東海大學資訊工程學系研究所

碩士論文

指導教授：陳隆彬　博士

以自動化部署技術實現具有安全性與回饋機制的合作運算平台

The Development of Cooperative Computing Platform with Security and Feedback Mechanism via Automated Deployment

研究生：李宸光

中華民國一百零一年七月

摘要

BOINC 志願運算平台是世界上最大的運算網格之一，已有許多科學運算計畫採用此種方案並獲得不錯的成效。本論文發展一個新的合作運算模式，使得第三方系統能透過志願運算平台獲取運算資源，以較低成本實現具有規模適應性之電腦計算系統。合作運算平台採用部署代理人技術。對於志工電腦，代理人能簡化虛擬機器的部署程序，提升志工意願；對於志願運算平台和第三方系統而言，代理人能確保身份識別的一致性，提高志工積分回饋機制的精確性。合作運算平台運用工作轉換、虛擬機器部署代理人、一致性身份識別等機制，將 BOINC 與第三方系統整合成為緊密連結的系統。

關鍵字： 志願運算、數位學習、虛擬機器

ABSTRACT

BOINC volunteer computing platform is one of largest grid computing in the world, there are many scientific computing project of use such the method and get good results. In this thesis, the development of a new cooperative mode, the third-party system through volunteer computing platform to obtain computing resources, and to implement large-scale adaptation of the computer system by low costs. Cooperative computing platform use the virtual machine deployment of agent technology. For volunteer computer, the agent can simplify procedures, enhance the volunteer will; for volunteer computing platforms and third-party systems, the agent can ensure the consistency of identification, increase volunteer integral feedback mechanism accuracy. Cooperative computing platform with the work of conversion, virtual machine deployment agent, consistent identification mechanism, and the use of these mechanisms integration BOINC and third- party systems become closely tied to the system.

Keywords：Volunteer computing, E-learning, Virtual machine

目錄

圖目錄

第一章　簡介

在這資訊技術日益成熟的年代，網路應用程式已經普及到各方面。所謂分散式運算即是由數台個人電腦各自獨立管理資源及處理作業，運用網路通訊作為相互溝通的橋梁，讓不同的電腦合作運算以減輕龐大資料量所需承擔的工作量。主從式架構[1]是一種開發應用程式及維護容易的分散式計算架構，它將伺服端與客戶端區隔開，每一個客戶端軟體都可以向一個伺服器發出請求。用戶端的應用程式扮演著和使用者溝通的角色，它強調簡單且操作容易的使用界面，以提供使用者進行查詢、修改等輸入輸出的作業，而伺服器則負責執行用戶端應用程式所傳來的命令，並將處理的結果回傳給用戶端的應用程式，直接將結果顯示在使用者的眼前。

1.1　研究動機與目標

主從式缺點為隨著系統功能與客戶端成員的增加，對伺服端的負荷會越來越大。現在有許多的科學研究計畫會產生極大量的資料，有時候大到無法利用超級電腦處理，或者不適合，科學家因此發明了利用網際網路及個人電腦組織的分散式運算[12][13]。研究計畫透過志願運算平台將資料分成數個小檔案，透過網際網路傳送至志願參加運

算的電腦，當這些電腦閒置的時候，預先安裝的 BOINC 軟體就會開始處理資料，運算完畢之後再傳回中央電腦伺服器。

隨著 BOINC 專案數量增加，愈來愈多的第三方系統希望與 BOINC 連線，透過 BOINC 獲取廣大的計算資源。為了達到這個目的，這些系統必須將其計算工作改寫為 BOINC 專案。然而，這些第三方系統有些已經運作一段長時間，不易改寫。再者，有些第三方系統是屬於互動性的應用程式，例如數位學習或是 Game AI 搜尋等，這些類型的應用都需要大量計算資源，同時必須保留原有的應用程式。

