SMP 叢集上考量記憶體之平行排程演算法
Memory Consideration for Parallel Job Scheduling on SMP Clusters

王逸民Yi-Min Wang, 簡克達Ko-Ta Chien, 張沅合Yuan-He Chang, 田尚修Shang-Hsin Tien
私立靜宜大學資訊管理學系
ymwang@pu.edu.tw, g9471001@pu.edu.tw, s9371006@pu.edu.tw, s9371048@pu.edu.tw

摘要

近年來，需要大規模計算、大量記憶體、以及儲存空間的問題，例如生物資訊、物理學、天文學、氣象預測、地震預報，以及各式各樣的商業問題等等，越來越多，且日益複雜。為了幫助人們解決這些問題，電腦科學家們努力不懈投入與高性能計算環境相關的軟硬體之研究與開發，叢集系統就是其中之一。

本論文主要在SMP（Symmetric Shared-Memorg Multiprocessor）cluster上，parallel jobs的backfilling scheduling 演算法研究。在叢集環境上，為了能有效利用系統資源，排程方法考慮的元素也從只考慮CPU（Central Processing Unit）到同時考慮CPU以及記憶體兩者，先前學者已經證實同時考慮CPU以及記憶體能有效減少反應時間，然而若嚴格限制系統資源必須足夠提供工作所需求的記憶體和CPU，才能允許該工作執行，這樣的作法導致許多jobs必須花費相當長的等待時間。

我們提出一個有限度放寬記憶體容量上限的方式，當記憶體不足以讓工作執行的時候，便將排程方法所允許的系統記憶體上限略為提高，讓原本無法執行的工作能夠執行，因此該工作的等待時間便能夠縮短。然而此方法會讓所有執行中工作的總需求記憶體比原本高，因此系統會產生較多的page faults，導致執行中工作的執行時間將比原本的情況下還要長，所以總體的執行時間拉長。我們的排程方法便是在等待時間和執行時間兩者中找出最佳化。我們撰寫模擬器模擬一個較大規模叢集系統，使用具有公信力的trace，實驗的結果證實我們提出的有限度放寬記憶體容量上限能讓反應時間更進一步的減少。

關鍵字：叢集、工作排程、記憶體考量。

Abstract

This paper studies the parallel job scheduling algorithm on SMP clusters. To reduce the number of page faults and shorten the executing time of jobs, previous research suggests that we must consider CPU and memory at the same time. That is, the ready jobs can't be executed until they get enough memory and CPU resources. In this way, as the memory resource can't satisfy the running jobs immediately, the jobs must spend time in waiting memory.

Instead of waiting for enough memory, in this paper, we suggest that the memory restriction can be lessened. Our method will reduce the waiting time without increasing too many page faults. That is, the method balances the execution time and waiting time at the same time. Experimental results show that our method performs better than some known job scheduling algorithms.

Keywords: Cluster, Job scheduling, Memory consideration

1 緒論

本論文主要在SMP cluster上，parallel jobs的scheduling問題的研究。先前學者所作出的研究[3]指出，在不考慮記憶體的因素下，有時會因為記憶體不足使得作業系統頻繁的產生page faults，因而延長工作的執行時間，而多考慮了記憶體容量之後，雖然工作的等待時間會延長，但是總體的平均反應時間會下降。

本研究將工作的等待時間以及因為記憶體不足多產生page faults而延長執行時間兩者間，再作進一步的研究。我們的排程方法同時考慮CPU和記憶體需求，而當缺乏記憶體時，我們的排程方法在對系統的記憶體容量上限這部分加以考慮，有限度地提升記憶體容量上限，更新之後，在排程方法所允許的記憶體容量範圍之內考慮工作是否能夠執行，可以執行的情況下便執行該工作，不可執行的情況下該工作便繼續留在等待佇列，我們將分別在EASY（the Extensible Argonne Scheduling System）、FCFS（First Come First Service）中加入此概念 [1][2][4][5]。

