適用於電子設計自動化軟體之排程管理機制

張欽智
中華大學資訊工程學系
助理教授
changc@chu.edu.tw

郭宗泰
中華大學資訊工程學系
碩士班
e10202010@ chu.edu.tw

摘要

電子設計自動化軟體廣泛應用於 IC 設計公司，但由於高速運算的電腦及相關軟體費用昂貴。因此一般公司購買的軟體版權數有限，須在有限的高速運算的電腦運算，在使用時一般依工作的重要性分批使用，但如此的安排往往並無法將這些資源做最佳的應用。

本文中研究排程管理機制，開發資源整合系統，整合所有運算環境上的資源。透過資源整合系統能提供運算環境上的資源監控，也能透過後端的處理來整合電子設計自動化軟體授權資源，並提供自動化的工作排程。而為了能提高工作排程上的效率，結合優先權排程演算法及最短工作優先排程機制，來減少工作的等待時間及往返時間，進而提升整個運算環境的效能。

為了能夠驗證的排程效能，透過實際的運算環境來做實驗，經過實驗的數據比較，發現結合優先權排程演算法及最短工作優先排程機制的組合能有效節省平均等待時間。

關鍵詞：電子設計自動化、排程演算法、資源管理

Abstract

Electronic Design Automation (EDA) software is widely used in IC design houses. But high prices of high-performance computers and EDA software limited the number of computers and software. Most companies will allocate these resources based on the importance of the job. But this type of arrangement cannot use these resources in the best way.

In this paper we study scheduling mechanism and develop a Resource Integration System (RIS) to monitor all system resources, integrating the resources of EDA licenses through backend processing, and provide automated task scheduler. To enhance the efficiency of task scheduler, we combined the Priority Scheduling (PS) algorithm and the Shortest-Job-First (SJF) algorithm to reduce job waiting time and thus enhance the effectiveness of the entire computing environment.

To verify the efficiency of the task scheduling, we did experiments in real computational environment. From the collected data, the PS plus SJF algorithm can effectively reduce mean waiting time.

Keywords: Electronic Design Automation (EDA), Job scheduling, Resource management

1. 緒論

在 IC 設計的環境中，需要透過一些軟體及硬體的設備，才能夠組成一個運算的環境。近年來在硬體的設備上價格，隨著製造技術及競爭，更新的速度不斷的在加快，CPU 的核心數量變多，RAM 的規格也愈來愈大，但價格並未快速成長。反觀電子設計自動化工具(Electronic Design Automation，EDA)的支出相對成長，因此在 IC 設計環境中軟體費用往往高於硬體設備，所以為了節省 EDA 授權的可用套數，通常都會採購效能較佳的硬體設備來做運算，因為硬體效能愈高，EDA 工具使用時間就相對愈少，所以使用者能在使用授權次數也隨著增加，因此在理想的環境下來看，不管是硬體或是軟體，它們的使用率都被提高了。

但事實上環境還是會遇到一些軟硬體支援性的問題，如何有效運用這些資源排入 IC 設計的工作，能盡速回應不同的需求。這也是考驗著系統管理者的能力，如何用新的解決方案來做好管理的工作，如何用簡單的方試來排除複雜的問題，如何在系統管理層面能夠更精進，而做好這些管理工作，為的就是讓系統環境能夠運作的更順暢，並提高運算資源的使用率，讓企業能有更高的競爭力。

在尚未採用負載平衡系統以前，對於這些

軟硬體資源，如 EDA 授權、工作站及伺服器等等，實際上整體的資源使用率其實不高，會遇到使用者佔用同樣的運算伺服器，而引起運算伺服器的負載不是太高不然就是太低，這就是造成運算資源使用不均的問題。於這些資源無法有效的被利用，因為在這些研發的設備上投資金額上是相當高昂的，如果可以有效的整合這些運算資源，並縮減使用者的操作時間及資源的等待間，並且還能解決運算資源使用不平均的問題，這樣就能夠提升使用者的工作便利性，進而增加工作效率，對企業而言還能夠節省經費上的支出，並增加專案的進度，進而提升公司的競爭力。

為了整合運算環境上的資源，通常都會採用負載分擔(Load Sharing)系統的方式來做整合，它可以解決在運算資源上整合的問題，所以希望以 Open Grid Scheduler 為主，來開發一個資源整合系統(Resource Integration System，RIS)，透過 RIS 來提供 1.負載平衡機制，解決資源使用不平均的問題，2.新增使用者介面，縮減工作時間並解決操作不便的問題，3.提供資源監控功能，讓資源負載狀態能透明化，4.增加自動排程機制，解決等待資源時間上的問題，在排程機制上，也會再加入排程演算法，來減少工作的等待時間及往返時間，讓使用者能夠更快的完成他們的工作。

2.相關研究

2.1 電子設計自動化(Electronic Design Automation)

