為 Teamster-G 開發一個工作流程管理系統
Developing A Workflow Management System for Teamster-G

梁廷宇
高雄應用科技大學電機系
lty@mail.ee.kuas.edu.tw

李偉丞
高雄應用科技大學電機系
cheng@hpds.ee.kuas.edu.tw

黃進安
高雄應用科技大學電機系
babyface@hpds.ee.kuas.edu.tw

摘要

Teamster-G 為本研究團隊在國科會經費贊助下所開發的一個計算網格系統。此系統可讓使用者以共享記憶體的介面，例如：Pthread、OpenMP 來開發網格應用程式，並且提供資源配置與動態重組的服務。然而目前 Teamster-G 並不支援工作流程（Workflow），這使得像 E-science、Grid ERP 等網格應用很難在此系統上被實現。為了解決這個問題，本研究為 Teamster-G 開發一個工作流程管理系統，讓使用者在 Teamster-G 上進行 Workflow 的編輯、提交、執行與監視。另一方面，我們為 Teamster-G 設計一個基於長時程資源可用度預測的資源配置方法。由於可以準確預知未來資源的可用度，因此 Workflow 整體的執行時間可以有效地被縮短。

關鍵字：Teamster-G、工作流程、長程資源可用度預測

Abstract

We have successfully developed an open source computational grid system called Teamster-G with the support of National Science Councils. This system enables users to develop parallel grid applications by means of the shared memory interface such as Pthread and OpenMP. Moreover, it provides users with transparent resource allocation and reconfiguration services to optimize the execution performance of user jobs. However, Teamster-G currently allows users to submit one job at one time. Therefore, it is difficult and inconvenient for users to develop workflow applications such as E-science and Grid ERP on Teamster-G. To address this problem, we develop a workflow management system for Teamster-G in this study. This system provides users with a set of web-services including workflow edition, submission, execution, and monitoring. Moreover, we proposed a new resource-selection algorithm based on long-term resource-availability prediction. Our experimental results have shown that the proposed algorithm is effective for reducing the makespans of workflows.

Keywords: Teamster-G、workflow、long-term resource availability prediction

一、 研究背景與動機

近來，網格計算技術[1]的成熟使得像 E-science[3] 和 Grid ERP[10] 等科學研究與商業應用得以在網際網路上實現。這使得人們可

藉由網格計算的機制，將分散在網路各處的計算資源、資料庫與科學儀器等加以整合，進而共同參與科學的實驗與分析。對跨國企業而言，亦可讓分散在世界各處的子公司很容易地協同處理公司的商業行程。由於這些應用都是屬於 Workflow 的模式，因此網格系統必須提供 Workflow 的管理服務，讓使用者能夠定義與提交想要執行的 Workflow，並追蹤其執行的狀態。

近來，本研究團隊（NCKU-、Leader、KUAS-HPDS Labs）在國科會的計畫經費補助下，已經成功地開發一個基於 OGSA 架構的計算網格系統，稱為 Teamster-G [6]。此系統可允許使用者利用共用記憶體的程式介面，如：Pthread 和 OpenMP[7] 去開發平行的網格服務與應用，同時也提供資源選擇、配置與重組的服務，讓使用者的程式可以獲得最佳的執行效能。然而，目前 Teamster-G 並不支援 Workflow 的工作模式，因此使用者仍然必須自己控制工作的分派順序。這對於 Workflow 應用的開發與執行而言，實在造成很大的不方便性。

有鑑於此，我們在本研究中特別為 Teamster-G 設計與實現一個 Workflow 的管理系統。透過此系統的服務，使用者可以在網頁上編輯與修改 Workflow、並提交 Workflow 給 Teamster-G 執行與即時地監視 Workflow 的執行狀態。在 Workflow 執行的過程中，工作流程管理系統會自動為 Workflow 的工作進行排程，並將工作提交給 Teamster-G 的 Broker 進行資源選擇與配置。Broker 會根據使用者設定的資源需求與未來資源可用度的預測進行資源的篩選，並選擇最佳的計算資源去執行使用者的工作。經過我們的效能評估，Workflow 的整體執行時間可以獲得明顯的改善。

本篇論文的其它章節組織如下：第二章相關研究；第三章 Workflow 管理系統架構；第四章、效能評估；第五章結論與未來工作。

二、 相關研究

在過去的研究中，被提出的 Workflow 管理系統所提供的使用介面與功能都很類似。例如：中國科學技術大學 Feng Zheng 與 Yang Shoubao[15]利用 Java web 元件 Portlet 來實現計算網格的入口網站，並且遵循 OGSA 提出的 Grid Service 標準。

中國復旦大學 Shaohua Zhang 與 Ning Gu[16]開發一個 Grid Workflow Portal。此 Portal 提供模組化與動態排程的方法。一旦資源效能下降，Resource Monitor 會馬上通知 Grid Workflow Engine 進行資源的重新配置。不過此系統無法讓使用者及時監視 Workflow 的執行狀態。