本實驗使用C語言撰寫一個叢集系統模擬器，在工作數據方面採用具有公信力的實驗數據[6][7]，模擬該工作數據在叢集系統上運作的情形，搭配不同的排程方法比較我們改良的排程方法所增進的效能多寡，實驗結果指出適度放寬執行中的工作總合需求記憶體容量上限，跟原本的FCFS以及EASY比較，能夠更有效地降低總體的平均反應時間。

本論文分為以下幾個部分：2.相關研究，簡單介紹相關排程方法的研究以及記憶體考慮相關的論文；3.有限度開放記憶體容量上限的排程方法；4.實驗環境，介紹我們實驗的環境；5.實驗結果；6.結論及未來發展方向。

2 相關研究

最簡單的排程方法是FCFS，此方法依照工作進入系統的時間依序執行，沒有考慮工作需求資源的差異以及需求龐大的工作造成的路障情形，FCFS的缺點是當有龐大資源需求的工作等待執行，因為它必須等待足夠資源才能執行，在這段等待期間使得系統有相當多的資源未被使用，這樣的方式造成整個叢集系統的效能低落。而後一個新的排程概念—回填法（backfilling）被提出來以增進系統使用率，在工作queue後面的job，若他所需要的資源較少，則有機會可以超越在前面的工作先執行，目的在於充分利用系統空閒的資源，著名的有conservative backfill（FCFS with backfilling）[1][2][4]和aggressive backfill（EASY）[1][2][5]。

在conservative backfill中，每個在等待佇列中的工作都擁有一個保留權，這個保留權依照該工作進入系統的時間排順序，保留權的功用為確保工作執行的順序與使用資源的權利，在queue中較後面的job若要超越前面的job先執行，規定有兩個：(1)現在系統中有足夠資源可供該工作使用；(2)該工作不能影響到先前進入系統而在等待佇列所有工作的執行優先權益。

在aggressive backfill中，只有在等待佇列中的第一個工作擁有保留權，而除了第一個工作之外，其他工作都可以被超越，在queue中較後面的job若要超越前面的job先執行，其規定有兩個：(1)現在系統中有足夠資源可供該工作使用；(2)該工作不能影響到等待佇列中的第一個工作。EASY能夠將現有的系統資源盡量做最大額度的利用，但是需求龐大的工作在EASY排程之下可能會有等不到足夠系統資源的情況，此為EASY排程方法的缺點。

上述的排程方法所考慮到的系統資源只考慮CPUs是否足夠讓工作使用，由於沒有考慮到記憶體的因素，因此排程方法分配給系統的工作，其所需記憶體可能超出系統負荷，如此系統在執行工作的時候，會頻繁地執行內文切換，進而造成執行時間的上升。在[3]中提出了將Gang加進了記憶體容量的考慮因素，當系統內部的記憶體和處理器個數不足以讓一個工作執行的時候，此工作便會繼續放置在等待佇列直到記憶體和處理器個數足夠使用，結果證實加進了記憶體考慮之後，能大幅度地降低平均反應時間。

前面的排程方法促使了我們將記憶體考慮納入space sharing方式的排程方法之中，然而同時考慮兩者雖然保證工作的執行時間不會變長，卻增加了工作的等待時間。如果能讓工作提前執行，在系統超出一個限度的記憶體上限之內，那麼工作所減少的等待時間，可以抵平因為內文切換所增加的工作執行時間，使得反應時間能再進一步地降低。本研究基於這個想法，對於space sharing的排程方法，加進了記憶體考慮的因素，同時也加進了記憶體上限的考慮方式。

