Linux 叢集電腦程序群組即時監控系統之實現

林作俊
宜蘭大學電子工程學系
cclin@niu.edu.tw

許致軒
宜蘭大學電子工程學系
r9642002@niu.edu.tw

摘要

在叢集電腦中執行平行程式，對於該程式之程序狀態的觀察，一直沒有一個友善且即時的監測工具。本篇研究在於如何於叢集電腦中，將平行程式的程序識別出來並且將其群組化以便監控，並實做出一個即時監控平行程序狀態的系統工具，並以有條理的方式展現出來。此系統工具可以為程式設計者或是系統研發者，提供一個良好的偵錯環境，而作業系統學習者，也能透過此系統瞭解到程序運作的相關資訊。

關鍵詞：Linux、叢集電腦、即時監控、程序群組、MPI

Abstract

There is no friendly tool for real-time guarding the states of parallel programs which are currently running or intend to run in cluster computers. This research is to develop a real-time monitoring system for the cluster computers. It extracts the important information of the target processes and groups the extracted information according to the correlation of the processes before they are stored in the archive. This system is a useful tool for computer programmers or computer system developers to improve the performance of their work. It is also helpful for the persons who are interested in learning the functionality of an operating systems.

Keywords:Linux, cluster, teal time monitoring, distributed system, process group, MPI

1.前言

近年來由於電腦與網路設備的效能不斷提昇，整合多部個人電腦的運算能力以取代高價的超級電腦已成為趨勢，這樣的架構稱為叢集電腦。叢集電腦擁有強大的運算能力，因此一般用在如影像處理、氣象預測、地震分析等需要強大運算需求的工作，以便在有限的時間內運算出所需的結果。在 Top500 [10]中第一名的超級電腦 BlueGene/L 有部份的應用就是在處理 3D 影像模擬以及地震預測分析。Linux 是一個功能完整的開放性作業系統，彈性的設計被多數的叢集系統採用為作業平台如 Top500 中排名第二的 Jaguar 以及排名第九的 Mare Nostrum 等[10]。在 Linux 叢集電腦上執行平形程式 [1]通常採用 MPI (Message Passing Interface) [6,7,8]這個標準讓叢集電腦各個節點的程序(process) [4]得以順利溝通。於 Linux 作業系統中，一個程序狀態與該程序使用的資源量以及該部機器的狀態可以利用該作業系統提供的指令完成簡易的查詢，例如：該程序使用的記憶體量、處理器的負載量、程序代號、程序名稱等等，但其訊息只適合一般使用者用來查詢有限的資訊，無法對程序與電腦狀態一窺全貌。因此，針對系統設計者或者是程式設計師而言並不適用，例如：我們想得知某個程式在記憶體的哪個區段執行、目前程序還能使用多久的 CPU 資源，某程序結束之後會輪到哪個程序執行，這些要求都沒有辦法透過 Linux 所提供的指令直接獲取。而針對某些特殊情況，如某工作已經開始執行，但因為 CPU 的異常導致其工作在序列(queue)中長時間的等待資源，這類情況的發生使用者沒有辦法利用 Linux 所提供的指令輕易得知。另外在執行平行程式的時候，擷取單一節點的程序狀態是沒有意義的，必須將其有關連性的所有程序資訊一起展現比較。因此我們認為有必要開發一套可應用在叢集電腦上的程序群組即時監系統。

本篇開發的程序群組即時監控系統除了可應用在擷取一般程式之程序相關資訊，亦可用在叢集電腦中用來擷取一個平行程式中程序群組的資訊。平行程式之程序資訊的群組化是透過將平行程式的所有相關程序標記起來，以群組的方式來展現有意義的訊息。本研究利用 ioctl (IO control) [4] 擷取程序資訊。分析 MPICH [9]套件，了解 MPICH 處理平行程式時的原理並實現群組化功能的系統監控工具。另外叢集電腦中執行的所有程序，不一定全部都為平行程式的程序，也有可能是循序程式(sequential program)，所以也必須要有方法識別程序的種類。本監控系統可以提供即時的程序狀態回報功能、紀錄程序狀態的變化、預約監控指定程序等功能，而深入觀察程序在整個執行的生命週期，配合監控整個系統工作的執行情況，可以得知系統是否順利運作，甚至未來可以發展出一套預先判斷系統異常情況產生的監控工具。而此系統工具也可以與 MPE [7]套件結合，深入檢查整個系統的狀況，以利排