本論文研究一種新的合作運算平台模式的概念，此種合作運算平台能整合第三方系統與 BOINC 志願運算平台，以較低成本實現具有規模適應性之電腦計算系統。本論文闡述 BOINC 與應用程式之間的安全性與回饋機制等多個重要議題的解決方案。

本論文以數位學習網站為應用，將其與 BOINC 整合，發展出低成本且具有規模適應性的數位學習系統。例如 UVa 線上評分系統 [2][20]提供一個只需在可以上網的環境，便能讓學生學習編寫程式的網站，同時該系統會對程式進行編譯，並使用測試數據來檢驗程式的正確性。數位學習網站中的題目有難度區分，除了有要編寫完整程式碼的題目，另外還有填空題的題型，對於初學者而言循序漸進的學習方式是了解程式如何撰寫最好方式。

1.2 論文貢獻

過往 BOINC 志願運算的應用程式進行運算工作時，是直接在志工主機的原始作業系統中執行，因此計畫資料保密性與志工電腦本身的安全性等都是重要課題。雖然數位學習網站提供了一個撰寫程式的平台，但為了能夠驗證多位學生所編寫程式碼的正確性，運作系統的主機勢必要有強大的處理能力。單靠一台伺服器運算龐大的資料量勢必會降低運作效率，所以我們將配合 BOINC 平台[3][11][14]來將工作分散給各個志願電腦運算，伺服器只需專注在工作的分配、結果的驗證與管理虛擬機器上，不必再進行運算的動作，大大的減輕伺服器負擔。

合作運算平台採用部署代理人技術。對於志工電腦，代理人能簡化虛擬機器的部署程序，提升志工意願；對於志願運算平台和第三方系統而言，代理人能確保身份識別的一致性，提高志工積分回饋機制的精確性。合作運算平台運用工作轉換、虛擬機器部署代理人、一致性身份識別等機制，將 BOINC 與第三方系統整合成為緊密連結的系統。

1.3 論文大綱

本篇論文組織為：第二章介紹虛擬化技術與 BOINC 平台，第三章介紹合作運算平台，第四章描述了虛擬機器部署代理人，第五章展示系統實作與討論，第六章為結論。

第二章　背景

2.1　虛擬化技術

虛擬化[4]最早出現於 1960 年代 IBM M44/44X 主機中，當時 IBM 公司為了讓使用者充分地利用大型主機資源，於是發明了虛擬化技術，允許用戶在一台主機上運行多個作業系統。IBM 公司運用邏輯分區(LPAR)與動態邏輯分區(DLPAR)的概念，將 CPU 或是磁碟機等資源分配給獨立的分區，在管理層面上提供很高的彈性與便利性。

虛擬化允許用戶在單一主機上運行多個異質作業系統，充分地利用昂貴的大型主機資源。雲端運算的基礎設施層 (IaaS，Infrastructure as a Service)運用虛擬化技術來管理虛擬機器(VM，Virtual Machine)[15][16]。由於虛擬技術功能愈來愈強大，伺服器的效能不斷的提昇，IaaS 運用虛擬化技術妥善利用電腦、網路、空間、電力、空調等資源。

Hypervisor[5]，又稱 virtual machine manager (VMM)，是虛擬化的核心元件。Hypervisor 屬於硬體虛擬化技術，讓多個 guest OS 能在同一台電腦運行。虛擬化的 Hypervisor 有以下兩種型態：

- Type 1 (Bare-Metal hypervisor)又稱為 Native VM。表示 hypervisor 直接安裝於空機或新機上，直接掌控硬體資源，硬碟無須先有 OS。VMware ESXi、Microsoft Hyper-V 等系統屬於此類。

- Type 2 (Hosted hypervisor)也稱為 Hosted VM，hypervisor 是安裝在 Windows 或 Linux 作業系統之中，而 guest OS 再安裝於 hypervisor 之中。Vmware Workstation、VirtualBox 等系統屬於此類。