3 有限度開放記憶體容量上限的排程方法

本研究從基本的space sharing排程方法為出發點，從原本只考慮處理器個數的排程考慮方式，進展到能同時考慮處理器個數和記憶體容量的排程方法，為了能夠觀察系統超出記憶體上限不同所造成的影響，對於記憶體容量考量的部分作了進一步的修改，公式(1)和(2)為判斷工作能否從等待佇列轉移到執行狀態的判斷式，公式(1)為考慮處理器個數是否足夠讓工作使用，當工作需求的處理器個數小於等於現有系統能提供的處理器個數，公式(1)便能夠成立。公式(2)為考慮記憶體容量是否足夠讓工作使用，並且加入超出記憶體上限的變數C，C為排程方法預設的超出記憶體上限百分比，工作需求的記憶體容量小於等於系統能提供的記憶體容量，公式(2)便成立，當公式(1)和(2)皆同時成立，工作才允許執行。

$$\text{Job require processors} \leqq \text{physical processors} - \text{used processors} \quad (1)$$

$$\text{Job require memory} \leqq \text{physical memory} * (1+C) - \text{used memory}, \quad C \geqq 0 \quad (2)$$

當系統使用的記憶體超出了可用容量上限，也就是我們將公式(2)中的C值設定為大於0的情況下，系統將會增加對執行中的工作做內文切換的動作，這個動作會用到較頻繁的磁碟讀取和寫入的動作。針對超出記憶體的比例和因此增加的內文切換導致工作執行時間增加的關係，現在的研究還沒有能準確預測的模型出現，我們根據[8]中對於記憶體容量跟內文切換關聯性的公式，改良成一記憶體懲罰模型（在3.2節加以詳述）。

3.1 加入等待時間考慮因素

為了增加考慮記憶體容量上限的排程方法之穩定性，我們針對擁擠情況下，等待中工作是否提前執行的判定，額外增加等待時間上的判斷，每個工作會因應他所需求的執行時間給予一個相對應的等待時間門檻值（公式(3)），在未超出門檻值的時候，等待中工作依照一般的排程方法排程，也就是只考慮CPU和記憶體是否足夠使用，當超出門檻值並且系統在擁擠情況時，等待中工作就能依照我們所提出來的考慮記憶體上限的新排程方法，來決定該工作是否能提前執行。

$$\text{Wait Time Threshold} = \text{Job's run time} * \text{Threshold} \quad (3)$$

在新FCFS及EASY排程方法裡，除了考慮資源方面加進了對記憶體上限的考量，其他排程步驟依然與舊有的排程方法相同。當執行中工作的執行時間增加，則本來預估可以執行的等待工作有可能要再等上一段時間，這多出的等待時間便是讓工作提前執行所做的犧牲，我們所期望的就是提前執行所減少的等待時間，能夠勝過因提前執行所增加的執行時間以及後續等待中工作的延誤等待時間。

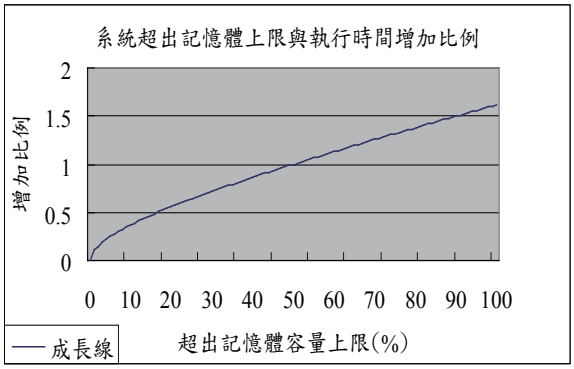
3.2 記憶體懲罰模型

公式(4)為我們參考[8]所提出來的內文切換增加比例的簡化模型，M*為現在所使用的記憶體容量大小，M為系統原本記憶體容量大小，N為我們因為記憶體不足所必須增加執行時間的增加比例，圖一為超出記憶體上限程度以及增加比例的對照圖，表1為增加比例範例。現在假設有一個執行時間為10秒的工作進入系統，而現在的排程方法允許超出記憶體容量上限60%，當該工作允許執行時剛好超出系統上限50%，依照我們的增加比例模型，參考圖一，當系統超出記憶體上限為總記憶體容量的50%的時候增加比例為1，於是將該工作以及所有執行中工作的剩餘執行時間乘上2倍，於是該工作需要執行20秒，其他執行中工作的剩餘執行時間也乘上2倍。

$$N=\frac{1}{2}(H+\sqrt{H^2-4})-1,$$

其中H=1+M*/M, M*≧M (4)