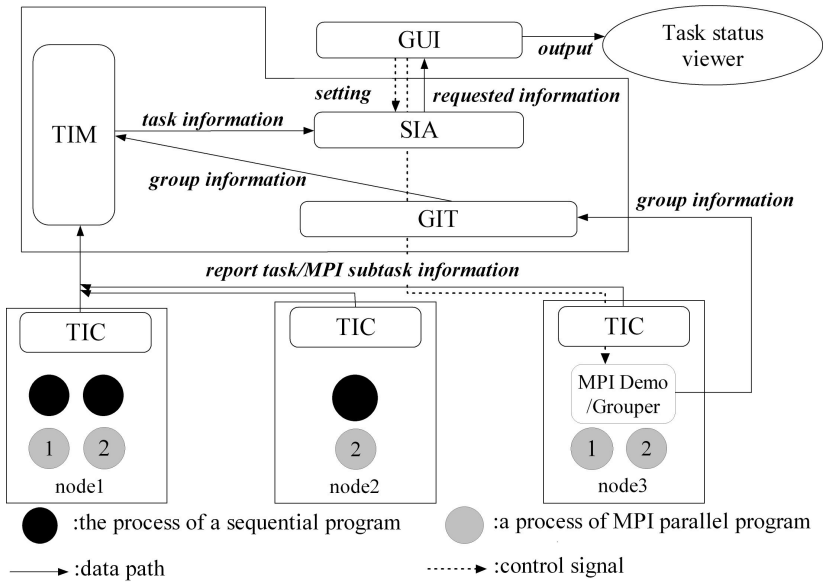

圖 1：監控管理子系統

除系統錯誤，例如：當 MPE 發現在某個運算節點的平行程序執行效能有異常的情況發生，但該運算節點又無執行額外的工作，本研究所發展的程序群組即時監控系統可用來觀察該運算節點所有程序目前的狀況以利排除問題。

本篇論文中，第二章介紹整體的架構以及子系統的功能，第三章將說明如何實現子系統，第四章說明系統工具的使用者介面，第五章針對本篇做總結。

2.架構

系統架構如圖 1 所示，整體包含三個子系統：資訊擷取子系統、資訊收集子系統和資訊展示子系統。在圖 1 中，三個運算節點含有資訊擷取子系統，每一個運算節點可以執行平行程式的一個程序，也可以執行不同的循序程式，而在 Linux 叢集電腦中有一個運算節點除含有 MPI/Grouper 模組，所有的平行程式必須透過此節點將程序分配到各個節點執行，在圖 1 中，運算節點內的灰色圓圈代表某個平行程式的程序，而相同數字的程序代表同一個平行程式所產生的程序；運算節點上方為資訊收集子系統，資訊收集子系統主要為收集資訊擷取子系統所擷取的資料並加以整理；而在圖 1 中，資訊收集子系統上方為資訊展示子系統，主要功能為將系統資訊展現給使用者知道。每個子系統包含了數個模組，以下說明每個子系統以及其所包含的組之功能。

2.1 資訊擷取子系統

資訊擷取子系統包含了以下這兩個模組：
MPI Demo/Grouper 模組和 Task Information Collector (TIC) 模組

- **MPI Demo/Grouper 模組：**當使用者登入叢集電腦後，透過擁有 MPI Demo 的運算節點執行平行程式，經我們修改過後的 MPI Demo 會監看使用者執行平行程式時所使用的相關設定，例如：使用的運算節點數目、使用哪些節點來運算等。為了區別平行程式以及一般單機所執行的程式，Grouper 模組解析 MPI Demo 擷取到的平行程式資訊，並給予平行程序一組特定的 GroupID，在不同運算節點間的平行程序的 GroupID 皆為相同，如此一來便能區分平行程式與循序程式，隨後其相關資訊將被傳送到資訊收集子系統中的 Group Information Table 儲存。

- **TIC 模組：**當循序程式或平行程式送至各個運算節點開始執行時，其執行中的狀態就會不斷的由 TIC 負責收集，此模組會紀錄程序相關資訊，如程序的狀態(state)、旗標(flags)、記憶體使用相關資訊(memory information)、程序識別碼(pid)、程序名稱(command)等十幾項的程序資訊。當程序的資訊被收集到後，資料將送至資訊收集子系統的 Task Information Manager 做更進一步的分析。

2.2 資訊收集子系統

資訊收集子系統包含了以下三個模組：
Group Information Table (GIT) 模組、Task Information Manager (TIM) 模組和 System Information Archive (SIA) 模組

