	7	1	8
Site_3	6	4	9	6	9	36
Site_4	7	9	1	1	4	36

表 5 Job Execution Process of the Algorithm Proposed in this Study

		Comp-Time	Comm-Time	Exec-Time	Res's ST	Res's FT	Earliest FT
Gridlet6	R1	0.61	27	27.61	0	27.61	0.35
	R2	1.33	27	28.33	0	28.33	
	R3	0.35	0	0.35	0	0.35	
	R4	0.36	27	27.36	0	27.36	
Gridlet2	R1	0.36	7	7.36	0	7.36	0.56
	R2	0.78	7	7.78	0	7.78	
	R3	0.21	0	0.21	0.35	0.56	
	R4	0.21	7	7.21	0	7.21	
Gridlet4	R1	0.31	0	0.31	0	0.31	0.31
	R2	0.67	10.8	11.47	0	11.47	
	R3	0.18	18	18.18	0.56	18.74	
	R4	0.18	10.8	10.98	0	10.98	
Gridlet1	R1	0.54	5.6	6.14	0.31	6.45	1.19
	R2	1.19	0	1.19	0	1.19	
	R3	0.31	9.33	9.64	0.56	10.2	
	R4	0.32	3.5	3.82	0	3.82	
Gridlet7	R1	0.61	0.8	1.41	0.31	1.72	0.36
	R2	1.33	0.5	1.83	1.19	3.02	
	R3	0.35	1.33	1.68	0.56	2.24	
	R4	0.36	0	0.36	0	0.36	
Gridlet0	R1	0.2	0	0.2	0.31	0.51	0.51
	R2	0.44	2	2.44	1.19	3.63	
	R3	0.12	3.33	3.45	0.56	4.01	
	R4	0.12	2	2.12	0.36	2.48	
Gridlet3	R1	0.24	4	4.24	0.51	4.75	0.5
	R2	0.52	0.74	1.26	1.19	2.45	
	R3	0.14	0.2	0.34	0.56	0.9	
	R4	0.14	0	0.14	0.36	0.5	
Gridlet5	R1	0.14	0.6	0.74	0.51	1.25	0.96
	R2	0.3	0	0.3	1.19	1.49	
	R3	0.08	1	1.08	0.56	1.64	
	R4	0.08	0.38	0.46	0.5	0.96	

4 Simulation Environment

GridSim simulation tool [11]以SimJava為基礎所開發而成，並以Java程式語言作為執行的discrete-event simulation package。它是以分層式的架構所組成的，因此允許使用者在架構中新增新元件或者新增新的一層。本研究以GridSim作為實驗工具，圖 2為本研究模擬環境之模組。Grid Resource模組為產生格網系統中計算資源的參數，如resource計算能力與其傳輸速度。Grid User and Gridlet模組是用來產生使用者及其所提出的工作要求。Grid-Scheduler模組，為一個接收系統中使用者所提交出來的工作之排程者。Simulation Program with Scheduling Algorithms模組，為一個使Grid Resource、Grid User and Gridlet、Grid-Scheduler等模組進行工作處理之模擬環境。Experimental Results模組，則將實驗模擬結果以文字方式呈現出來，以作為之後圖形化圖表的比較與分析之依據。

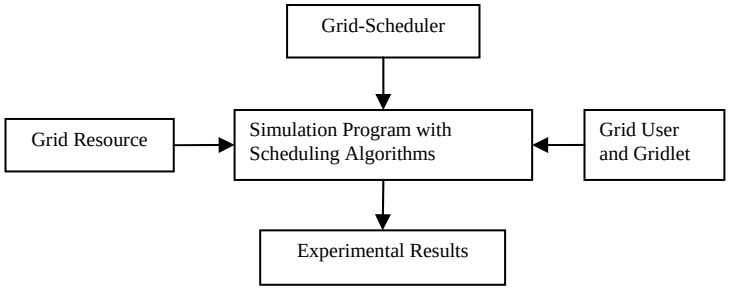

圖 2 Simulation Environment

5 Experimental Results

5.1 Experimental Arguments

本研究實驗建立在四種meta-computer上；由於文獻[7]研究中證明MEET演算法優於其他演算法，因此，本研究將集中MEET演算法與本研究所提出之演算法的比較。表 6為本研究之meta-computer環境。