計算出增加比例之後，針對現有系統中執行工作的剩餘執行時間，乘上我們所計算出的增加比例，公式(5)結果便是我們因為超出記憶體容量之後的改變的剩餘執行時間（會比原來多），爾後當一些執行中的工作結束之後，此時記憶體壓力減輕，我們要將工作的執行時間回復到原本它應該的剩餘執行時間，我們套用公式(6)。



圖一　系統超出記憶體上限與執行時間增加比例

表1　超出記憶體上限與時間增加比例範例表

M*/M	H	N
1.1	2.1	0.37
1.3	2.3	0.72
1.5	2.5	1

For （all running jobs）
Job’s exec_time=Job's exec_time*(N+1)　(5)
For （all running jobs）
Job’s exec_time=Job's exec_time*1/(N+1)　(6)
其中exec_time為execution time

4 實驗環境

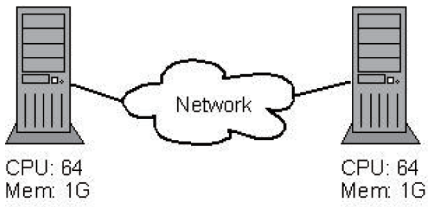
我們運用模擬器搭配上具有可信度的工作數據（Workload trace）DAS2-fs2[6]，來進行本研究的實驗。為了將原始數據的需求符合現有的硬體資源水準，將DAS2-fs2所有工作的記憶體需求等比例放大5倍，表2、表3為實驗數據DAS2-fs2 的工作數據，其中表3顯示在各種執行時間範圍下的jobs個數以及這些jobs的平均記憶體需求（單位為Mbytes），此數據顯示大部分的jobs 對CPU及memory的需求量都不大，蠻符合一般的情形。本研究模擬的平台為一個SMP Clusters，圖二為模擬硬體架構的表示圖，由兩個具有64個CPUs以及1G記憶體的SMP系統組成一個Clusters。

表2　DAS2-fs2工作需求處理器個數與記憶體容量分布

CPU需求	Job 數目	記憶體需求（MB）	Job 數目
0-5	32622	100	40921
6-11	10487	200	6399
12-17	7559	300	4266
18-23	873	400	6312
24-29	4503	500	2896
30-34	1402	600	223
35-40	1798	700	20
41-46	726	800	0
47-52	1058	900	0
53-58	12	1000	2

表3　DAS2-fs2工作執行時間與平均記憶體需求分佈

執行時間（小時）	job數目	平均記憶體需求（MB）
0-5	60903	107.09
6-10	24	81.81
11-15	17	115.84
16-20	10	88.46
21-25	16	136.62
26-30	9	91.44
31-35	3	175.15
36-40	14	101.50
41-45	5	60.12
46以上	39	368.20



圖二　128個CPUs、2G總記憶體的SMP Clusters

4.1 評估標準

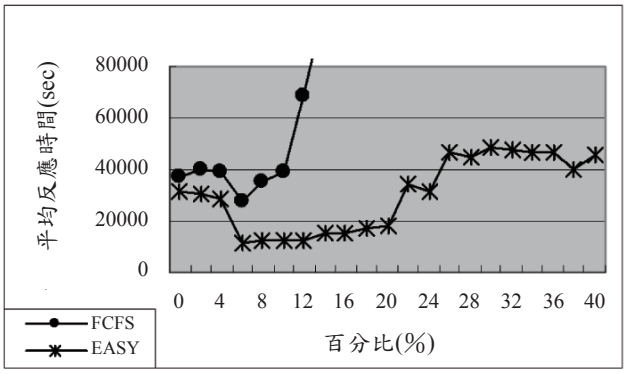
本研究採用平均反應時間（response time）和average bounded slowdown（公式(7)）作為我們評估排程方法的依據，bounded slowdown公式中當執行時間小於十秒時設為十秒，這個規定是防止因有些工作執行時間過短造成評估結果不準確。[1]