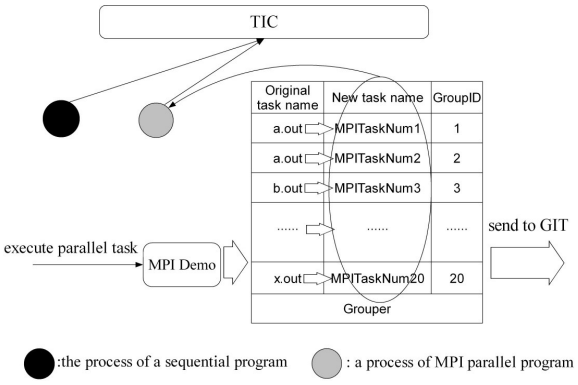

圖 2：Grouper 架構圖

- **GIT 模組：**此模組儲存由節點端的 Grouper 傳送來的資料，並將節點間的群組相依性紀錄起來，接著資料傳送給 Task Information Manager (TIM)。

- **TIM 模組：**此模組收集由節點端的 TIC 以及 GIT 傳送過來的資料，並將資料格式化成 SIA 所儲存的格式以便利用。

- **SIA 模組：**此模組將各個節點間正在執行的程式之程序資訊儲存到資料庫中，並快速作更新，以便當資訊展示子系統的模組(GUI)讀取時，可以達到動態顯示的功能。

2.3 資訊展示子系統

資訊展示子系統主要是 GUI 這個模組。此模組將負責展示 SIA 中資料庫的資料，它有系統地將程序的執行狀態展現給使用者知道，而 GUI 模組有許多展現資料的方式及功能，例如：觀察某個指定的運算節點內全部的程序狀態、針對某個程序做細部的觀察設定、預約程序觀察設定以及程序歷史紀錄設定等功能可以讓使用者選擇

3.模組之實現

以下我們將詳細說明每個模組的實現，也將說明每個模組之間的關係。

3.1 MPI Demo/Grouper 模組

MPI Demo 主要的功能為紀錄使用者執行程式時所使用的相關設定，例如：使用的運算節點數目、使用哪些節點來運算、程式名稱等等。使用者在執行平行程式時，都是使用 MPICH 所提供的指令格式，所以要達到紀錄使用者設定的目的，就需對 MPICH 的 mpirun 指令進行改寫，而 mpirun 的指令是使用 shell script [5] 程式語言，該語言對檔案的處理，例如：批次處理、匯出匯入等很容易完成且不須編譯。改寫的第一步驟為針對 mpirun 在執行平行程式時跟隨執行檔出現的參數做解析，本系統並沒有解析使用 mpirun 指令時所有可搭配的參數選項，只有解析常用之 mpirun 指令參數選項如指定運算節點數目功能、指定特定節點運算功能等等，此步驟對後續的影響非常大，指令格式必須正確的解析，才能夠正確辨別使用者下達的指令，在解析的過程中需要其他 MPICH 的設定檔配合。如設定檔中的 machine.Linux 的主機設定檔，舉例來說，如果使用者沒有指定運算節點，這時我們可以得知運算節點是依照主機設定檔依序使用，假如使用者有用某種規則指定特定運算節點，我們必須先藉由使用者使用的參數來判斷規則，然後判斷出是哪些運算節點會被使用，再紀錄起來。第二步驟為修改 mpirun，透過修改將參數以及執行檔案名稱解析出來並紀錄在資料庫當中，此時我們已可得知使用者執行平行程式時的指令格式以及其目的執行檔，而存放在資料庫當中的資料將會被 Grouper 模組所運用。

而在 Grouper 方面，在此有兩個議題需要思考：(1)如果使用者連續執行兩次相同名稱且參數值相同的平行程式，例如連續執行兩次名稱為 a.out 的平行程式並指定運算節點 1,3，在運算節點 1 和 3 中，該如何辨識兩個平行程序之差異。(2)在之前有提到，對於平行程式，擷取單一節點之程序狀態是沒有意義的，但是當透過 TIC 模組將某個運算節點中所有的程序資訊擷取出來之後，該如何分辨哪些程序為平行程式分配到此運算節點上的程序或者是此運算節點上的循序程式。以上兩個問題可以透過 Grouper 模組來解決，如圖 2 所示，此模組主要功能為將平行程式更名並且給予更名過後的平行程序一組特定 GroupID，以便區分一般程序以及平行程序，當使用者透過 MPI Demo 執行平行程式之後，Grouper 模組會將其檔案名稱依執行順序轉換成 MPITaskNum1、MPITaskNum2、…、MPITaskNum20，並且給予對應的 GroupID，本篇同時使用更名搭配給予 ID 的原因為為後續開發判別更多不同類型的群組化程序做保留，而此動作完成後，便將此資訊傳送給資訊儲存子系統中的 GIT，而本篇設定叢集電腦最多可以同時執行 20 個平行程式。如此一來便可以解決上述兩個議題，透過更名的動作以後，可以藉由更改過後的名稱或者是更簡單的 GroupID 來辨別每個平行程式，因為在不同的運算節點間的平行程序的 GroupID 皆為相同；而因為平行程序多了 GroupID，但在資料庫當中的循序程序並沒有此欄位的資料，在查詢此欄位時資料為 NULL，所以也可以藉由此 GroupID 欄位來辨別平行程式和循序程式。而此作法有以下幾個