德國 C&C 實驗室 Junwei Cao 與英國華威大學 Stephen A. Jarvis 等人實現了一個名為 GridFlow 的入口網站[2]。此系統架構分成三個部份：User Portal、Global Grid Workflow Management 與 Local Grid Sub-workflow Scheduling。User Portal 提供使用者可以在系統上編輯工作流程；Workflow Management 則是負責執行與監控工作流程；Sub-workflow Scheduling 是負責避免資源衝突，為工作挑選可用的計算資源。

東海大學 Chao-Tung Yang、Cheng-Fang Lin 與 Sung-Yi Chen[14]在網格計算環境上加入 Job Monitor 與 Resource Monitor，使得 Workflow 管理系統可以提供更完善的功能，並且支援 Job Scheduling 的服務。

另一方面，對於 Workflow 的工作排程問題，也有許多方法已被提出來。例如：奧地利因斯布魯克大學 Marek Wieczorek 等人所提出的 Myopic[11]方法。此方法會為最先可以開始執行的工作進行資源配置，並選擇可以最快完成此工作的資源，來執行此工作。此排程方法的優點是簡單且容易實現。但是其 FCFS（First Come First Serve）的排程策略對於縮短 Workflow 之

Makespan 並不是非常有效。相對地，加拿大曼尼托巴大學 Muthucumaru Maheswaran 與美國普渡大學 Shoukat Ali 等人共同提出了 Min-min 與 Max-min 兩種排程方法 [4]。其中 Min-min 會評估每個工作配置至每個資源的執行時間，然後找到最早被執行完的一組工作與資源配對（Ti, Rj），並將工作 Ti 配置至 Rj 上，接著再對其它工作進行資源配置。至於 Max-min 則是將工作與資源依照計算量與計算能力排序，然後將有最大計算需求的工作配置至計算能力最強的資源上執行。基本上，如果某個工作的計算量明顯比其他工作來得大，則採用 Max-min 會比較好，因為 Workflow 的 makespan 會被最大計算量的工作之完成時間所決定。相反地，若是工作的計算量都很類似時，則採用 Min-min 會比較好。另外，澳洲墨爾本大學 Mustafizur Rahman、Strikumar Venugopal 與 Rajkumar Buyya 提出一個可有效縮短 Workflow Critical Path 之時間長度的排程方法稱為 DCP-G [5]。此方法會找出 Workflow 之 Critical Path，並且將 Critical Path 上面的工作配置於可讓其子工作之絕對最早開始時間（Absolute Earliest Start Time）最小化的計算資源上面執行。實驗證明，對於縮短 Workflow 整體時間而言，此方法比 Min-min 與 Max-min 更為有效。綜合而言，這些方法都為靜態的排程，意即 Workflow 尚未執行前就已經決定了每個工作的計算節點位置，而且都是只針對循序（Sequential）的工作，並沒有考慮平行（Parallel）工作。更重要的是這些方法在排程時並沒有考慮資源的動態性與未來的可用度變化，所以必須依賴後續動態的重新排程來彌補資源動態性所造成的效能損失。

相較下，本計論文所實現的 Workflow 管理系統提供了完整的 Workflow 管理服務，包括編輯、排程與監視等功能。另一方面，由於計算網格具有動態性，資源會隨時加入或者離開，並且可用度也會隨著資源本身的負載狀態而改變。為此我們的系統採用動態排程方法，而且對於不同性質的工作有不同的排程演算法。更重要的是我們的排程演算法中，有考慮計算資源未來的可用度，因此可有效降低資源重新配置的頻率與 Workflow 的整體執行時間。

三、 系統架構

在本研究中，我們使用 Liferay Enterprise Portal [13]來為 Teamster-G 開發一個 Workflow 管理系統。Liferay Portal 為一套專門為企業建置入口網站的開放原碼軟體，具有支援單一簽入 (single-sign on)、異質性、與 SOA 之特性，故符合 Teamster-G 的特性與要求。而我們所開發 Workflow 管理系統主要的組成單元包括 Workflow Editor 、Workflow Scheduler 以及 Workflow Monitor。其中 Workflow Editor 和 Workflow Monitor 是以 Portlet 作為使用介面。圖一為 Teamster-G 整合 Workflow 管理系統的架構。

基本上，Workflow Editor 的主要功能在於讓使用者透過其介面，編輯其所要執行的 Workflow。當使用者編輯完 Workflow 之後，便可將 Workflow 提交給 Workflow Scheduler 進行排程。Workflow Scheduler 會負責將 Workflow 的工作按照定義的先後順序，一一地提交給 Teamster-G 的 Broker，以進行資源配置並將工作分派至遠端的計算資源上執行。最後，Workflow Monitor 則是負責收集來自 Broker 以及遠端計算資源的回應，藉以即時地提供 Workflow 的工作執行狀態。