2.2 BOINC 運算平台

柏克萊開放式網絡運算平台(Berkeley Open Infrastructure for Network Computing，簡稱 BOINC）[6][7]，是目前最大的分散式運算平台，其目的在為各研究者提供匯集全球各地大量個人電腦的龐大運算能力。該系統原本是設計於 SETI@home[8]計畫，之後也廣泛的被應用在數學、醫學、分子生物學、氣象學與天文學等各個科學領域[17][18][19]。

BOINC 是一個開放原始碼軟體，任何人都可以輕易取得，不僅能貢獻自己的電腦加入計畫幫忙運算，也可建立自己的計畫。BOINC 中最著名的計畫是 SETI@home，是一個搜尋外星人的計畫，該計畫靠著絕佳的推廣度，與大家對於外星文明的好奇心，吸引了許多的自願者前來貢獻他們的電腦資源進行運算。

BOINC 是由 BOINC 伺服器與客戶端程式所組成。BOINC 伺服器主要的任務在於協調各志願者的工作，產出並發送工作檔（WorkUnit），驗證運算回傳的結果，處理使之成為研究人員所需要的數據。志願者則透過客戶端程式的命令列(command line)或圖形使

用者介面(GUI)加入個別的 BOINC 計畫，計畫伺服器會向個人電腦提供工作檔，當個人電腦完成工作後，客戶端程式再把結果上傳至伺服器。安裝 BOINC 客戶端的個人電腦在閒置時會使用個人電腦的 CPU 進行運算，即使個人電腦正被使用，假如仍有空閒的 CPU 週期，BOINC 也會用來運算。然而個人電腦可能會在運算過程出現錯誤，所以 BOINC 伺服器一般會把同一工作檔複製後並傳送給數個志願者，最後再比較各個回傳結果。

2.3 BOINC 工作流程

BOINC 的工作流程如圖 1，Work generator 先從 BOINC Project 中取得任務並轉成工作檔儲存於資料庫中。Transitioner 將工作檔複製成多份 result 存放於資料庫，Feeder 再抓取資料庫中尚未執行的 result 放於 Queue 中，等待 Scheduler 抓取派送給志工電腦運算。

Scheduler收到志工電腦回傳的運算結果後，Transitioner先檢查工作檔的複本是否都已收到，接著Validator會比對複本的結果，若結果相同則再送給Assimilator進行合併複本的動作，同時File deleter刪除不再使用的工作檔。