Bounded slowdown =

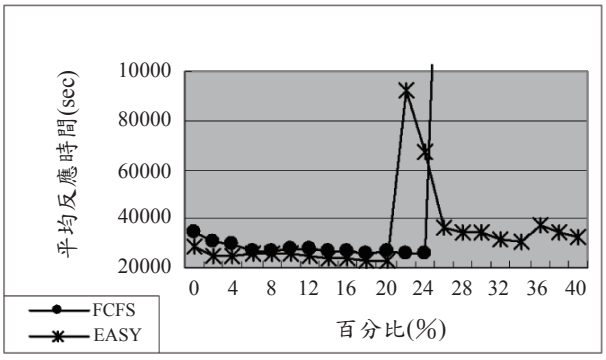
$$\frac{\text{wait time} + \text{Max}(\text{run time}, 10)}{\text{Max}(\text{run time}, 10)} \qquad (7)$$

5 實驗結果

首先我們將目標放在相同環境下不同排程方法間的差異性，我們的實驗環境為128個CPUs，工作數據採用DAS2-fs2，測試不同記憶體容量下EASY和 FCFS排程方法的優劣性。圖三為EASY與FCFS之間的比較，圖三(A)中在記憶體上限為0%的情況下EASY與FCFS處理同樣的工作數據，EASY所得到的平均反應時間比FCFS少了將近4000秒，EASY的排程效率比FCFS好上許多；當總記憶體容量為1.5G或2G時，工作等待系統資源的時間減少許多，因此平均反應時間跟1G比較之下有相當大的差異，在記憶體容量比較充足的情況下，EASY所表現出來的平均反應時間比FCFS好。

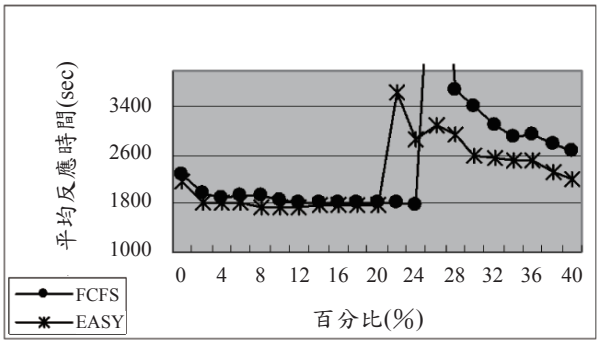


A



B

當排程方法加進有限度開放記憶體上限的考量後，圖三(A)在開放記憶體上限10%之內能夠有效的減少反應時間，反應時間上EASY表現出來的實驗結果比FCFS好；FCFS排程方法開放記憶體在26%到40%時，



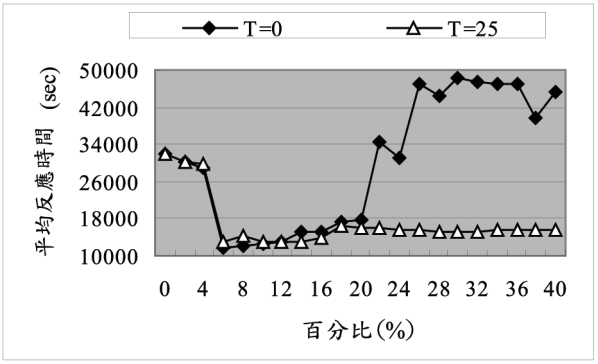
C

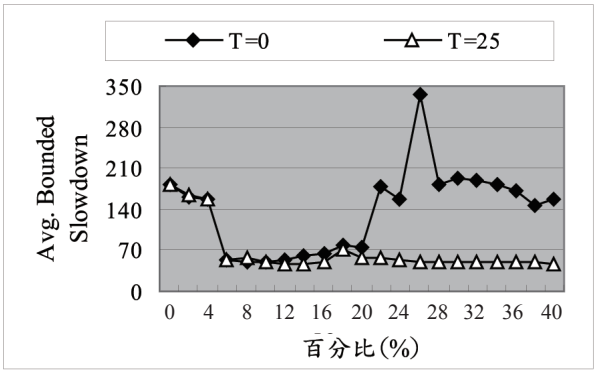
圖三　FCFS與EASY在相同環境不同記憶體容量上限的比較
(A)記憶體為1G (B)記憶體為1.5G (C)記憶體為2G