以下我們將針對 Workflow 管理系統的各個組成單元做一詳細的介紹。

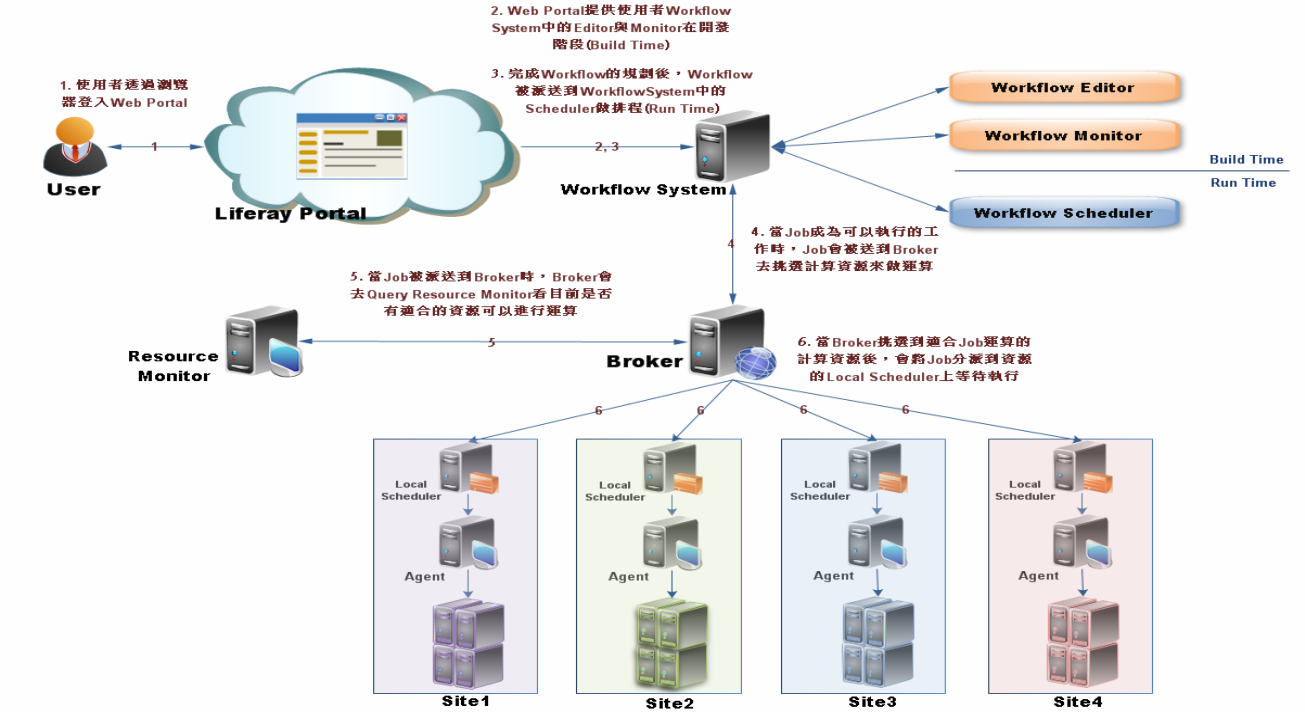
（一）Workflow 編輯器

Workflow 編輯器的介面如圖二所示。使用者必須對一個新增工作的屬性作定義，然後按

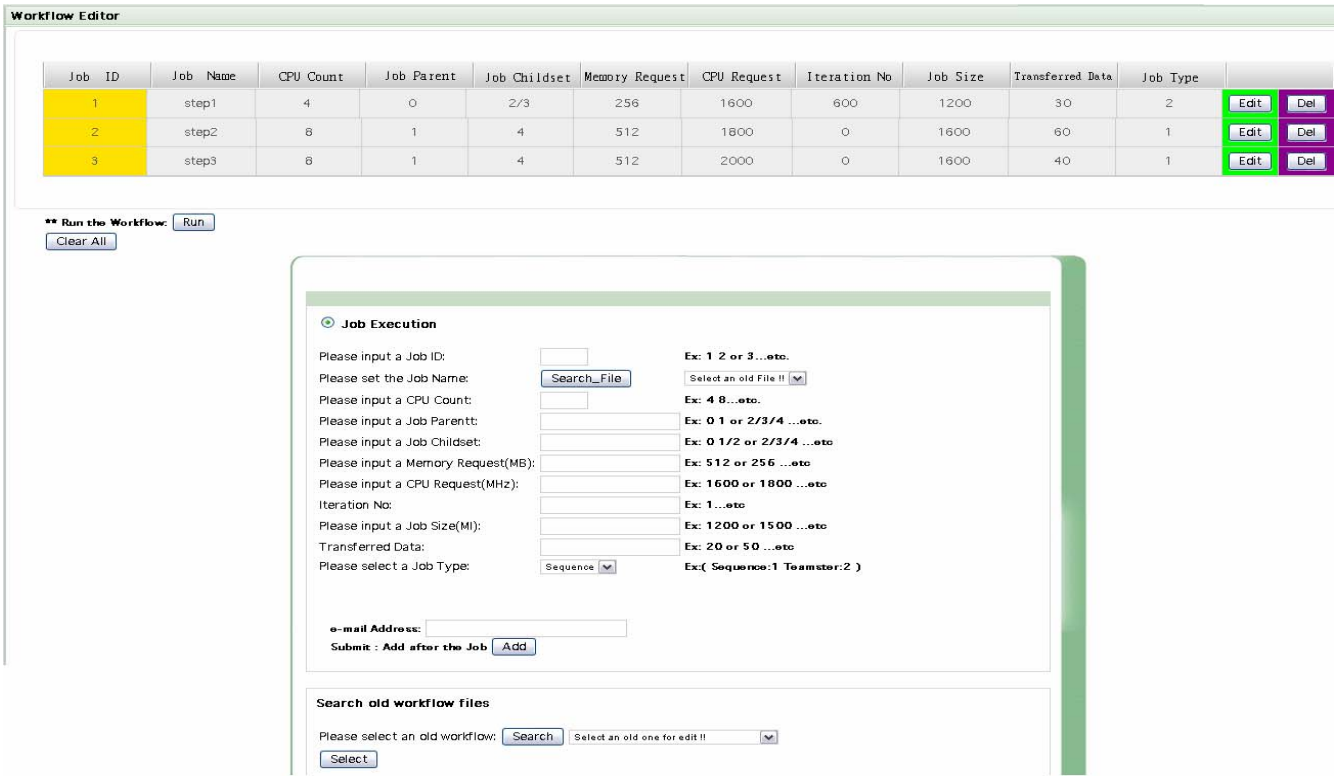
下 Add Button 將這個 Job 的屬性做儲存動作。系統會自動顯示出已被儲存的 Job 的資訊，並且讓使用者可以繼續新增 Job。當使用者想對某個 Job 做修改時，可按下 Job 資訊欄位中的 Edit Button 來對 Job 做修改。修改完畢後，可按下 Save Button 將 Job 的資訊更新。另一方面，使用者亦可將過去編輯過的 Workflow File 載入，再度做編輯與執行。要達到此目的，使用者必須在 Search old workflow files 欄位按下 Search Button 尋找使用者編輯過的 Workflow。選定 Workflow 後並按下 Select Button，系統會自動將被選擇的 Workflow 之所有 Job 資訊顯示於欄位中，供使用者再度編輯與執行。至於 Workflow Job 的每個屬性用途如下所述：

- Job ID: Job 的編號。
- Job name: Job 的執行檔案名稱。
- CPU Count: 執行 Job 時所需要的 CPU 個數。
- MI：此 Job 的運算量，單位為百萬個指令。
- Transferred Data（TD）:資料的傳輸量，單位 Mb。
- Iteration No: 工作執行回合數。
- Job Parent: Job 的 Parent set，可為零或數個
- Job Childset: Job 的 Child set，可為零或數個
- Memory request: 記憶體容量的需求，單位 MB。
- CPU request: CPU 能力需求，單位為 MHz。
- Job type: Job 的執行類別，若為平行程式則選擇 Teamster，否則為 Sequential 程式。