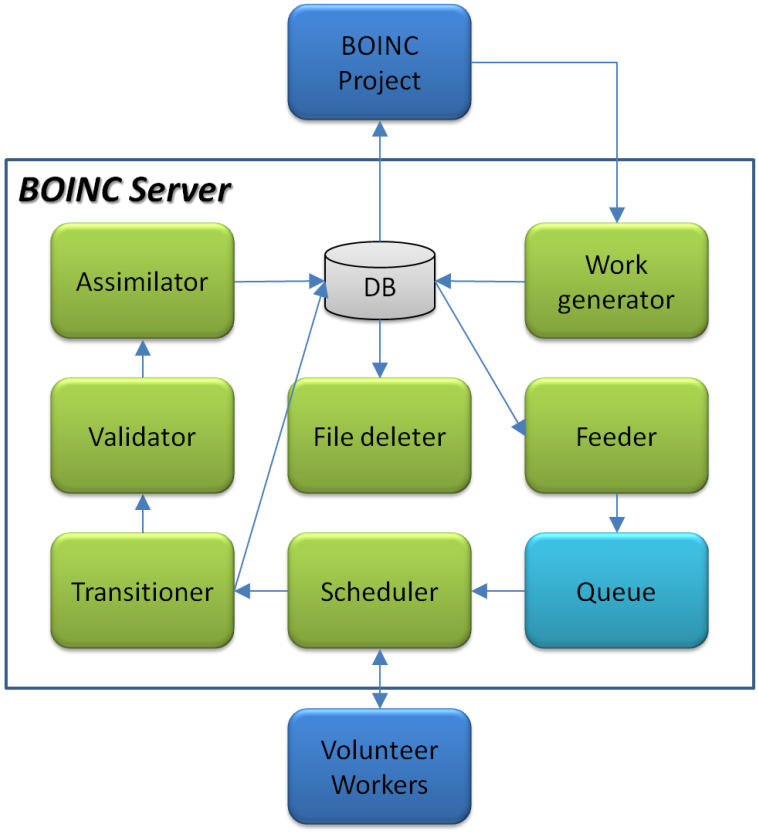

圖 1 BOINC 工作流程圖

第三章　合作運算平台

3.1　第三方系統與志願運算平台

所謂第三方系統指的不是由某個軟體同時開發，但是可以與該軟體整合在一起使用的系統。對於 BOINC 志願運算平台而言，有許多人希望將現有系統與 BOINC 整合來達成志願運算的技術，以減輕伺服器的工作量。

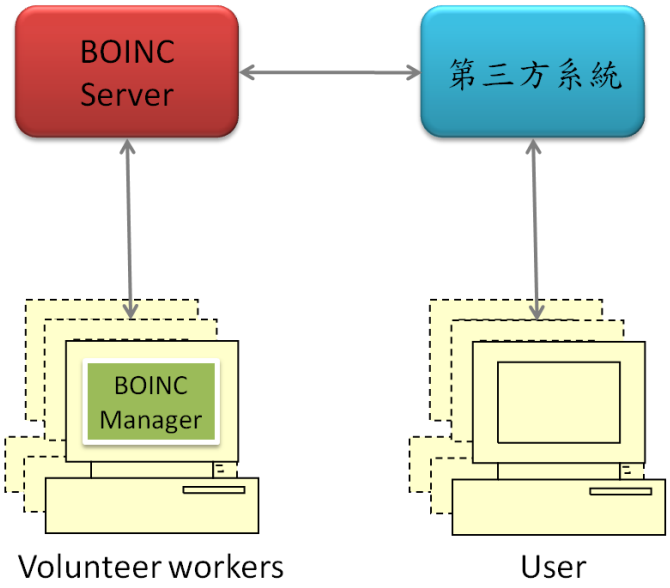

圖 2 第三方系統與 BOINC 志願運算平台的整合

目前有不少整合第三方系統與 BOINC 的例子，運作模式如圖 2 第三方系統與 BOINC 志願運算平台的整合圖 2，

需要運算的工作是由使用者操作第三方系統產生或第三方系統自行產生，再由 BOINC Server 將工作分派給志工電腦做運算，最後再將運算結果回傳給第三方系統。[9]即是圍棋遊戲與 BOINC 整合的應用，

電腦會詢問下圍棋時下一步棋該如何走；BOINC 伺服器中的 Work generator 將該問題產生成可運算的工作檔，經由 Scheduler 派送工作給 Client 端去執行運算，透過運算完的結果可分析出下一步棋可以走的位置資訊，並回傳該資訊給圍棋遊戲的電腦。

圖 3 是 Online Judge 與 BOINC 整合的應用，學生直接在 Online Judge 上進行測驗，Work generator 將學生撰寫的答案產生工作檔，再交付給 Clients 去運算。與圍棋遊戲不同的地方是，學生不僅是工作的產生者，同時也可能是運算工作的志願者。