表 6 各meta-computer所擁有的PE數

Resource Name	WWG	Equal	4Small_4Big	Power2_2
Resource0	4	8	4	2
Resource1	10	8	4	4
Resource2	2	8	4	8
Resource3	22	8	4	16
Resource4	16	8	12	2
Resource5	2	8	12	4
Resource6	4	8	12	8
Resource7	8	8	12	16
Total PE	68	64	64	60

本研究之工作類型與其參數設定如表 7。每個工作在處理時會經由LJF排序。每個類型的工作皆有五種平均大小的工作(每個工作需要的PE數量)，此五種平均為：$\bar{s}=10,15,20,25,30$，$\bar{s}$表示工作所需處理器之平均個數。每種類型的工作中的各個平均皆有50個list的工作列表，每個工作列表中包含1000個jobs，這1000個jobs平均且隨機分散在$\left[1..2\bar{s}-1\right]$的範圍之內。

表 7 工作類型與參數設定

	Comp-	Comm.-	Comp- & Comm.-
Length	1400	14	1400
File_size	14	1400	1400
Output_size	1	1	1

5.2 Comparison Metrics

表8為本實驗研究在四種不同meta-computer系統中，所得之 average earliest finish time of lists of jobs實驗結果。在每個 meta-computer 系統中，以 LJF-LS-MEET 和 LJF-LS-Proposed兩種initial job ordering、job scheduling 和 processor allocation algorithms 的組合，搭配上Comp-、Comm-、與Comp-&-Comm-三種initial job ordering，以陳列在四個meta-computers上使用不同演算法所得之the average earliest finish time of lists of jobs結果。

表 8 Collection of all Results with Different Algorithms on Four Meta-computers

	LJF-LS-MEET			LJF-LS-Proposed		
	Comp-	Comm.-	Comp- & Comm-	Comp-	Comm.-	Comp- & Comm-
$\bar{s}$	(a) on WWG					
10	578	22230	22576	756	28	3150
15	1221	58119	58721	1120	51	4783
20	2080	114860	115627	1472	65	6135
25	3097	189337	189960	1838	86	7757
30	4684	306030	305464	2205	102	9305
$\bar{s}$	(b) on Equal					
10	700	27354	27784	697	6	925
15	1512	76039	76756	1032	9	1393
20	2611	160193	164415	1358	13	1801
25	4222	283768	283812	1695	17	2257
30	5587	382809	385226	2032	22	2753
$\bar{s}$	(c) on 4Small_4Big					
10	495	16612	16986	735	12	1486
15	1047	44070	44785	1092	20	2223
20	1972	96073	96536	1434	30	2887
25	3220	186593	187893	1786	37	3595
30	4269	261543	262549	1786	47	4396
$\bar{s}$	(d) on Power2_2					
10	376	13566	13808	800	14	1836
15	817	36677	36920	1185	23	2702
20	1499	81401	82160	1559	33	3592
25	2532	153673	155221	1941	42	4367
30	3450	220724	222083	2327	51	5232

根據表8可得到圖3, 4, 5之結果。在下列各圖中可看出幾種現象：

(1) 在圖3中，其工作類型為重視計算能力，而不重視傳輸能力。因此length設定為1400。當 $\bar{s}$ 小時，以MEET作為processor allocation的演算法時，會得到比較佳的結果，因為它有效的將工作平分到各個PE上執行，因此使得平均結果較小；當 $\bar{s}$ 大時，以本研究所提出之演算法作為processor allocation的演算法時，會得到比較佳的結果，因為它考慮到工作在哪個resource上進行會得到較佳的結果，可有效的將工作完成時間平均的分配到各個resource上，因此使得平均結果較小。

(2) 在圖4與圖5中，其工作類型皆為高度重視傳輸，因此file_size設定為1400。以MEET作為processor allocation的演算法時，不管工作大小為何、工作是否經過排序，其所得結果皆因為沒有考慮到傳輸，導致大量非必要性的傳輸花費，而拉長整個排程長度；相對的，使用本研究所提出之演算法作為processor allocation的演算法時，因為它除了將計算時間加入考慮之外，也將工作可以開始的時間、工作所需要的傳輸花費，都在工作的評估階段加入考量，因此，可使非必要性的傳輸花費節省下來，避免拉長整個排程長度，以致於可得到較佳的結果。