以上這些屬性會在控制工作分派順序以及工作資源配置時分別被 Workflow 排程器與 Teamster-G 的 Resource Broker 參考。



圖一、Teamster-G 之 Workflow 管理系統架構圖



圖二、Workflow 編輯器之 Portlet

(二) Workflow 排程器

為了提高排程的平行度，本 Workflow 管理系統排程器採用了多執行緒的架構。排程的演算法則類似 HEFT[9]，此演算法是目前最常見的 Workflow 動態排程演算法之一。基本上，一個工作若為 Workflow 的進入點（Entry point）或者其所有父工作（Parent Jobs）都已經被執行完畢，則此工作即可進入到 Ready Queue，等待被分派到 Teamster-G 的計算資源上執行。排程器會週期性地檢查 Ready Queue 是否有工作正等待被執行。當有多個 Ready 的工作時，排程的執行緒會為 Ready Queue 裡的每個工作計算權重(Weights)，然後將權重大的工作優先分派給 Teamster-G 的 Broker 進行資源配置與執行。其權重的計算公式如下所示。

$$W(T_i) = MI_i + \sum_{k \in S_i} MI_k \qquad (1)$$

上式中，S_i為工作T_i的所有後代工作的集合，MI_i則為T_i的計算量。

當排程器選定下一個要執行的工作後，即會將選定工作的屬性傳給 Teamster-G 的 Broker 進行資源選擇與配置。如前所述，我們在資源選擇配置必須瞭解未來資源可用度。為了達到這個目的，我們為 Teamster-G 開發了一個資源監視器（Resource Monitor），此監視器可支援長程（Long-term）的資源可用度預測[8]。經由實驗證明，這個監視器對於資源負載的預測可以提供比 NWS[12]的更高的準確度，而且預測的時間長度更長。簡單地說，Teamster-G 的資源監視器是將資源的負載波形轉為由 0~9 數字所組成的串鏈碼（chaincodes），然後再經由循序樣式探勘，以挖掘出潛藏在串鏈碼裡面的經常性負載樣式（frequent workload patterns）。每個負載樣式所記載的內容如下：