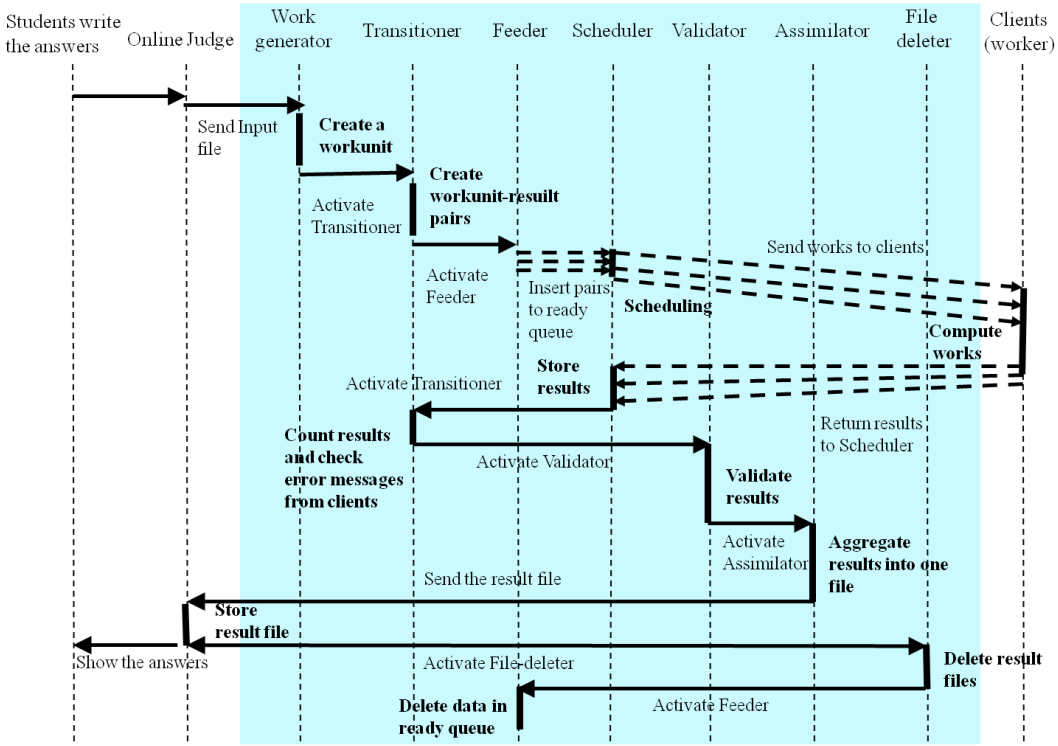

圖 3 Online Judge 與志願運算平台的整合

傳統大型互動式教學網站的建置與維護成本頗高。首先必須建置高運算能力的伺服器來因應尖峰時刻的客戶需求，但這些伺服器在非

尖峰時刻大多處於閒置狀態。另外，程式語言的線上教學與評量系統讓使用者上傳程式碼到伺服器端接受評量，讓伺服器暴露於危險中。本論文以數位學習網站做為第三方系統，透過志願運算平台獲取運算資源，整合成為一個合作運算平台。在此平台中，除了讓學生可以在數位學習網站上答題，更鼓勵學生成為志願者幫忙運算。因此，學生不僅是生產者，負責送出工作到教學網站，同時也是消費者，負責計算工作。透過管控生產者與消費者的數量，每個客戶端只需貢獻少許資源便能組成高效能的教學系統。

合作運算平台延伸自 BOINC 平台，但是有很大不同。比較如下：

- 大多數的志願運算平台是匿名制，而合作運算平台的客戶端身分是實名制。採用實名制的目的是要配合回饋機制，以鼓勵學生加入運算工作，畢竟參與人員多寡是合作運算系統成敗關鍵。
- 合作運算平台的客戶端不被允許觀看計算內容，以線上教學應用為例，客戶端不可檢視另一位學生所上傳的作業程式碼與輸入輸出等資料。客戶端必須部署虛擬環境，以區隔客戶端本地作業系統與合作運算環境。
- 合作運算平台其工作產生者和工作解決者都是客戶端主機即學生電腦；合作運算平台的工作是由 BOINC 系統產生，客戶端主機只負責解決工作。因為學生可以上傳任何程式碼到伺服器，合作運