在1.5G或是2G的情況下平均反應時間有下降的趨勢，這是由於26%時，因為page faults增加，使得執行中工作的執行時間增長，影響到等待中工作的等待時間，然而開放到28%或更高的上限時，這些等待中工作有可能可以提前執行，因此等待時間降低造成平均反應時間呈現下降趨勢。

實驗結果得知開放記憶體在總記憶體上限10%以內能夠降低平均反應時間，而在10%以上減少的等待時間無法抵消增加的工作執行時間，甚至會出現平均等待時間反而延長的情況。因為EASY的表現普遍比FCFS好，因此之後的測試結果將著重於EASY排程方法的改良。

圖四是EASY排程方法加入等待時間考量後門檻值為0及25的結果，根據門檻值為25的實驗結果顯示，我們的新排程方法能夠有效並穩定地減少平均反應時間，因此我們會好奇在不同的門檻值和記憶體容量設定下，其平均反應時間的改變是否有不同。

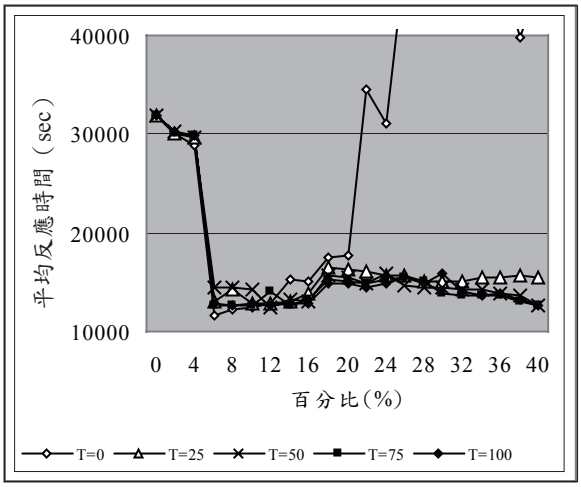




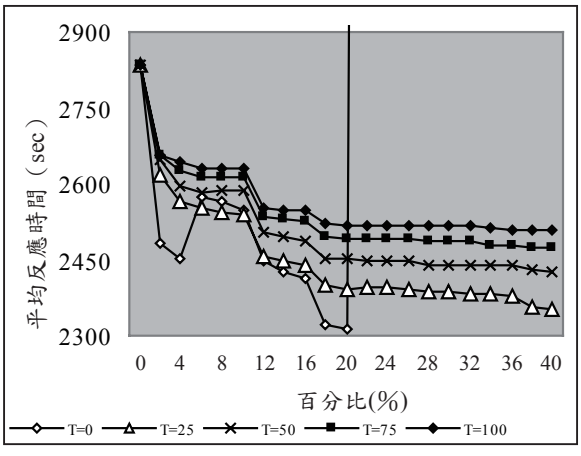
圖四　加入等待時間考量在不同門檻值的EASY排程方法
（CPU: 128, Memory: 1G）

圖五和圖六為門檻值在0（未使用門檻值的對照組）、25、50、75、100時所模擬出來的實驗結果，我們可以發現1G情況下門檻值為25以上時與門檻值為25時相差不遠。而在記憶體容量為1.5G以及2G時，記憶體資源不足的情況較為緩和，門檻值為25表現最佳，門檻值為0表現最不穩定，而門檻值較大時，雖然穩定，但是系統會因為等待時間較久而影響效能，實驗結果得知門檻值設為25所獲得的結果最好。