Nid: 計算資源的編號。
Start: 負載樣式起始時間，範圍為 0~23。
Length: 負載樣式的時間長度，單位:小時。
Hp: 負載樣式中第一個數字。

Support: 過去有出現過這個負載樣式的機率。
Pattern: 從 Start 開始之後的負載樣式。

當資源監視器要預測某個資源在未來 t 小時內的可用度時，會根據資源的目前狀態以及經常性負載樣式去預測未來 t 個小時內最有可能發生的負載樣式，然後根據此樣式去評估未來 t 小時內的可用度。例如：假設目前時間為早上 6 點，Nid0 資源目前的 CPU 負載狀態為 3(平均使用率 30%)，要預測其未來 2 小時內的資源可用度。其步驟將先查出（Nid=0, Start=6, Length=1, Hp=”3”, Pattern=”3”）的 Support 值（假設為 0.8）。接著，再找出（Nid=0, Start=6, Length=2, Hp=”3”）且 Support 值最高（假設為 0.4）的 Pattern(假設為”32”)。最後根據下列公式去計算其未來兩個小時內的 CPU 可用度(CPU-availability, CA)。

CA=confidence*(length-tload/9)*C_{max}*3600
tload=Pattern 的所有數字總和
C_{max}=最大的計算能力，單位MIPS。 (2)

假設前述的Nid0 資源的C_{max}為 2000MIPS，則其未來可用度計算如下：

CA=(0.4/0.8)*(2-5/9)*2000*3600=5200,000(MI)

其代表意義為此計算資源可以在未來兩個小時內，完成 5200,000 百萬個指令。若能知道工作所需的計算量，則可以評估出在每個計算資源上的執行時間，進而選擇出可在最短時間內完成此工作的計算資源。運用相同的原理，Teamster-G 的資源監視器亦可預測計算資源的未來可用網路頻寬可用度（Network bandwidth availability, NBA）與記憶體可用度（Memory availability, MA），並將預測的結果提供給 Broker 作為資源選擇配置的依據。

在資源配置的過程中，Broker 會持續地詢問 Resource Monitor 關於未來 t 小時內所有可用資源的可用度，直到資源選擇完畢。例如：對於 Sequential 工作而言，Broker 首先會詢問 Resource Monitor 未來 t（初始為 1）個小時內所有可用資源的未來可用度，然後根據工作的 CPU Request 和 Memory Request 兩個屬性先對可用的資源進行篩選，並且將符合條件的計算資源以 CA 值進行遞減排序。接著判斷排序第一的計算資源的 CA 是否大於工作的 MI 值。如果成立，則將工作配置到這個資源上執行。如果不成立，則繼續詢問 Resource Monitor 未來 t+1 小時內的資源可用度並重複前述的步驟，直到資源選出為止。值得一提的是目前 Resource Monitor 的資源可用度預測只支援到 24 個小時內。若是 t 等於 24 時，仍然找不到可將工作的計算量消化完畢的計算資源，將會選擇配置在未來 24 小時內可用度最高的計算資源給此工作。

相對地，Parallel 工作的資源配置必須考慮資料通訊的代價。首先，Broker 會對於工作的計算需求（CR：單位百萬指令）與通訊需求（TR：單位 MB）進行評估。其評估的公式如下所示：

$$CR = MI * CPU\ Count * Iteration$$
$$TR = TD * Iteration \qquad (3)$$

接著，Broker 將根據下面所述的步驟進行資源選擇。

1. 判斷工作所需的節點數是否小於可用的節點數。若是，繼續下一個步驟；否則 Reject 其資源配置的請求。
2. 若 t 大於 24 則跳到 Step 7；否則，則詢問 Resource Monitor 未來 t(初始為 1)小時內的資源可用度（包括 CPU 和網路頻寬）。
3. 將可用資源依其所屬的 site 分組，然後根據工作的 CPU Request 和 Memory Request 兩個屬性先對各 site 的資源進行篩選，並且將符合條件的計算資源以 CA 值進行遞減排序。
4. 進行資源組合，並計算各資源組合G_j的CA與NBA（即G_j裡所有節點之CA和NBA的總和）。
5. 判斷各組G_j的CA和NBA是否分別大於工作

的CR與TR。若滿足此條件，則G_j為候選的資源組合。

6. 若沒有任何候選資源組合，則將 t 值加一並回到 Step 2；否則，則選取 CA 最大的候選資源組合。

7. 判斷 t 是否小於 24，若是則將前一步驟所選擇資源組合配置給等待分派的工作利用；否則，則將 t=24 時，最大之 CA 的資源組合配置給工作利用。

由於Teamster-G可支援跨site的程式執行，所以在進行資源組合時，會考慮將不同site的資源組合在一起執行同一個平行工作。其組合的方式如下所述：假設目前有 3 個site，則其site的組合方式可為S1, S2, S3, S12, S13, S23 與S123。先將節點數小於工作所需的CPU數的site組合濾掉，然後分別對其餘的site組合的所有節點按照CA值作遞減排序，並取其前N個節點作為可用的資源組合G_j。