限制：(1)假如使用者利用 Linux 作業系統所提供的工作查詢指令如 ps、top 時，此時使用者不認識改名過後的平行程式之名稱，容易造成混淆，但工作查詢指令對於平行程式來說並沒有多大的用處，因為平行程式的執行狀態需要觀察所有相關之運算節點，所以如果使用者欲觀察平行程序的狀態，必須使用本篇的系統監控工具。(2)如果有程式名稱為 MPITaskNum1、MPITaskNum2、...、 MPITaskNum20 等等，雖然 Grouper 模組運作不會出錯，但是容易造成名稱的混淆，所以如果使用本系統監控工具，使用者不可將平行程式名稱命名為類似之格式。

而實做上是使用對於檔案讀寫相當快速的 shell script，將其與資料庫作結合，當有平行程式被執行之後，透過修改過後的 MPI Demo 將執行的檔案名稱以及指令格式紀錄在資料庫當中，而接下來 Grouper 將會修改執行檔名稱，並且將此對應及 GroupID 加入該資料庫，完畢之後將其資料庫的資料傳送給 GIT。

3.2 Task Information Collector (TIC)模組

此模組的主要功能為擷取程序的各種相關資訊；而另外一個功能就是將收集起來的資訊傳送至資訊收集子系統中的 TIM 存放。要擷取程序的相關資訊，一種方式是透過 ioctl (IO control) 來達成，ioctl 為作業系統中 user mode 與 kernel mode 交換資訊時常使用的一種方式，從其名稱上來看，ioctl 本意為針對 I/O 設備進行控制操作，但實際上 ioctl 並沒有限制其操作的設備必須是真正的 I/O 設備，也可以是任何一個核心虛擬設備，也就是一個核心模組。而在核心空間(kernel space)中定義很多的資料結構與函數，因應許多不同的需求所設計，例如：文件類型操作結構 struct file_operations (include/linux/fs.h)、協定類型操作結構 struct proto_ops (include/linux/net.h) 等等，只要在使用者空間(user space)打開這些設備，如 I/O 設備可以用 open function 打開，網路協定可以用 socket function 打開，而開啟相關設備描述檔(在 liunx 系統當中，設備都被視為一個個的特殊檔案)之後，即可透過描述檔中使用 iotcl 來與 kernel space 交換訊息。要自行定義一個特殊的 ioctl 功能，有兩種方式，一種是在現有的 kernel 中直接修改相關代碼，例如想透過 socket 描述符進行 ioctl 的操作，可以在/net/ipv4/af_inet.c 中的 inet_ioctl()函數中添加自己所需的相關程式碼，然後再重新編譯即可；第二種方式是定義自己的 ioctl 核心設備，將其模組化，在需要時載入核心中。了解 ioctl 大致概念後，我們使用上述第二種方式，因為模組化的關係，只需要在使用時將其載入核心，這樣的方式會使核心本體較為精簡，而且不需要重新編譯核心的特性，對於一般使用者較具有親和力。本篇在 Linux kernel [2]中建立一個虛擬 I/O 裝置，而建立該虛擬裝置目的為將其對應到一個核心模組(kernel module)，其核心模組的功能為擷取存放在 kernel space 裡面的程序資訊。透過上述機制可以實做一個擷取程式將程序中的資料傾印並儲存起來，實做上因為核心語言是使用 C 語言加上部份組合語言，所以在實做擷取程式上並不會有太大的困難，需要注意的是不同 Linux 核心版本，對於放置程序資訊的位置以及紀錄的欄位，都有所不同，不過只需要修改擷取程式中擷取資訊的位置，以及擷取的欄位，便能適用於任何的 Linux 核心版本。要注意的是針對於平行程式，TIC 擷取到的程式名稱，並不是原始使用者所執行的平行程式之執行檔名，而是經過 Grouper 轉換過後的名稱。

TIC 的另外一項功能就是將儲存的資料透過網路傳輸程式送至程序中的 TIM，此部份我們採用的是主動式的傳輸，也就是被監控的程序不需要主動要求傳輸，在節點端即會透過 TIC 一秒更新一次所儲存的資料，將資料傳輸至程序，在傳輸過程中，因為資料的正確性非常重要，所以是使用 TCP 傳輸模式。

3.3 Group Information Table (GIT)模組

GIT 負責接收來自運算節點中 Grouper 傳送過來的群組資訊，資訊主要包含了更名後的平行程式名稱以及其對應的原始名稱、GroupID、目的端主機(在本篇是用主機 IP 來代表)，而目的端主機個數通常為多個(執行平行程式)，而此表的資訊將會傳給 TIM 作整合。