在早期是透過人工繪圖的方式來做電路的設計，隨著科技不斷的進步之下，現在自動化設計都已發展的很成熟，而透過電腦的輔助來設計出積體電路，稱之為電子設計自動化(Electronic Design Automation，EDA)，在 IC 設計的產業有很多的設計流程，都需要透過這些不同的軟體來協助設計，所以在不同 IC 設計流程下也會有不同的 EDA 工具，而 IC 設計流程從前段直到後段，會使用不同 EDA 工具來處理電路的模擬(Simulator)、分析(Analysis)、合成(Synthesis)、佈局(Layout)、繞線(Routing)及驗證(Verification)等等，而透過這些 EDA 工具，及專業的設計人員，可以加快開發流程，縮短 IC 設計時間，目前在 EDA 產業中常見的 EDA 工具公司有 Candence(益華科技)、Sysnopsys(新思科技)、Mentor Graphics (明導科技)等等。:

使用 EDA 工具需要有授權的授權才能使用，所以在 IC 設計環境之下，通常會架設授權伺服器來管理授權的套數，當開啟 EDA 工具時，EDA 工具會透過設定好的環境設定檔去找到伺服器，當找到伺服器後會根據所開啟的 EDA 工具去查詢有無可用授權套數，若有，伺服器會在總套數中減去使用的套數，當總套數被用完時，就無法再提供使用，需要等到授權歸還，才能再供下一個人用。

2.2 Open Grid Scheduler 環境介紹

網格運算(Grid Computing)的概念在 1990 年代就已經被提出，它是一種被應用在高速運算的一種分散式架構，可以視它為一台虛擬的超級電腦，它由一群異質性(Heterogeneity)電腦所組合而成[7][8][9][10][11] 。目前網格運算也被運用在 IC 設計的環境中，而且都是被運用在節點的部份[1][2][3][4][5][6]，在市面上已有許多相似系統，它們的優點包括提供平行處理、負載平衡、容錯性、彈性的排程、運算環境的整合及授權機制等等，功能相當完整，而業界中常見的系統有 IBM 公司所出的商業版 Load Sharing Facility (LSF)及 Oracle Grid Engine (OGE)，OGE 的前身是 Sun Grid Engine (SGE)，Open Grid Scheduler(OGS)是 SGE 一個免費版本[12]。

因為 LSF 是收費的商業軟體，它提供產品支援但收費高昂，對 IC 設計公司而言，又是另一項的成本的支出，相對 LSF 而言，可以看到部份的公司，選擇免費版 SGE 來導入到 IC 設計的環境中，除了可以節省成本的支出，又能享有 SGE 帶來的優點，例如將資源管理簡單化，讓存取效率增加，增加系統的彈性和擴充性，及最佳化的運算資源應用，當然 OGS 也是和 SGE 一樣有著相同的優點。

接下來介紹 OGS 的運作方式，首先來介紹環境中的角色，在圖 1 中每一個小方塊代表是一台主機(Host)，每一個主機所負責的角色如下。

- 客戶端(Client)：代表使用者已經登入到登入伺服器。
- 提交主機(Submit Host)：提供使用者來提交工作到主控主機。
- 主控主機(QMaster Host) ：是 OGS 的核心伺服器，它處理所有排程和負載平衡的工作。
- 隱藏主控主機(Shadow Master Host)：為次要主控主機的角色，是擔任主控主機的備

援。

- 執行主機(Execution Host)：負責處理來自主控主機分配的運算工作。
- 管理主機(Administration Host)：負責系統上的設定，也提供主控台介面來讓系統管理者使用。

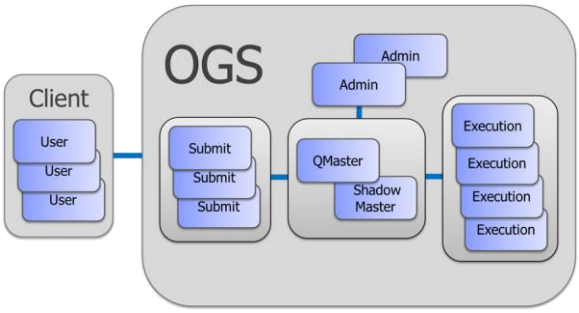


圖 1 OGS 運作環境

2.3 排程演算法(Scheduling Algorithms)

接下來介紹常見的一些排程演算法，目前常見的演算法如下[13]:

- 先到先服務 First Come First Serve (FCFS)
  FCFS 優點是演算方式簡單，容易被實作，運作原理是以先進先出(FIFO)的方試運作，缺點是若有大的工作一直佔用資源，小的工作會一直在等待，所以會發生護送效應(Convoy effect)。
- 最短工作優先 (Short Job First，SJF)
  SJF 優點是能讓工作有最短的等待和往返時間，但不容易被實作，運作原理是將排隊中的工作，選出最小的先做，最大的工作排到最後，若有相同大小的工作就以 FCFS 來排程，缺點是工作執行時間不容易被預測，而且當一直有小的工作被送進來，大的工作會一直處於排隊的狀態，因此會產生饑餓現象(Starvation)。
- 優先權排程(Priority Scheduling，PS)
  PS 優點是能讓最急迫的工作先使用資源，運作原理是先設定好工作的優先權，採用優先權高的工作為先，優先權低的為後，缺點是優先權高的工作一樣會產生饑餓現象(Starvation)。
- 輪循服務 (Round Robin，RR)
  RR 優點是具有公平的等待時間，原理是設定一個時間量(Time quantum)，不管工作大小，都只能在固定的時間內使用資源，若工作太大就排到下一次，若工作小在執行完後就不用在排，缺點是時間量若設太大就如同 FCFS，若設太小會一直發生內容轉換(Context Switch)，負擔會變重且效率上會變差。