在資源配置成功後，Broker 會回應一個”Accepted”的訊息給 Workflow 的排程器以修改工作的執行狀態，並且將工作的執行檔以及輸入檔傳送至 Local Agents。Local Agents 收到來自 Broker 的工作派送後，即會將執行檔分散在選擇的計算節點上執行。當工作執行完畢後，Local Agents 會送”Finished” 的訊息給 Workflow 的排程器，以修改工作的執行狀態。

(三) Workflow 監視器

當使用者將 Workflow 提交給系統執行後，便可以透過 Workflow Monitor 的 Portlet 來監視 Workflow 的工作情況。基本上，每個工作的執行狀態可分為六種，分別為 Waiting、Pending、Accepted、Running、Finished 與 Failed。當一個工作正等待被分派到計算資源上做運算的時候，其狀態為“Waiting”。當 Workflow Scheduler 將工作分派到 Broker 去進行資源配置，其狀態改為”Pending”。一旦 Broker 回應資源配置成功，則狀態改為”Accepted”。此時代表 Job 已經被派送給計算資源的 Local Agent，並等待被執行。當計算資源開始執行 Job，則狀態會轉為”Running”。最後當 Job 執行完成時，計算資源上的 Local Agent 會回應 “已完成”訊息給 Workflow 排程器，此時工作狀態為”Finished”。若在執行過程中發生錯誤導致程式執行失敗，會回傳”執行失敗”訊息給 Workflow 排程器，此時工作狀態則改為”Failed”。使用者可藉由”Search”以及”Refresh”兩個按鈕，來選擇所要監視的 Workflow 與即時更新 Workflow 的執行狀態。另一方面，本 Workflow 管理伺服器會在 Workflow 執行完畢後，以 e-mail 通知使用者。使用者可以從 Workflow Monitor 看到每個工作的完成時間以及輸出檔內容。

四、 效能評估

在本研究中，我們針對 Workflow 的排程進行效能測試，藉以評估 Teamster-G 之資源配置方法的有效性。為了達到此目的，我們撰寫了一個模擬程式，對不同工作數目與類型的 Workflows 進行工作排程、資源配置與執行，並紀錄 Workflow 的起始與結束時間，以計算出每個 Workflow 的 Makespans。在實驗中，我們比照 R. Buyya 的論文[5]，總共製造出 fork-join、parallel 和 random 三種不同類型的工作流程。每個 Workflow 的 Job 數為 50、100、200 與 300，其工作量與計算節點需求如表一所示。至於 Workflow level 的最大寬度(即一層的 Job 數)則設為$W = \left\lceil \frac{N}{a} \right\rceil$，其中 N 為 Workflow 的工作數目，而 a 則設為 10。

另一方面，我們在模擬程式中，製造了九個 sites 共五百個計算節點，各個 site 的資源組態則如表二所示。為了製造計算資源的本身負載(local load)，我們在成大與高應大校園裡收集了一個月的 CPU 負載與網路流量波形，來當成是這些計算節點的 CPU 與 Bandwidth 的負載波形。在實驗前，我們先對計算節點的負載波形進行經常性負載樣式的探勘。當模擬程式開始執行後，其探勘結果將作為預測資源未來可用

度與資源配置的根據。

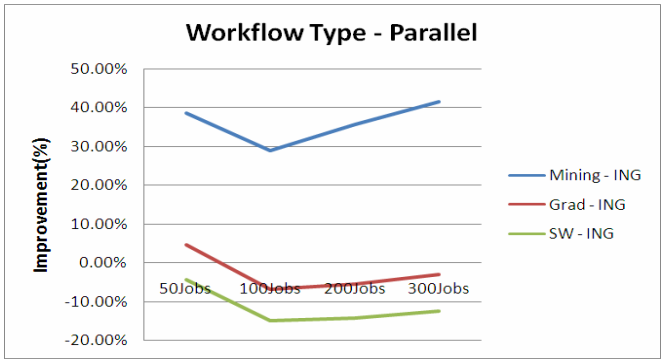
表一、Workflow 的工作需求

工作數.	50	100	200	300
每個工作所需的計算節點數目				
平均值	11.5	11.5	11.5	11.6
每個工作所需的計算需求（單位：百萬指令）				
平均值	95.2M	94.2M	102M	102M
每個工作所需的資料傳輸需求（單位：Mb）				
平均值	553K	524K	591K	588K

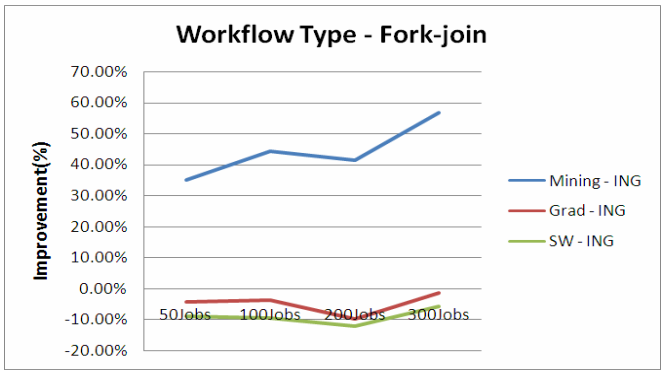
表二、計算節點的資源組態

Site ID	資源數量	計算能力 (MIPS)	通訊能力 (Mbps)
S1	50	1700	100
S2	60	1800	100
S3	60	1600	100
S4	50	1800	100
S5	55	1600	100
S6	65	1800	100
S7	60	1700	100
S8	50	1800	100
S9	50	1700	100