算平台的工作與資料是動態產生的，事先無法預知；一般 BOINC 平台專案的工作是靜態固定的，例如某一支天文資料分析程式，只有被分析的資料是動態隨時間收集而來的。

3.2 合作運算平台之系統架構

我們運用 BOINC 平台把編譯與執行的動作分給志工電腦去運作，讓數位學習網站本身只需負責學習進度的管理，進而提昇數位學習網站的穩定度。合作運算平台可分成兩大部分，分別為數位學習網站與 BOINC 平台。

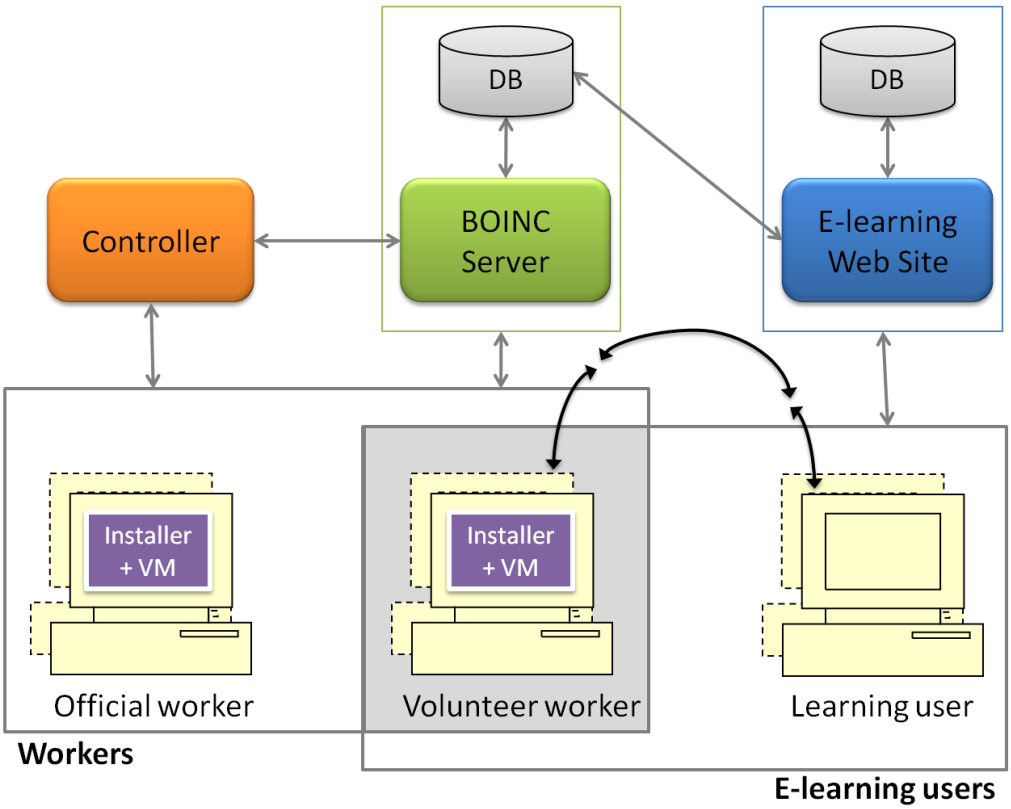

圖 4 合作運算平台架構圖

如圖 4，學生透過數位學習網站進行教材學習，依進度上傳作業

程式碼。數位學習網站會將程式碼傳送給 BOINC 資料庫，BOINC 會到資料庫中檢查是否有需要編譯執行的程式碼，若有則產生工作檔派送給志工電腦進行運算。運算結果先回傳到 BOINC 資料庫進行兩份相同工作檔的結果比對，若不同則重新傳送工作檔，相同則將結果回傳給數位學習網站，由網站資料庫進行分析答案是否正確。使用者能從網頁得知處理進度與結果。

架構圖中的 Controller 負責客戶端虛擬機器的管理。Controller 可以對虛擬機器下達指令來達到管理的功能，如指定用戶暫停或執行專案計畫、查詢虛擬機器連線狀態、或虛擬機器版本的更新等。