由於 PS 需要人工定義優先權，且沒有特定的順序，所以不列入比較。SJF 不管是在平均等待時間或平均往返時間都是當中最優異的，接者是 RR，最後才是 FCFS。排程的研究是個典型的研究議題，相關原理也應用在雲端工作排程，最近的調查(Survey)研究可參閱[14] 。

3. 研究方法與系統架構

3.1 資源整合系統(Resource Integration System)

為了改善之前提到運算環境所遇到的問題，如資源使用不平均、使用操作不便、等待資源時間太長、負載狀況不夠透明化及資源環境無法有效做整合，所以在這個章節中的研究將建立一個資源整合系統，簡稱 RIS，來解決所遇到的問題，RIS 必需配合 OGS 的 Load Sensor 元件來同步運作，從圖 2 來看 Client 的使用者可以透過 RIS 系統，來使用已被整合過的運算資源，這些資源包含 EDA 授權及 Computing Pool 中的所有機器設備，使用者透過 RIS 就可以清楚知道，有那一些可用的資源或是授權，並可以利用它提供的功能來提交工作，而且系統管理員可以透過 RIS 系統來整合並管理整個運算環境的資源。

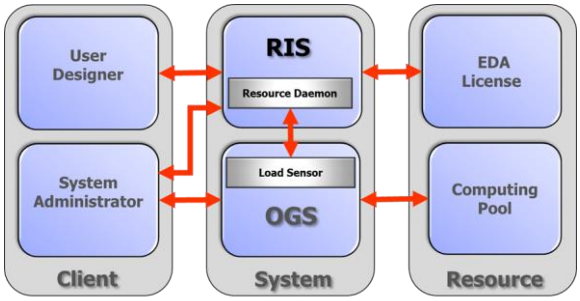


圖 2 資源整合系統

3.2 RIS 運作方式

RIS 的運作部份，如圖 3 所示，RIS 需要 OGS 上的 Load Sensor 元件來同步 RIS 系統上的資源常駐程式(Resource Daemon)，Resource Daemon 負責更新最新的負載狀況，並供資源監控(Resource Monitor) 來做查詢，也負責授權 Monitor，並將授權使用狀態更新至 OGS 中的資源屬性複合值(Complex Value)，OGS 可以從資源屬性複合值中得知授權使用狀況。資源屬性複合值是用來定義一些環境資源的屬性，還

可以根據不同環境來分配需要的資源，例如全域環境(Global)，主機(Host)環境，可被辨識資源序列(Queue)環境。Resource Daemon 也負責排程演算法(Algorithms)，它會將被提交的工作依照 PS+SJF 的演算方式去取得對應的權重值，然後再送入 OGS 中重新對工作做排序。使用者可以透過使用者介面(User Interface)來查看系統上的負載，及授權的使用狀況，它也提供工作提交的功能，讓使用者能更簡單的來提交工作。而系統管理員(administrator)也可以透過 Resource Monitor 介面來記錄負載的使用狀況，並依這些記錄來調整負載政策(Load Policy)，這樣負載平衡(Load Balancing)機制才能更完善。

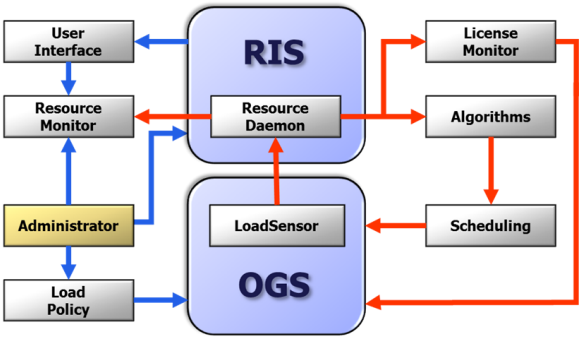


圖 3 RIS 運作方式

3.3 常駐程式(Resource Daemon)

資源常駐程式負責啟用授權監控、資源監控及演算法這些程式，並提供使用者介面，所需要顯示的資訊，而使用者介面除了顯示來自資源常駐程式提供的資訊之外，還需負責排程演算法的前置作業，如下所述。

1. 授權監控(Monitor)：

授權 Monitor 程式會先去讀取授權的清單，清單中包含所有授權伺服器的所使用固定的埠號碼(Port Number)及主機名稱(Host Name)，在查詢之前會去判斷授權 伺服器是否還在運作，若沒有，系統會發出郵件通知系統管理員處理，若授權 伺服器還在運作，會去取得授權資訊並比對 EDA 工具清單，並將要被使用到的 EDA 工具名稱顯示做好格式的處理，再利用 OGS 提供的工具將資訊寫入到 OGS 的複合值中，更新的時間是和 Load Sensor 元件同步的。

2. 資源監控(Resource Monitor)：

Resource Monitor 一開始會詢問使用者要列出那一些資訊，選擇 N 會提供所有主機的資訊，例如 CPU 使用率、RAM 使用率及核心數量等等，選擇 Y 會顯示已經定義好的訊息格式，而這些格式在日後可以任意增減，當選擇好選項後會將這一些資訊做格式化，最後再顯示給使用者，而更新的時間也是和 Load Sensor 元件同步。