3.4 Task Information Manager (TIM)模組

此模組的功能為：(1)收集並格式化 TIC 傳送過來的資料，而接收的資料為每個運算節點中所有程序的資訊(包含平行程序和循序程序)。在接收方面是使用 socket 網路傳輸程式，而封包傳輸格式是使用 TCP 傳輸模式。TIM 模組為被動接收資料，並非主動要求運算節點內的 TIC 提供資料，運算節點會不間斷的迅速提供資料以便更新程序的狀態。(2)將格式化過的程序狀態資訊以及經由 GIT 傳送過來的群組資訊整合成一個完整的資料，並且將資料傳送給 SIA。

3.5 System Information Archive (SIA)模組

此模組的功能為將經過 TIM 處理過後的資料儲入 SIA 資料庫中，此資料庫內存放著最

後經過各個模組整合後的資訊，而此資訊將提供給資訊展示子系統的 GUI 模組使用。實做上我們使用與 PHP5 結合度高且存取資料速度快的 MySQL4 [3]，經過 TIM 處理過後的資料，因為有統一的格式，所以可以整筆透過簡單的 shell script，將資料自動匯入資料庫中，因為資料會迅速的更新(TIM 不停的接收 TIC 以及 GIT 的資訊)，所以資料庫也必須同步的更新，作法是在當 TIM 接收到資料後，馬上執行一系列更新資料庫的動作，最後存入資料庫中，SIA 資料庫會以每秒更新一次的頻率作更動。另外必須注意的是 SIA 會接收使用者設定，如果使用者要求使用紀錄歷史紀錄的功能，這時會啟動一個 shell script 程式，此 shell script 中會對資料庫有匯出歷史紀錄的動作，輸出的資料為文字檔格式，單筆的歷史訊息包含執行時間、執行檔名稱、使用的運算節點、要求紀錄的欄位的資訊，而紀錄的時間間隔根據使用者透過 GUI 的設定而定，最短可以達到每秒紀錄更新一次的頻率。

3.6 GUI 模組

此模組主要提供兩種功能：(1)將 SIA 資料庫的資料透過 GUI 模組的整理有系統的展現給使用者觀看。(2)使用者可以透過 GUI 來對 SIA 資料庫的資料要求動作做設定，例如 GUI 更新程序資訊的頻率以及預約監控的設定，是否紀錄歷史訊息等等。實做上 GUI 模組是用 PHP5 [3]網頁語言所寫成的一個使用者介面，如此一來使用者在遠端也能透過網頁瀏覽器來觀看實驗室中叢集電腦平台上正在執行的工作的細部狀態。當使用者透過 GUI 對 SIA 資料庫的資料要求動作做設定時，在 SIA 端會有程序負責作要求接收，如果 SIA 模組與 GUI 模組在不同的主機上面，兩者之間資訊的傳遞就必須要透過 socket，假如兩個模組是在同一部主機上面，只需使用 shell script 即可，而使用者介面以及一些數據的展示情況將在下一章介紹。

4.使用者介面之數據展現模組

使用者介面主要是透過 GUI 模組展現，下面介紹該系統的使用資訊以及數據展現。

(1) 首頁及說明頁面：

說明了網站的主要功能以及注意事項，如所圖 3 示，比較需要注意的是注意事項第五點以及第六點，第五點在說明假如使用者在監控單一程式模式中，勾選了監看多個運算節點而沒有輸入當初系統分配給該平行程式的 GroupID，此時系統會判定監看多個運算節點

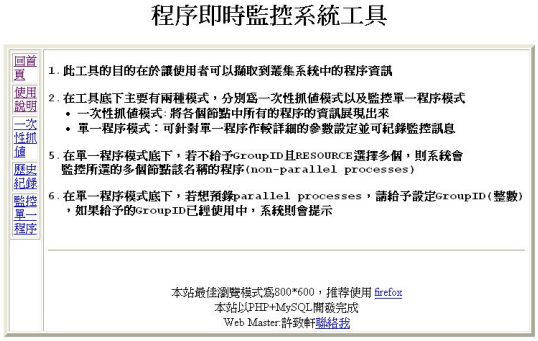

圖 3：首頁及說明文件

該名稱的循序程序而非平行程序，另外假如在運算節點執行的程序當中沒有該名稱，則會轉移到預約模式中，預約模式會將此程序名稱紀錄起來並且在該程序運作時即開始監控；第六點在說明如果使用者想預約監控平行程式，必須要指定 GroupID(1~20)，假如使用者指定的 GroupID 目前在資料庫當中已經有平行程式正在使用，則系統會有錯誤提示。