3.3 數位學習網站與 BOINC 平台之身份綁定

3.3.1 工作編號對應

當學生在數位學習網站撰寫完程式碼，系統會將程式碼上傳至 BOINC 資料庫中，且依序以學生帳號、題目編號及一組隨機產生的六位數字來當作該程式碼名稱。當資料庫中多出需要編譯的程式碼，Work generator 會至資料庫中抓取該程式碼，並將程式碼包裝成工作檔，以該程式碼名稱另外加上建立時間及第幾組測試編號來命名。此命名方式可以得知工作是由哪位學生所產生，及學生所撰寫的題目編號，以確保當工作運算完後，系統能將結果回傳給撰寫程式碼的學生

知道。

3.3.2 志工身份認證

每位學生在數位學習網站上各自都有一組帳號密碼，此組帳號密碼是數位學習網站管理者依據學生修課名單所給予的，學生可以利用該帳號密碼登入數位學習網站進行線上答題的動作，同時數位學習網站會對學生的答案進行編譯與給分的動作。當編譯程式的份數少時，數位學習網站的主機還有能力運作，但是當所要編譯的程式份數量大時，主機的運作效率可能會降低，此時就需要學生來當作志願運算者，以減輕主機的負擔。

志工身份認證指的就是判斷數位學習網站與 BOINC 平台上的帳號名稱是否相同。因此系統採用了實名制的方法，實名制指的是同一位使用者，在數位學習網站與 BOINC 平台必須使用相同身份。透過實名制系統將能確認某位使用者對於專案的貢獻度，鼓勵使用者參與。本系統將使用自動化部署，將數位學習網站的身分匯入BOINC平台，以確保實名制的施行。

3.3.3 回饋機制

為了增加學生擔任志願者的意願，一旦學生加入志願運算的工作

中，系統便會依據志願者參與運算的貢獻度來給予學生在數位學習網站中額外的分數，因此要求志願者在數位學習網站與 BOINC 運算專案的帳號必須相同，如此才能讓學生進行加分的回饋動作。

另外，BOINC 會根據學生身份給予產生的工作優先權。學生從數位學習網站上傳程式碼後，BOINC 會對該工作進行身分確認的動作，若該位學生不是志願運算者，則 scheduler 給予該份工作較低的執行優先權，反之，若學生是志願運算者則 BOINC 會優先派送他的工作到志工電腦作運算。