3. 使用者介面(User Interface)：

User Interface 除了包含授權監控及資源監控，還包含工作提交、工作狀態查詢及工作的刪除。

- 工作提交:

在提交工作之前使用者必需先 Source EDA 工具的環境設定檔，接下來要去決定工作完成時間，再根據時間選擇工作代碼，代碼有分 SJF 的代碼及 PS 的工作代碼，但因為要取得高優先權的代碼，所以需要輸入密碼，平常使用者不會有密碼，只有專案較趕的時後才會提供，當選好代碼後再輸入執行的檔案名稱，接下來程式會重新將要執行的檔案名稱前面加上代碼，再提交工作到 OGS 中，等主控主機接收完畢後會回傳一組工作 ID，當訊息輸出給使用者之後就結束流程。

- 工作狀態查詢:

當使用者選擇此選項之後，程式會向 OGS 查詢工作的狀態，如果沒有任何工作在排程清單中，程式會顯示無工作 ID 的訊息給使用者，如果有工作就將所有的 ID 輸出給使用者，當顯示完畢後就結束這個流程。

- 工作刪除:

當使用者選擇此選項之後，程式會要求使用者輸入要刪除的工作 ID，程式會向 OGS 查詢此工作 ID，如果沒有任何工作在排程清單中，程式會顯示無工作 ID 的訊息給使用者，如果有工作 ID 就輸入該此工作 ID，當送出要刪除的工作 ID，程式會收到來自 OGS 的訊息，並顯示工作狀態，當顯示完畢後就結束這個流程。

4. 演算法(Algorithms)：

PS+SJF 的排程演算法負責工作的排程順序，一開始程式會定時的去偵測是否有未分配權重值的工作，如果有程式會再判斷工作狀態是否在等待的狀態(wq)，如果有就會取得工作名稱的第一個代碼，這個代碼是由使用者所指定的代碼，程式會去比對這個代碼，若是 p 開頭就給 SJF 的權重，如果是 h 開頭就給 PS 的權重，更改權重之後主控主機會更新工作的排列順序。排程演算法流程圖如圖 4 所示。

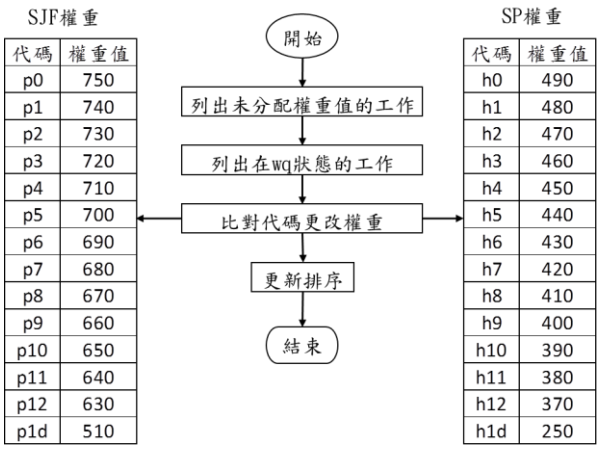


圖 4 PS+SJF 排程演算法流程圖

4. 系統實作

4.1 OGS 系統及授權環境的整合

在這一個章節，將介紹 OGS 的安裝及環境的基本設定，並將 EDA 授權動態寫入至 OGS，好處是不需要 OGS 上做 EDA 授權套數上的調整，不論 EDA 授權套數在未來有任何變動，OGS 上也會隨著 EDA 授權現有的套數來做改變，將依照以下的步驟來完成。

1. 在建置的前置作業中，列出一些安裝資訊表，去定義環境中每一台主機的角色，之後將依這張表做參考設定。
2. 在網址 http://sourceforge.net/projects/gridscheduler/files/下載 Grid Engine ，並建立一個 sgeadmin 的帳號來做管理，如圖 5 所示。

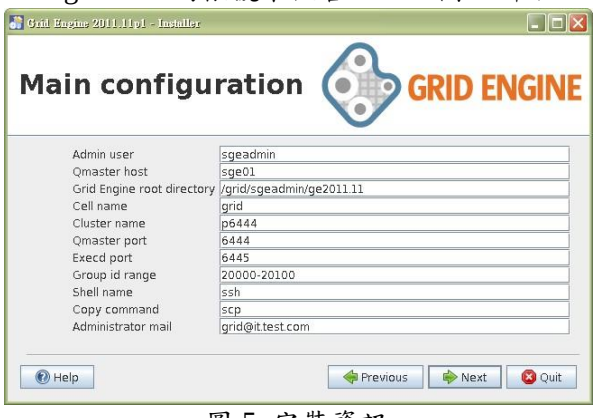


圖 5 安裝資訊

3. 安裝完畢後，登入到每一台主機當中，根據每個角色的不同去啟用對應到的 Daemon，這些 Daemon 是跟著角色名稱而執行他們的工作。
4. 開啟 QMON 的主控台介面，如圖 6 所示。

啟用後選”Host”，會另外開啟”Host Configuration”的設定介面。而它的功能是用來定義主機的角色，所有列出的主機必須被定義，定義後 OGS 就能根據這些定義出來的角色來運作。