(2) 全部顯值畫面：

顯示某個節點中所有程序的訊息，因為欄位較多，所以區分成五類：在整個程序資訊結構(task_struct)中前面幾個重要的欄位(important fields coded at the beginning, IFCB)、一些常見的程序資訊(general process information, GPI)、針對程序中檔案系統相關的一些資訊(Miscellaneous information, MI)、針對程序中與記憶體效能相關的一些資訊(memory management and the performance information, MM)、程序身份認證的一些資訊(credentials and the limits, CAL)，如圖 4、圖 5、圖 6、圖 7、圖 8 所示，因為資料庫建構以及辨識方便，在每個分類中都包含著 PID 欄位編碼，另外本篇在 CAL 中資料表中加入了自行定義的 GroupID 欄位。如果要全部一起展示每部主機內的資料，展示頁面會太多，所以展示頁面均使用主機 PC1 中的資料來作展示而不展示所有運算節點，但稍後會針對相同欄位不同主機中的平行程序資訊作比較討論。圖 4 為 PCB 中 IFCB(Important fields hard coded at the beginning)區段的欄位資訊，其中較重要的為 STATE 欄位，STATE 為 0 表示程序正在執行或者是在 ready queue 中等待，而為 1 表示程序被 lock 住，正在等待(在 waiting queue 中等待)某種情況的發生。

PC1

(A)PID (B)STATE (C)FLAGS (D)PTRACES

(A)	(B)	(C)	(D)
1	1	400100	0
2	1	8040	0
3	1	8040	0
4	1	8040	0
8	1	a040	0
9	1	40840	0
10	1	8040	0
46	0	400100	0
2001	1	400140	0
2010	1	400040	0
2159	1	400100	0
2141	1	400100	0
2160	1	400100	0
2166	1	400100	0
2876	1	400100	0
6745	0	400100	0
6816	1	400100	0
6830	1	400100	0
6839	1	400100	0
6868	1	400100	0
6872	1	400100	0
6946	1	400100	0
6958	0	400100	0
7425	0	400100	0

圖 4：PC1 中某個時間點的 IFCB 欄位資訊。

PC1

(A)PID (B)EXIT_CODE (C)EXIT_SIGNAL (D)TGID (E)PRIO (F)STATIC_PRIO (G)TIME_SLICE

(A)	(B)	(C)	(D)	(E)	(F)	(G)
1	0	0	1	116	120	12
2	0	17	2	134	139	244
3	0	17	3	105	110	93
4	0	17	4	105	110	66
8	0	17	8	115	120	96
9	0	17	9	125	120	6
10	0	17	10	113	110	4
46	0	17	6546	116	120	29
2001	0	17	2001	116	120	2
2010	0	17	2010	116	120	6
2159	0	17	2159	117	120	15
2141	0	17	2141	116	120	89
2160	0	17	2160	116	120	9
2166	0	17	2166	116	120	3
2876	0	17	2876	117	120	93
6745	0	17	6745	116	120	26
6816	0	17	6816	115	120	24
6830	0	17	6830	115	120	30
6839	0	17	6839	115	120	30
6868	0	17	6868	115	120	41
6872	0	17	6872	116	120	1
6946	0	17	6946	116	120	90
6958	0	17	6958	116	120	17

圖 5：PC1 中某個時間點的 GPI 欄位資訊。

PC1

(A)PID (B)LINK_COUNT (C)TOTAL_LINK_COUNT

(A)	(B)	(C)
1	0	0
2	0	0
3	0	0
4	0	0
8	0	0
9	0	0
10	0	0
46	0	1
2001	0	0
2010	0	0
2159	0	1
2141	0	0
2160	0	1
2166	0	1
2876	0	0
6745	0	0

圖 6：PC1 中某個時間點的 MI 欄位資訊。

PC1

(A)PID (B)min_flt (C)maj_flt

(A)	(B)	(C)
1	310	18
2	0	0
3	0	0
4	0	0
8	0	0
9	0	0
10	0	0
46	364	0
2001	282	0
2010	136	0
2159	468	0
2141	443	1
2160	468	0
2166	467	0
2876	1458	5
6745	302	0

圖 7：PC1 中某個時間點的 MM 欄位資訊。

圖 5 為 PCB 的 GPI(general process information)欄位資訊，其中較為重要的欄位有 PID, TGID, PRIO(priority), 和 TIME_SLICE 等欄位， PID 為系統給予程序的代號。TGID 表示 thread 狀態的一個群組 ID，假如新的程序是被既有程序所建立的，其 TGID 的值即就為程序之 PID 值。PRIO 系統分配給該程序優先權值，會隨著系統的情況而變動，而 STATIC_PRIO 則是系統在程序執行開始給予該程序之預設優先權值，不會隨著時間改變，而 TIME_SLICE 代表系統分配給此程序使用 CPU 時間 的長短。圖六為 PCB 中 MI(memory Information)欄位資訊，其中遞迴的 symbolic link 紀錄在 LINK_COUNT，遞迴的 symbolic link 類似作業系統中檔案的”捷徑”;而連續性的 symbolic link