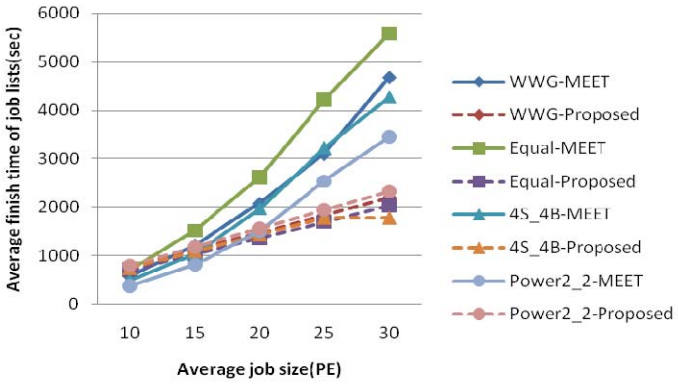

圖 3 Comparison of LJF Computation-intensive Jobs

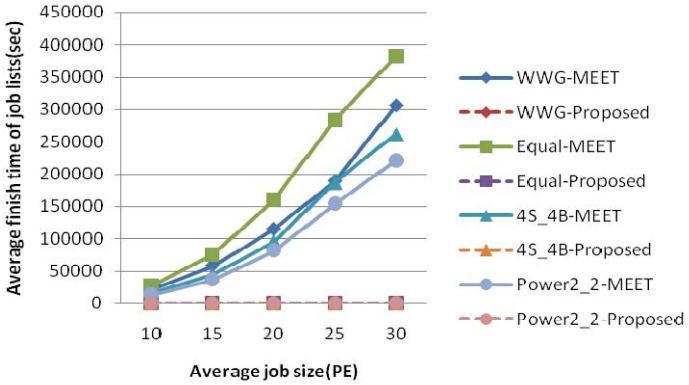

圖 4 Comparison of LJF Communication-intensive Jobs

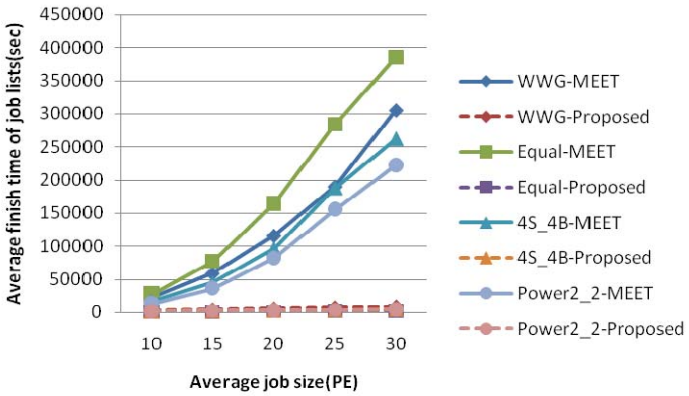

圖 5 Comparison of LJF Computation- & Communication-intensive Jobs

5.3 Influence Factors

在此組合式的評估方法當中，包括one initial job ordering strategies(LJF)、one job scheduling algorithm(LS)、two processor allocation algorithms(MEET, proposed in this study)。從實驗數據中可得知，在computation-intensive的工作類型下，若工作太小，則無法顯示出本研究所提出之演算法的效用；相反地，在工作普遍都大時，則可顯示出本研究之方法明顯優於MEET演算法。若在高度重視communication的情況之下，由於本研究之方法在分配工作前的工作評估有加入工作的開始時間與工作所需的傳輸花費之考慮，因此，可在節省不必要的傳輸花費之下，找到工作的最早完成時間；相反地：MEET演算法只單純考慮系統中是否有足夠的resource可以提供目前的工作，而沒有考慮到加上傳輸花費後，是否值得將工作分配出去，因此，可能造成大量不必要的傳輸花費，而拉長整體的排程長度。

6 Conclusions and Future Works

本研究提出了一種異質性資源配置的演算法，此演算法透過將處理器的計算能力(MIPS)轉換成每秒所能提供的time unit進行評估處理。因此，在加入communication cost model的格網系統中，使用此演算法可取得工作的最早完成時間、減少不必要的資料傳輸花費、縮短整個工作列表的完成時間、且能增加系統的使用率。

未來本研究將可考慮下列方向：

(1) 動態的排程方法：將靜態的排程方法修改為動態的排程方法，系統中可能隨時有新工作的產生。

(2) 多個Grid-Scheduler：本研究屬於一種集中式的排程機制，工作透過一個集中式的Grid-Scheduler，進行工作的評估，以將工作分配至分散式的系統當中。因此，當工作量大時，可能在單一Grid-Scheduler處理時產生瓶頸。

Acknowledgement

This work was sponsored in part by the National Science Council of the Republic of China under the contract number: NSC 95-2622-E-142 -001 -CC3.