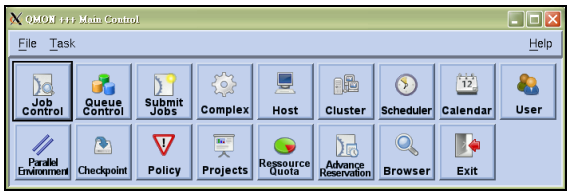


圖 6 QMON 主要控制台介面

5. 在 Administrator Host 中指定的主機可以變更所有 OGS 中的設定，在其它主機中，並無權限可以做任何設定上的變更，而 Administrator Host 中不需要啟動 Daemon 就可以來做使用。
6. 接下來為了防止主控主機因為其他因素的故障，而造成 OGS 系統停擺，所以必需建立系統的備援機制，在預設中會在主控主機上去啟用 sge_qmaster 以及 sge_shadowd，而隱藏主機只啟用 sge_shadowd。
7. 已經完成一些基本的設定，接下來用 sgeadmin 這個帳號來測試工作的提交。
8. 在 QMON 這個主控台，如圖 7 所示，選擇 Complex，Complex Configuration 這個圖形介面提供一些選項來設定資源屬性複合值，透這個設定可以讓 OGS 知道目前有多少授權的資源可以被使用，實驗主要以常用的 2 套 EDA 工具來做研究，這 2 套分 EDA 工具別是 Verilog 及 Hspice，將這 2 套之後會用到的授權加到 Complex Configuration 之中，並且需要定義 hspice 及 VERILOG-XL 這二個 ETA 工具為 INT 的屬性，而且是可以被請求及消耗。

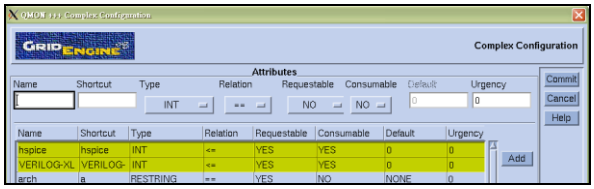


圖 7 Complex Configuration

再來需要透過程式來取得授權的內容，並將授權的資訊回傳至 OGS 中，這樣 OGS 才能讀取最新的授權套數，先將需要的 2 個授權，將他們加入到 complex_valuse 中，使用指令 qconf -me global 來編輯內容，並且加入“complex_values hspice=0”，VERILOG-XL=0”到檔案中，這個值的內容先設定為 0，之後會透過程式來更新它。

4.2 調整負載平衡政策

因每家 IC 設計公司環境都不同，所以負載平衡政策也不同，如網路架構、運算設備和硬體數量等等，這都需要配合他們的資源來做負載平衡的調整，而在整合授權的資源後，也已設定好負載平衡的政策，舉 Slots 的設定為例，如圖 4.10 所示，OGS 的 Slots 決定一台 Computing 可以被提交多少工作，在這裡設定有多少核心就可以提交多少工作，找出 3 台有 24 個核心的運算伺服器來測試，分別為 sge04、sge05 及 sge06，接下來模擬 sge04 允許有 2 個 Slots，sge05 允許有 3 個 Slots，而 sge06 可以有 4 個 Slots，再來準備 10 個工作來提交，同時提交 10 個 sleeper.sh 的工作後可看到前 9 個工作已在執行狀態，而第 10 個工作因為已達 Slots 上限，所以暫時進入等待的狀態。而其它的資源可以透過相同方式來更改負載平衡政策，所有的資源都是可以被彈性做調整的。

4.3 資源監控及使用者介面

資源監控系統會提供所有主機的使用狀況，在管理的部份可以監控所有主機負載狀況，並且會定時收集負載記錄，作為調整資源上的參考數據，透過分析也可知道，是否需購買新的設備等等。另外也提供使用者能查看負載使用狀況，例如使用者可以參考核心及 RAM 數量，來決定有那些機台可以執行多核心，RAM 的數量是否足夠新專案的進行，作業系統(OS)的版本是否可以執行最新的 EDA 工具等等。

接著利用 Perl 完成資源監控的程式，透過指令"ris －h" 可以查看主機的負載狀況，在，如圖 4.11 所示可以看到的所有主機資訊。透過其他的選項，例如在"ris －l"，可列出主機的登入相關資訊，讓使用者能知道有那些批次工作(Batch Job)的主機是不能登入的，而非批次工作的主機是可以登入的，且可用來執行非批次工作的 EDA 工具，最後也可以列出機器的類型(Type)，讓使用者知道他能透過那些機器來提交工作。

接下來利用 Perl 來完成整個使用者介面，如圖 8 所示。RIS 使用者介面會提供 1.授權的使用狀況，2. 顯示所有機器的負載，3. 提交工作的相關功能，依序來介紹每一個選項的功能。

```
{sge07}/datool/sgeadmin/script>./ris.pl

<Resource Integration System V2.1>
Number Command Funtion
1)     ris -l  Show license status
2)     ris -h  Show Host resource
3)     ris -q  Run queue funtion
q)     Exit
Enter number to continue :
```

圖 8 RIS 使用者介面

1. 授權的使用狀況:

從之前的 OGS 設定中，就已經將授權的環境整合到系統內，並將所有的授權使用套數寫入至 complex_valuse 之中，當使用指令 ris －l，程式會去讀取授權 Monitor 執行後產生的值，透過程式整理格式化之後再顯示給使用者，顯示的內容有授權的總套數、目前已使用的套數及可用的套數。