PC1

(A)PID (B)UID (C)GID (D)USER (E)COMM (F)GroupID

(A)	(B)	(C)	(D)	(E)	(F)
1	0	0	root	init	
2	0	0	root	ksoftirqd	
3	0	0	root	events	
4	0	0	root	kblockd	
8	0	0	root	pdflush	
9	0	0	root	kswapd0	
10	0	0	root	aio	
46	0	0	root	test	
2001	0	0	root	sendmail	
2010	smmsp	51	smmsp	sendmail	
2159	0	0	root	login	
2141	0	0	root	mdadm	
2160	0	0	root	login	
2166	0	0	root	login	
2876	mysql	27	mysql	mysqld	
6745	0	0	root	ioctl	
6816	testuser1	0	testuser1	a.out	
6830	testuser2	0	testuser2	a.out	1
6839	testuser2	0	testuser2	a2.out	2
6868	0	0	root	mysql	
6872	0	0	root	bash	
6946	0	0	root	test_server	
6958	0	0	root	test	
7425	0	0	root	ioctl	

圖 8-1：PC1 中某個時間點的 CAL 欄位資訊。

PC2

(A)PID (B)UID (C)GID (D)USER (E)COMM (F)GroupID

(A)	(B)	(C)	(D)	(E)	(F)
1	0	0	root	init	
2	0	0	root	ksoftirqd	
3	0	0	root	events	
4	0	0	root	kblockd	
8	0	0	root	pdflush	
9	0	0	root	kswapd0	
10	0	0	root	aio	
46	0	0	root	test	
2008	0	0	root	sendmail	
2040	smmsp	51	smmsp	sendmail	
2141	0	0	root	mdadm	
2164	0	0	root	login	
6872	0	0	root	bash	
7450	0	0	root	ioctl	
7200	testuser2	0	testuser2	a2.out	2
8100	0	0	root	ioctl	

圖 8-2：PC2 中某個時間點的 CAL 欄位資訊。

紀錄在 TOTAL_LINK_COUNT，表示該程序連接到其他程序的連結數目。圖 7 為 PCB 中 MM(Memory Management)欄位，其中 min_flt 代表較不重要的記憶體分頁錯誤產生數量(不需重新讀取 memory page 校正)，maj_flt 也是代表較危險的記憶體分頁錯誤產生個數，需要重新讀 memory page 來校正。圖 8-1 與 8-2 分別為同一時間點對於 PC1 以及 PC2 中 CAL

圖 9：單筆程序監控之設定畫面。

(credentials and limits)欄位的比較，其中 CAL 欄位包含了程序 ID (PID),使用者 ID (UID),使用者群組 ID (GID)，使用者名稱(USER)，執行檔案之名稱 (COMM),還有我們自行定義的 GroupID，以下我們針對兩者欄位內資料的內容作分析。我們首先可以在圖 8-1 觀察到，在 PC1 中，有兩個平行程序正在執行分別為 a.out (GroupID 為 1)以及 a2.out (GroupID 為 2)，另外我們可以注意到 PID 為 6816 和 6830 的兩個執行檔名稱相同都為 a.out，不過我們可以藉由 GroupID 分辨出何者為平行程序以及何者為循序程序。同時觀察圖 8-1 和 8-2，我們可以看出名稱為 a2.out 的平行程式正同時在這兩個運算節點中執行，因為其 GroupID 都為 2，如果我們不透過 GroupID 欄位，單純觀察 PC1 以及 PC2 的 a2.out 程序，我們無法得知兩者之間的關聯性，所以透過 GroupID 機制可達到辨識群組化的功能。

(3) 單筆程序監控相關畫面：

單一程序監控介面可以設定指令、GroupID、秒數、程序欄位指定、運算節點設定、是否紀錄歷史訊息等功能，如圖 9 所示，而假如設定有錯誤，介面會產生偵錯訊息讓使用者知道如圖 10 所示。最後設定完畢之後，畫面切換到目前所有單筆追蹤情況的一個目錄，如圖 11 所示，而在圖 11，畫面會展現目前追蹤的程序還有其相關設定，而未執行中的程序，則等待追蹤當中，而對於欲預約監控平行程序，因為此時系統沒有給予 GroupID，必須事先設定時給予 GroupID，如圖中 a.out4 所示，使用者預先設定 GroupID 為 4，而系統會判斷此 GroupID 是否被使用，如果沒有的話則允許使用，另外使用者也可以執行刪除錯誤設定的程序。