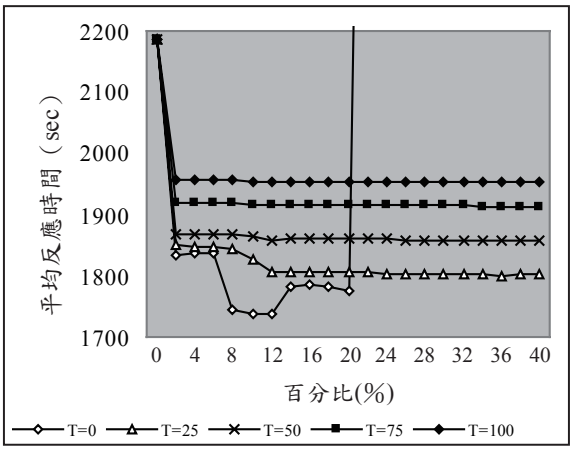
我們可以獲得一個結論，在記憶體資源嚴重不足的情況下，門檻值最好設在50以上，減低因為工作提前執行所增加的延誤等待時間，並且將受排程方法影響的工作數量控制在中等程度，而當記憶體資源大部份時間都足夠使用時，建議門檻值設為25左右，在這個門檻值之下，針對工作執行時間所做的等待時間考量，能夠充分利用而做到有助於反應時間改善。



A

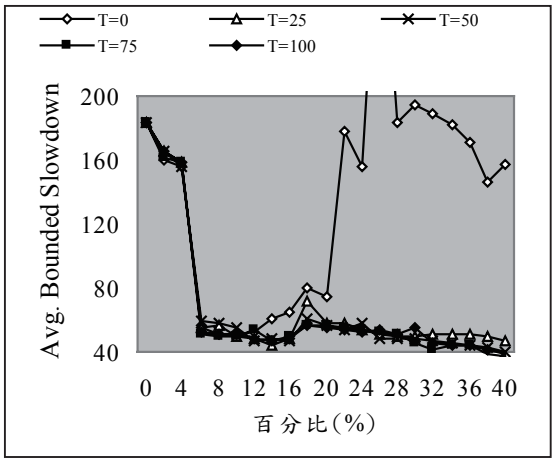


B

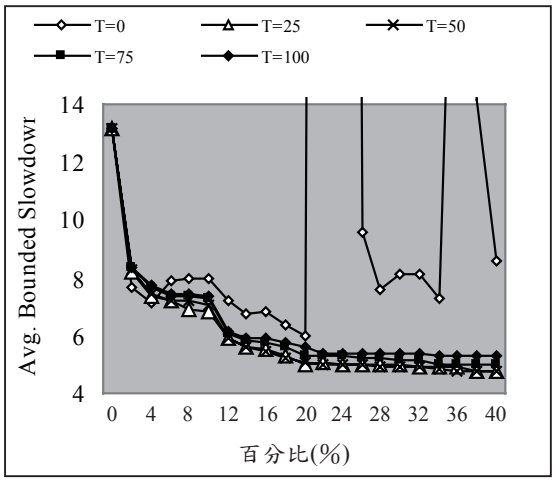


C

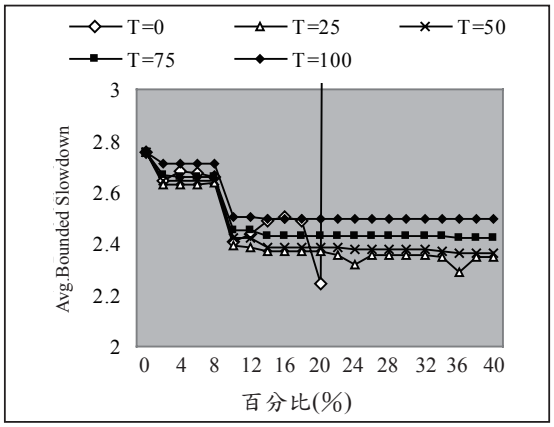
圖五　DAS2-fs2工作數據在不同系統記憶體容量下，不同門檻值的平均反應時間 (A)記憶體為1G (B)記憶體為1.5G (C)記憶體為2G