2. 顯示所有機器的負載

在 OGS 的環境中，所有的主機都會去執行 sge_execd 的 Daemon，因為 sge_execd 會將所有主機的負載傳送給主控主機，所以會利用 ris －h 的指令，透過程式去主控主機中讀取所有主機的系統資訊和目前的負載狀態，另外也提供在指令上加上－l 選項，經由這選擇可以列出登入資訊，因為這些資訊很透明，所以使用者可以完全的掌握目前所有主機的使用狀況。

3. 提交工作的相關功能

當使用者將他要執行的工作準備好後，可以執行指令 ris －q，如圖 9 所示。選項 0 和選項 1 的流程相同，選項 0 因為是高優先權所以要有密碼限制，可以防止使用者搶用，利用選項 1 做說明，當使用者進入選項 1 之後，系統會要求輸入工作的執行時間，例如使用者預估工作時間是在 30 分內，他選擇對應的代碼是 p0(若是高優先權則代碼是 h0)，接下來使用者輸入工作名稱後，系統會重新將工作名稱改為"p0.工作名稱"，如果 OGS 接收到"p0.工作名稱"，OGS 會給予 1 個工作 ID，有了這個 ID 使用者可以更改工作狀態和刪除工作，而使用者也需透過這個 ID 才能知道工作目前的使用狀況。

在提交工作後可以透過選項 2，來顯示工作狀態。可以看到剛才的工作 ID 及工作的名稱，而根據工作 ID 可以看到目前對應到的"status"是在 qw 的狀態，而且尚未被送到"queue"中的主機執行。

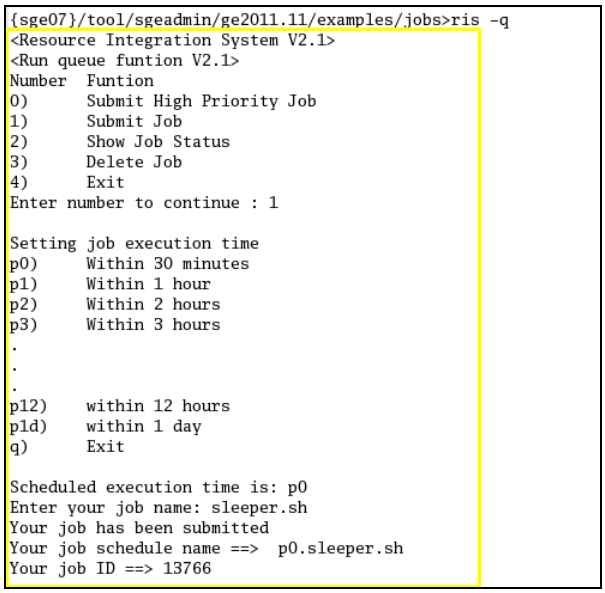


圖 9 Submit Job

如果使用者想刪除他們提交的工作，他可以透過選項 3 來刪除工作，使用者只要輸入工作 ID 後就可以刪除他的工作。

5. 系統實驗與分析

在這個章節中將針對排程演算法來做比較，為了減少實驗數據的差異，在軟硬體的設備上，全都選用相同規格的運算伺服器，如表 1 所示，讓實驗數據的影響因素可以減至最少，而設定實驗環境，如表 2 所示，單機單工作環境選用的 EDA 工具為 Verilog，而運算伺服器只用一台，並連續提交 42 個工作在同一台運算伺服器上，為了模擬 CPU 滿載，所以在 OGS 上將運算伺服器的 Slots 數由原本 24 改為 1，指的是只有 1 個核心可用，分別在每一個演算法中執行 10 次。在多機多工作環境中 EDA 工具改為 Hspice，運算伺服器從一台新增到三台，並連續提交 42 個工作到每一台運算伺服器中，而 Slots 數由原本 24 改為 2，原因是 Hspice 的工作是設定 2 個核心來同時執行，在每一個演算法中，都會被執行 10 次。

表 1 硬體規格

	Server 規格
機型	Acer R360 F2 1U Server
OS	CentOS release 5.8
CPU:	Intel(R) Xeon(R) E5-2620 @ 2.00GHz X 2
RAM:	256GB

表 2 測試環境

測試環境	EDA 工具名稱	伺服數	Slots 數	演算法	執行次數	工作數
多機多工作	HSPICE	3	2	SJF	10	42
				PS + SJF		
				FCFS		

5.1 測試設定

在實驗進行之前，先準備 14 個不同代碼來代表工作執行時間，如表 3 所示，針對這些工作在實驗之前已經在 2 個環境的 EDA 工具上先執行過一次，並將時間調整至設定的時間內，例如先執行一次 Verilog 或 Hspice，如果執行數據的時間是落在 1 個小時以上 2 個小時以內設定這個工作為 p2，同樣的若時間落在 12 個小時以上 1 天以內設定這個工作為 p1d。在實驗中這些代碼在 SCSF 只是用來做順序記錄，不影響排程順序。在 SJF 的排程中，p0 代表是最短的執行工作，p1d 代表是最長的工作，SJF 是依照這個順序來排序。在 PS+SJF 的排程中，由於會有高優先權的工作，所以除了會有 p 的代碼外還定義 h 代碼，如表 4 所示，而 h 代碼的規則和 p 代碼相同，但被排程的順序會高於 p 代碼。以上這些規則在多機多工作的環境，也是以相同的方式來做實驗。