圖 10：單筆程序監控之設定偵錯畫面。

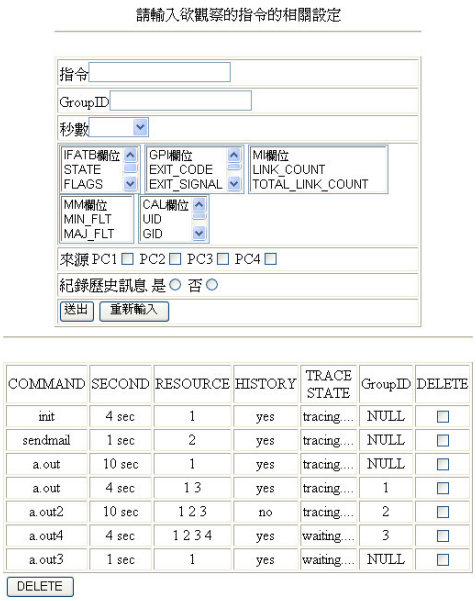

圖 11：單筆程序監控的整體展示頁面。

5.結論

本篇在功能的實做上達到了期望的目標，使用者介面可以清楚的展現後端叢集電腦中平行程式之程序的執行狀況，而將各個功能模組化的實做方式可以達到明確分工、偵錯方便以及將來功能更新上修改程式的便利性。不過還是有一些問題存在，例如：使用者假如預約 20 筆的平行程序，此時 GroupID 皆被保留住。倘若使用者想要執行這 20 筆以外平行程序，系統將會發生錯誤。另外還有些功能尚待加強以及改進，例如：安全性與身份認證的動作，不同使用者觀看不同程度的程序執行狀態、程序更新的速度提昇以及當程序被擷取資訊時，系統額外負擔的降低。身份認證機制的實現可以透過擷取程序擁有者這個程序資訊欄位，配合 Linux 系統處理帳號密碼的相關指令搭配 shell script 實現出來，而程序擷取的更加即時性以及系統額外負擔的降低，或許就必須要更深入了解系統核心的運作原理，找出更佳的執行方式或者是整合與簡化擷取步驟來實做。而此監控系統工具將來能配合一些異常狀況統計數據獲得的結果，例如在經過長時間的監控之後，得知某種異常現象可能為某種情況而導致，而發展成一套可以預防系統異常的監工具。

目前網際網路的新世代「網格」(Grid)[11]是一個嶄新的科技概念。網格將由網際網路以及其中的各類資源為主幹，使用者將可以像使用自己的電腦一樣來使用所有的這些程式、服務和儲存空間。在如此龐大的系統中，如果有伺服器節點出錯，影響的層面將比單一機器甚至是叢集系統都還要巨大。因此，精細且清楚明瞭的監控系統是不可或缺的。運用在網格的監控系統應該具有在問題發生時，可迅速找出原因以及問題的節點。以本篇的監控系統來說，如果將其直接運用在網格上面將會有資訊雜亂的缺點，因為網格含有非常大量的節點以及程序，若將每筆程序資訊全部都擷取的話，則耗費的頻寬將是巨大且浪費的。故需要有不同的機制來監控網格中程序的運作，希望在往後的研究，能夠針對該方面網格的技術與應去作相關的研究。

致謝

此篇論文為國科會專題研究計畫之相關成果，編號 NSC 95-2815-C-197-007-E。

參考資料

[1]Ananth Grama, Vipin Kumar, Anshul Gupta, An Introduction to Parallel Computing: Design and Analysis of Algorithms 2/e, Addison Wesley, 2003-02-16

[2]Daniel Bovet, Marco Cesati, Understanding the Linux Kerne l3/e , O'Reilly, 2005-11-17

[3]Elizabeth Naramore, Jason Gerner, Yann Le Scouarnec, Jeremy Stolz, Michael K. Glass, Beginning PHP5, Apache,and MySQL Web Development, Wrox Press, 2005-01-24

[4]John O'Gorman, The Linux Process Manager, John Wiley, 2003-02-13

[5]Meadors, Linux Shell Script Programming, THOMOSON, 2003-03-01

[6]Michael J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw-Hill, 2003-06-05

[7]William Gropp,Ewing Lusk, Guide of mpich,a portable Implement of MPI Version 1.2.2

[8]MPI document http://www.mpi-forum.org/docs/docs.html

[9]MPICH document http://www-unix.mcs.anl.gov/mpi/mpich1/

[10]http://www.top500.org/
[11]lan Foster, Carl Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, 2/e, Morgan Kaufmann, 2003-11-08