References

[1] Buyya, R., and Murshed, M., "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," The *Journal of Concurrency and Computation: Practice and Experience (CCPE)*, Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[2] Ernemann, C., Hamscher, V., Schwiegelshohn, U., Yahyapour, R., and Streit, A, 2002, "On Advantages of Grid Computing for Parallel Job Scheduling," In *Proceedings of the 2nd IEEE/ACM international Symposium on Cluster Computing and the Grid* (May 21 - 24, 2002). CCGRID. IEEE Computer Society, Washington, DC, pp. 39-46.

[3] Foster, I., and Kesselman, C., The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, San Francisco, CA, 1999.

[4] Gehring, J., and Preiss, T., "Scheduling a Metacomputer with Uncooperative Sub-schedulers," In *Proceedings of the Job Scheduling Strategies For Parallel Processing*, Lecture Notes on Computer Science Vol. 1659, pp. 179–201, San Juan, Puerto Rico, April 1999.

[5] Hamscher, V., Schwiegelshohn, U., Streit, A., and Yahyapour, R., "Evaluation of Job-Scheduling Strategies for Grid Computing," In *Proceedings of the First IEEE/ACM international Workshop on Grid Computing*, Lecture Notes on Computer Science Vol. 1971, pp. 191-202, Springer-Verlag, London, 2000.

[6] LCG Project Overview, http://lcg.web.cern.ch/LCG/overview.html.

[7] Li, K., "Job scheduling and processor allocation for grid computing on metacomputers," *Journal of Parallel and Distributed Computing*, Vol. 65, Issue. 11, pp. 1406-1418, November 2005.

[8] Smarr, L., and Catlett, C.E., "Metacomputing," *Communications of the ACM*, Vol. 35, Issue. 6, pp. 44-52, June 1992.

[9] Sulistio, A., Yeo, C. S., and Buyya, R., "A Taxonomy of Computer-based Simulations and its Mapping to Parallel and Distributed Systems Simulation Tools," International Journal of Software: Practice and Experience, Volume 34, Issue 7, Pages: 653-673, Wiley Press, USA, June 2004.

[10] Taiwan UniGrid Project, http://www.unigrid.org.tw/index.html.

[11] The GRIDS Lab and the Gridbus Project, http://gridbus.csse.unimelb.edu.au/gridsim/.

[12] TIGER Grid Broker, http://gamma2.hpc.csie.thu.edu.tw/.

[13] TIGER Grid, http://gamma2.hpc.csie.thu.edu.tw/ganglia/.

[14] Wang, C.M., Chen, H.M., Chang, C.C., and Wu, J.J., "A High-Performance Virtual Storage System for Taiwan UniGrid," *Proceedings of the Third Workshop on Grid Technologies and Applications*, CHU, Hsinchu, Taiwan December 7-8, 2006.

[15] Yahyapour, R., "Design and evaluation of job scheduling strategies for grid computing," Doctoral Thesis, Computer Engineering Institute, Lehrstuhl für Datenverarbeitungssysteme, November 2002.

[16] Yang, C.T., and Ho, H.C., "A Shareable e-Learning Platform Using Data Grid Technology," In Proceedings of the 2005 IEEE international Conference on E-Technology, E-Commerce and E-Service (Eee'05) on E-Technology, E-Commerce and E-Service, EEE, IEEE Computer Society, Washington, DC, pp. 592-595, March 29 - April 01, 2005.

[17] Yang, C.T., Chen, S.Y., Chen, T.T., Yeh, Y.C., and Chen, C.Y., "Design and Implementation of Monitoring and Information Services Using Ganglia and NWS for Grid Resource Broker," Proceedings of the Third Workshop on Grid Technologies and Applications, CHU, Hsinchu, Taiwan December 7-8, 2006.

[18] 林誠謙, 李世昌, 鄧炳坤, "下一代網際網路新紀元－全球格網(World Wide Grid)," 自然科學簡訊, 第十五卷第四期, pp. 123-124, November 2003.