表 3 工作時間定義

p0	p1	p2	p3
半小時內	1 小時內	2 小時內	3 小時內
p4	p5	p6	
4 小時內	5 小時內	6 小時內	

p7	p8	p9	p10
7 小時內	8 小時內	9 小時內	10 小時內
p11	p12	p1d	
11 小時內	12 小時內	一天內	

表 4 高優先權工作時間定義

h0	h1	h2	h3
半小時內	1 小時內	2 小時內	3 小時內
h4	h5	h6	
4 小時內	5 小時內	6 小時內	

h7	h8	h9	h10
7 小時內	8 小時內	9 小時內	10 小時內
h11	h12	h1d	
11 小時內	12 小時內	一天內	

5.2 測試環境 – 多機多工作

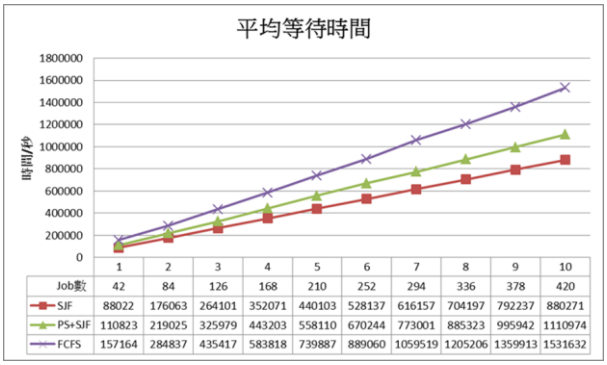
在多機多工作的測試環境中，在測試環境中，將利用表 3 中定義的 14 個工作數，產生 3 倍的數量，共有 42 個工作數來提交，而執行的次數設定為 10 次，在這 10 次當中透過隨機亂數的方試，這 42 個 Job 在每一次的執行中，只會出現過一次，利用同一台運算伺服器，並模擬將 CPU 為滿載，只留一個核心來做排程，因為在高負載的情況下如果可以正常排程，那在平日負載低的時後，更不會有無法排程的狀況發生，以下是個別實驗三種不同排程演算法的執行結果。

1. FCFS 是 OGS 預設的排程演算法，依照(附錄 1)的順序來執行，透過 FCFS 排程之後，原本的執行順序並不會被改變，在執行之後將排程的執行結果，共有 10 次的數據。
2. SJF 排程演算法中，工作也依照(附錄 1)的順序來執行，經過排程的處理之後，可以看到工作依最短到最長的順序來排程，若有遇到相同的代碼並不會被重新排序，而是採用先到先執行(FCFS)的方式來處理，最後將執行的 10 次數據，全部都記錄到(附錄 3)。
3. PS+SJF 排程演算法中，所有工作也是依照順序來執行，而執行之前會以隨機亂數的方試選出 10 個工作來當高優先權，並以 h 代碼來表示，經由 PS+SJF 的排程之後，可以看到原本工作的順序都被改變，所有 h 代碼的工作都已先被 PS 的排程優先處理，再經由 SJF 的排程將 h 代碼的工作重新排序，使最短工作能先做，而 p 代碼的工作也是經由 SJF 排程而改變原本的執行順序，排程後也是依最短到最長的順序被執行，最後再將執行 10 次的數據，全部都記錄。

EDA 工具採用 Hspice，而執行的運算伺服器為 3 台，模擬 CPU 在高負載的狀態，而 CUP 保留 2 個核心來處理，並透過 RIS 的負載平衡機制，將工作提交到每一台伺服器中。

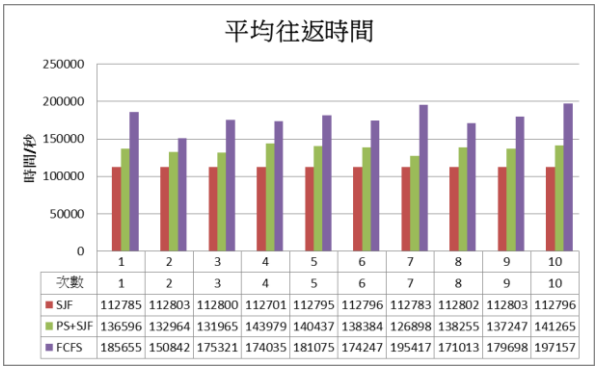
最後整理數據，並算出平均等待時間及平均往返時間後，接下來就如同單機單工作的測驗方式一樣，透過折線圖來分析，實驗的結果一樣是 SJF 及 PS+SJF 的數據優於 FCFS。

再來分析測試環境，經過比較 SJF 在每 10 次中的平均等待時間都比 FCFS 提升 41%，而 PS+SJF 提升 26%，在平均往返時間 SJF 比 FCFS 提升 36%，而 PS+SJF 提升 22%。



	1	2	3	4	5	6	7	8	9	10
Job數	42	84	126	168	210	252	294	336	378	420
SJF	88022	176063	264101	352071	440103	528137	616157	704197	792237	880271
PS+SJF	110823	219025	325979	443203	558110	670244	773001	885323	995942	1110974
FCFS	157164	284837	435417	583818	739887	889060	1059519	1205206	1359913	1531632