A



B



圖六 DAS2-fs2工作數據在不同系統記憶體容量下，不同門檻值的average bounded slowdown (A)記憶體為1G (B)記憶體為1.5G (C)記憶體為2G

6 結論

系統資源短缺會造成整體系統效能的嚴重下降，本研究針對前人學者所提出的space sharing方法作改良，對其加進記憶體考慮的新型排程方法，同時針對在系統資源短缺的情況下，有限度地開放記憶體上限將能增進排程方法的效能，我們使用模擬器模擬真實工作環境的結果，並且搭配不同硬體規格的環境以求得記憶體容量對工作排程的影響，經過實驗結果指出，我們的排程方法對於FCFS 能獲得最明顯的改善。針對EASY 排程方法，加進等待時間考量的排程方法可以獲得的效果不錯，在DAS2-fs2中，最佳情況可以減少許多反應時間，而加入等待時間門檻值的選定後，可以兼顧反應時間的減少以及穩定地降低反應時間。

6.1 未來發展

我們的研究工作還沒有結束，本研究並沒有去針對系統現在執行中工作的剩餘執行時間及這些工作佔有記憶體的情形做分析。剩餘時間短的工作或佔據資源多的工作是否盡可能讓他早些完成離開系統較有利？這些因素都是往後演算法改良必須考慮的因素。

致謝

本研究部份成果在TANET 2007發表，部份經費由國科會（編號NSC95-2221-E-126-005）和靜宜大學（編號PU96 11100 C25）提供，在此致謝。

感謝前後協助本研究之同學：李驕芸、陳岱宗。

參考文獻

[1] Dror G. Feitelson, Ahuva W. Mu'alem, "*Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling,*" IEEE Transactions on Parallel and Distributed Systems, Vol. 12, Issue 6, June 2001, pp.529-543.

[2] Dror G. Feitelson, Larry Rudolph, "*Parallel Job Scheduling -- A Status Report,*" 10th Workshop on Job Scheduling Strategies for Parallel Processing, June 2004, pp.1-16.

[3] Dror G. Feitelson, Anat Batat, "*Gang Scheduling with Memory Considerations,*" In Proceedings of 14th International Parallel and Distributed Processing Symposium, 2000, pp.109-114.

[4] D. Lifka, "*The ANL/IBM SP Scheduling System,*" 1st Workshop on Job Scheduling Strategies for Parallel Processing, April 1995, pp.295-303.

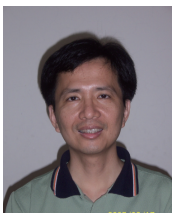
[5] J. Skovira, W. Chan, H. Zhou, and D. Lifka, "*The EASY-loadleveler API Project,*" 1st Workshop on Job Scheduling Strategies for Parallel Processing, April 1996, pp.41-47.

[6] Parallel Workload Archive, http://www.cs.huji.ac.il/labs/parallel/workload/l_das2/index.html

[7] The DAS2 5-Cluster Grid Logs, http://www.cs.vu.nl/das2

[8] Y. C. Tay, Min Zou, "*A Page Fault Equation for Modeling the Effect of Memory Size,*" Performance Evaluation, Vol. 63, Issue 2, 2006, pp.99-130.

作者簡歷



王逸民（Yi-Min Wang），目前是私立靜宜大學資訊管理系教授。主要研究領域為叢集計算、無線通訊和計算機結構等。

簡克達（Ko-Ta Chien），私立靜宜大學資訊管理研究所碩士。主要研究領域為叢集計算。

張沅合（Yuan-He Chang），私立靜宜大學資訊管理系畢業。主要研究領域為叢集計算。

田尚修（Shang-Hsin Tien），私立靜宜大學資訊管理系畢業。主要研究領域為叢集計算。