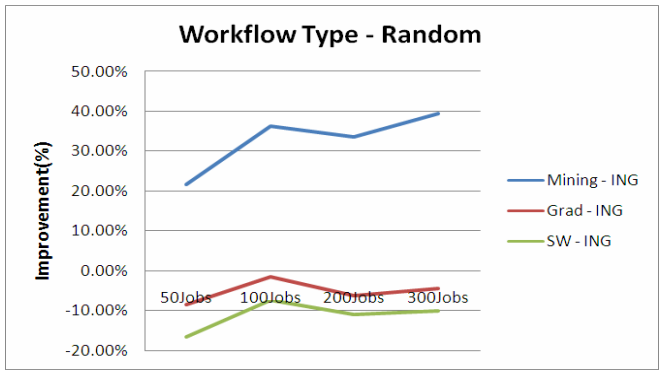
我們的實驗結果如圖三、四與五所示。除了本論文所提出的資源選擇配置方式（Mining）外，我們也應用只參考目前的負載狀態(ING)(即選擇目前負載最輕的)，以及兩種準確度較高的 NWS 預測資源負載方法: Average Slide Window (SW:視窗大小 K 設為 20)和 Gradient（Grad, 學習參數子 g 設為 0.05）於模擬程式的資源配置中，以作為資源配置效能的比較對象。從實驗結果可以看出，我們的資源選擇配置方法相較於 ING 方法，可以為 Workflow 縮短 20~50%的 Makespans，而且隨著工作數目的增加，其改善幅度亦有隨之增加的趨勢。另外，相較於 Parallel 和 Random 而言，對 Fork-join 的 Workflow 的效能改善明顯較大。其主要原因是 Fork-join 的 Workflow，其同一層的工作必須完全執行完畢才可以執行到下一層的工作，故其每一層的執行時間會被最慢完成的工作所決定，因此資源配置對於這種 Workflow 的效能影響會比另外兩種大。



圖三、對於 Parallel workflow 的效能改善



圖四、對於 Fork-join workflow 的效能改善



圖五、對於 Random 的效能改善

另一方面，實驗結果顯示根據 Grad 和 SW

的資源選擇配置方法並不能總是有效地改善 Workflow 的 Makespans，有時甚至還比 ING 的資源配置方式差。造成這樣的差距，是因為 Grad 和 SW 只能預測下一個 Step 的短時間資源負載狀態。然而從我們對於資源負載的實際觀察發現，大多數的計算資源的負載波形會呈現階梯式的方波，而且同樣的負載程度會持續一段時間，例如："333388888555577777222222…"。因此當工作被提交的時間點若是落在另一層負載 level 的起始點時，則 SW 和 Grad 將會來不及反應，進而造成對下一個 Step 的負載程度預測錯誤。相對地，ING 則會預測準確，因為接下來連續幾個 Step 的負載狀態都一樣。另一方面，若是工作被提交的時間點若是發生在一個負載 level 的結束點時，ING 則會繼續預測相同的負載程度，但因為下一個 Step 的負載已經改變，所以其預測結果是錯誤的。至於 SW 和 Grad 則會繼續預測錯誤，因為它們學習的速度仍然無法趕上突然的狀態變化。由此可知，為何基於 SW 和 Grad 之資源配置方法的效能會比基於 ING 的方法差。

相對於這些方法，我們的資源可用度預測方法能夠從資源過去的負載歷史中學習到"在什麼時間出現什麼樣的負載狀態時，下一個 Step 最有可能的負載狀態是什麼"，所以不管工作是什麼時候被提交，都可以準確地預測計算節點的未來負載狀態。更重要的是我們的方法可以預測未來 N 個 Steps 的長程資源負載樣式與可用度，所以能夠有效地為需要長時間執行的 Job 找出哪一組計算資源是最佳的，進而大幅地縮短 Workflow 的 Makespans。

五、 結論與未來工作

在本論文中，我們成功地利用 Liferay Portal 為 Teamster-G 計算網格開發了一個 Workflow 管理系統。在此系統的支援下，使用者可以透過 Portlet 來編輯與定義 Workflows 的工作屬性與資源需求，並且將它們提交給 Teamster-G 執行，以及即時監視 Workflow 的執行情形。這將讓 Teamster-G 的使用者可以很容易且有效率地開發屬於 Workflow 的網格應用。另一方面，為了有效地縮短 Workflows 的 Makespans，我們在工作排程中運用了一個可支援長程資源未來可用度的預測器。在此預測器的支援下，Temaster-G 的 Resource Broker 可以評估哪一個計算資源可以最早將工作執行完畢，並將此資源配置給此工作利用。在模擬實驗的結果中顯示，由於 Broker 能夠有效地為使用者的工作選擇較佳的計算資源，所以 Workflow 的 Makespans 明顯地被縮短約 20～50%。