圖 10 10 次工作數的平均等待時間



	1	2	3	4	5	6	7	8	9	10
次數	1	2	3	4	5	6	7	8	9	10
SJF	112785	112803	112800	112701	112795	112796	112783	112802	112803	112796
PS+SJF	136596	132964	131965	143979	140437	138384	126898	138255	137247	141265
FCFS	185655	150842	175321	174035	181075	174247	195417	171013	179698	197157

圖 11 10 次工作數的平均往返時間

6. 結論

從企業實際的角度來分析實驗數據，如果說平日沒有緊急的專案情況下，也就是沒有工作高優先權的要求之下，PS+SJF 排程演算法就等同是 SJF 排程演算，能發揮 SJF 帶來的效益。若遇到專案很緊急的時候，PS+SJF 雖然會犧牲一些等待及往返時間，但它卻有高優先權的彈性，也能夠防止饑餓現像發生，實際上我們在 IC 設計環境當中，都會遇到專案很趕的時候，但相較之下，專案能及時完成的重要性，會比遠比 PS+SJF 犧牲掉一些等待及往返時間來的重要許多。

現在加入 PS+SJF 的排程演算法後，經過實驗數據的比較，在平均等待時間上比 FCFS 提升 26 ~ 41%，而在平均往返時間比 FCFS 提升 22 ~ 36%，所以 PS+SJF 的排程演算法，確實能減少工作的等待時間及往返時間。

在 RIS 未完成之前，運算環境並沒有排程系統，所以使用者只能自己提交工作，但透過

PS+SJF 的排程演算法，不僅可以全自動排程，且可以減少工作的等待時間及往返時間，讓使用者可以使用更少的時間讓工作跑完，而負載平衡政策更可以讓系統資源充份的被利用，使用者介面可以讓使用者操作可以更便利，所以 RIS 在資源環境的整合上，已經為帶來一定的效益。

透過 PS+SJF 的排程演算法，已經可減少工作的等待時間及往返時間，但在研究過程中發現還有其它方法可以再增加授權及運算伺服器的使用率，並且可以再進一步的縮減工作的等待時間及往返時間，以下還有一些方向，可以再做未來的改進與研究。

實驗過程中發現二個特性，第一觀察到每天的授權及運算伺服器使用率，在夜晚會比白天來的低，第二發現部份的 EDA 工具已經有支援多核心，當核心數愈多工作的加速比也會比較好，例如 Hspice 就有支援多核心，而綜合這 2 點可以做一些調整來做改善，例如當在夜晚的時候系統偵測到資源使用率偏低時，假如使用者是使用雙核心來執行，而當下的授權可用套數是充足的，且運算伺服器的負載也不高，這時候可以透過排程系統，將使用者提交的 Hspice 工作內容做更改，可以將使用者定義的 CPU 參數由 2 增為 4 或 6 核心以上，這樣不僅可以提升授權及運算伺服器的使用率還能提高工作的加速比，降低工作的等待及往返時間。

參考文獻

[1] 李中原，「以網格運算架構做 IC 設計運算資源分配」，國立交通大學，資訊管理學程碩士論文，2005。

[2] 曾金豐，「以模糊理論建構一個高效能之計算環境-以 IC 設計公司為例」，中華科技大學，資訊管理學系碩士班，2007

[3] 張智鈞，「適應於 QoS 與網格經濟模型之服務導向排程技術」，中華科技大學，資訊管理學系碩士班，2007

[4] 許滿滿，「運用網格運算的分散式資源管理於 IC 設計環境」，國立交通大學，管理學院碩士在職專班資訊管理組，2008。

[5] 陳慶豪，「網格節點工作排程分析之研究」，中國文化大學，資訊管理研究所，2009

[6] 羅道蹟，「網格效能最佳化之網格執行節點的工作排程分析」，中國文化大學，資訊管理學系 2011。

[7] Marcus Bisogn, Grid Computing: Distributed Computing Architecture, CreateSpace Independent Publishing Platform, 2016.

[8] Erik Putrycz, “Design and implementation of a portable and adaptable load balancing framework, 2003 conference of the Centre for Advanced Studies on Collaborative research, October 2003.

[9] Aravind Menon, Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman, Willy Zwaenepoel, “Diagnosing performance overheads in the Xen vtual machine environment”, 1st ACM/USENIX international conference on Virtual execution environments, June 2005.

[10] Kevin J. Barker, Nikos P. Chrisochoides, “An Evaluation of a Framework for the Dynamic Load Balancing of Highly Adaptive and Irregular Parallel Applications, Proceedings of the 2003 ACM/IEEE conference on Supercomputing, November 2003.

[11] Weiguang Shi, M. H. MacGregor, Pawel Gburzynski, “Load balancing for parallel forwarding”, Transactions on Networking (TON) , Volume 13 Issue 4, August 2005.

[12] Open Grid Scheduler, http://gridscheduler.sourceforge.net/

[13] Avi Silberschatz, Peter Baer Galvin, and Greg Gagne, Operating System Concepts, 9th Ed., Wiley, 2012.

[14] Luiz F. Bittencourt, Edmundo R. M. Madeira, and Nelson L. S. da Fonseca, Scheduling in hybrid clouds, IEEE Communications Magazine, vol. 50, no. 9, Sep. 2012, pp. 42 – 47.