對服務導向架構 (service-oriented architecture, SOA)的計算網格而言，如何維護系統的 QoS 是一個很重要議題。為了解決這個問題，過去的方法經常會採用預先資源保留的方法。然而，目前的資源保留方法都沒有將資源未來的可用度考慮進去，因而無法有效地維護工作的服務品質。為此我們將會設計一個基於未來可用度的資源預留方法，並將它實現於 Teamster-G 計算網格之中，以有效維護 Teamster-G 的工作服務品質。

六、 誌謝

在此感謝國科會工程處對於本研究計畫（NCS 96-2221-E-151-018-MY3）的經費贊助。

七、 參考文獻

[1] Mark Baker, Rajkumar Buyya, and Domenico Laforenza, "Grids and Grid Technologies for Wide-Area Distributed Computing", *International Journal of Software Practice and Experience (SPE)*, vol. 32, issue 15, pp.1437-1466, 2002.

[2] Junwei Cao, Stephen A. Jarvis, Subhash Saini, Graham R. Nudd, “Workflow Management for Grid Computing”, *Proc. of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 198-205, 2003.

[3] Addis M., Ferris J., Greenwood M., Li P., Marvin D., Oinn T. and Wipat A., “Experiences with e-Science workflow specification and enactment in bioinformatics” *Proceedings of UK e-Science All Hands Meeting*, pp. 459–466, 2003.

[4] M. Maheswaran, S. Ali, H.J.Siegel, D. Hensgen, and R. Freund, “Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems”, *8th Heterogeneous Computing Workshop (HCW’99)*, pp.30, 1999.

[5] S. V. M. Rahman and R. Buyya, “A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids”, *Proceedings of IEEE International Conference on e-Science and Grid Computing*, pp.35-42, 2007.

[6] Tyng-Yeu Liang, Chun-Yi Wu, Ce-Kuen Shieh, Jyh-Biau Chang, “A Grid-enabled Software Distributed Shared Memory System on Wide Area Network”, *Future Generation Computer Systems*, vol. 23, issue 4, pp.547-557, 2007.

[7] Tyng-Yeu Liang, Shig-Hsien Wang, Ce-Kuen Shieh, Ching-Ming Huang and Liang-I Chang, ”Design and Implementation of the OpenMP Programming Interface on Linux-based SMP Clusters”, *Journal of Information Science and Engineering*, vol. 22, pp.785-798, 2006.

[8] Tyng-Yeu Liang, I-Han Wu, Sheng-Yuan Chen, “A Long-term Resource Availability Predictor Using Frequent Workload Patterns”, *The 5th Workshop on Grid Technologies and Applications (WoGTA’08)*, pp.125-130, 2008.

[9] H. Topcuouglu, S. Hariri, and M.-Y. Wu.,” Performance Effective and Low-Complexity Task Scheduling for Heterogeneous Computing”, *IEEE Transactions on Parallel and Distribution Systems*, vol. 13, issue.3, pp260–274, 2002.

[10] Tsung-Li Wang, Chung-Ho Su, Pei-Yun Tsai, Tyng-Yeu Liang, Wen-Hsiung Wu, “Development of a GridERP Architecture: Integration of Grid Computing and Enterprise Resources Planning Application”, *Proc. of the 4th IEEE Conference on Wireless Communication, Networking and Mobile Computing*, pp.1-4, 2008.

[11] M. Wieczorek, R. Prodan, and T. Fahringer. “Scheduling of Scientific Workflows in the ASKALON Grid Environments”, *ACM SIGMOD Record*, vol. 34, issue 3, pp. 56-62, 2005.

[12] R. Wolski, “Dynamically Forecasting Network Performance Using the Network Weather Service,” *Journal of Cluster Computing*, vol.1, pp. 119-132, 1998.

[13] Jonas X. Yuan, “Liferay Portal Enterprise Intranets”, *Packt Publishing*, 2008.

[14] Chao-Tung Yang, Cheng-Fang Lin, and Sung-Yi Chen,“A Workflow-based Computational Resource Broker with Information Monitoring in Grids”, *Proc. of the 5th International Conference on Grid and Cooperative Computing (GCC 2006)*, pp.199-206, 2006.

[15] Feng Zheng, Yang Shoubao, “Implementation of Oriented-Service Grid Portal based on Portlet”, Computer Science Department, University of Science and Technology of China, 2004.

http://www.chinagrid.net/grid/paperppt/USTC/Oriented-Service_Grid_Portal.pdf

[16] Shaohua Zhang, Ning Gu, and Saihan Li, “Grid Workflow Based on Dynamic Modeling and Scheduling”, *Proc. of International Conference on Information Technology: Coding and Computing (ITCC2004)*, vol. 2, pp.35-9, 2004.