第四章　虛擬機器部署代理人

4.1　自動化部署之架構

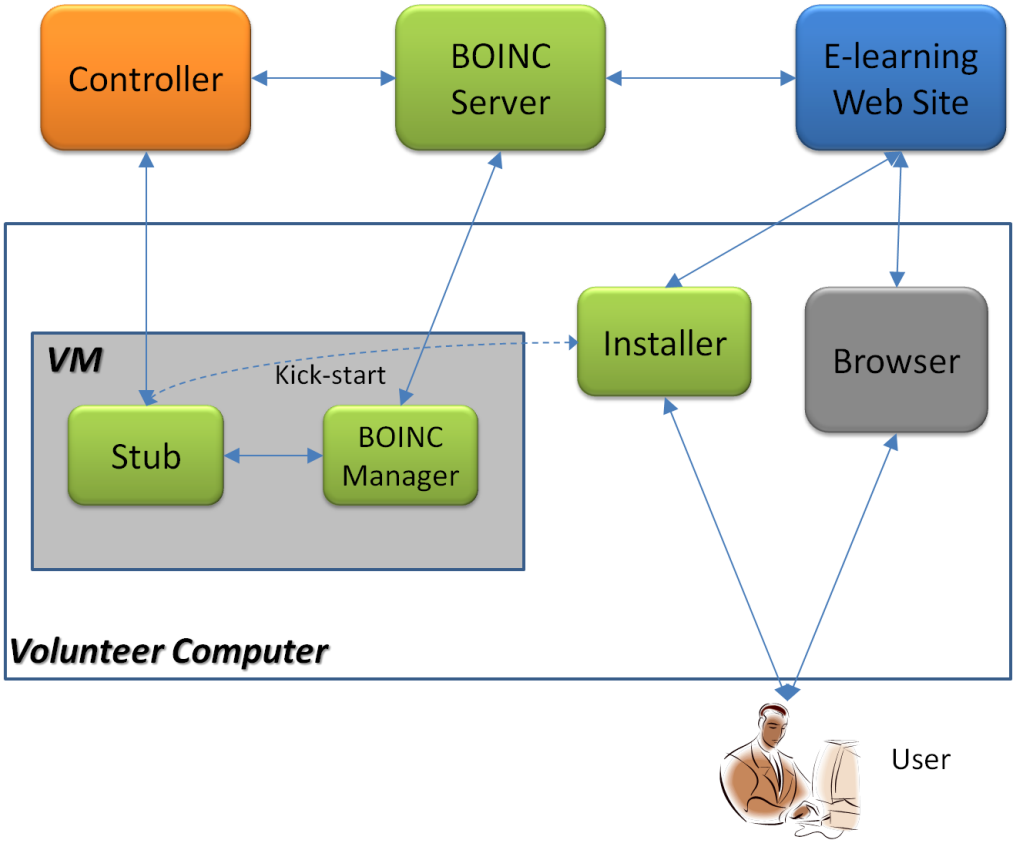

圖 5 自動化部署

關於自動化部署虛擬環境，本系統發展一個以代理人技術為基礎，易於使用且安全的工作流程來實施自動化部署，如圖 5。開發流程主要依賴兩個元件：Installer 與 Stub。志願者先到數位學習網站下載 Installer，利用 Installer 抓取數位學習網站的帳號資訊，並使用該帳號去註冊成為 BOINC worker，利用此方法能夠讓數位學習網站與 BOINC 兩個平台的會員帳號相同，以確保實名制的施行。

Installer 被執行後，會依照以下步驟實施自動化部署：

1. FTP 取得虛擬機器檔案。

2. 啟動虛擬機器。

3. 在虛擬機器中執行 BOINC Manager(client 端)。

4. 以數位學習網站帳號密碼註冊成為 BOINC worker。

5. 連線到 BOINC 運算計畫，參與解決工作。

志願者可以自由的開啟或關閉虛擬機器，但在虛擬機器運行期間，志願者不被允許登入虛擬機器。因此在虛擬機器配置與操作的過程中，所有的訊息交換是在 Installer 與 Stub 間，此方式是為了隔離虛擬機器不被志願者所存取，即不給予志願者非法窺探的機會，確保資訊的安全性。

4.2 伺服端

伺服端主要是由數位學習網站、BOINC 平台與 Controller 組成。一般在沒有 BOINC 平台的情形下，數位學習網站也有著處理資料的能力。當學生上傳欲評分的程式碼，數位學習網站將程式碼進行編譯執行並將結果跟預設好的正確解答比較，再把結果放回資料庫中，並將結果顯示到學生的瀏覽器。

通常在作業繳交期限之前幾分鐘，或是某些上機時段，大多數學生總會同時上傳程式碼，這時候數位學習網站就要依靠 BOINC 平台強大的分散式運算能力來幫忙運算。首先學生使用數位學習網站來撰

寫程式並上傳，上傳後的程式碼被儲存於資料庫中以便BOINC抓取，接著 BOINC 將程式碼取出包裝成工作檔，並等待志願運算者抓取工作檔編譯處理。志願者編譯處理完工作檔後會產生輸出檔，輸出檔會回傳給 BOINC 進行比對相同的工作檔結果是否相同，最後再由數位學習網站進行答案的比對，並將結果展示給學生看。

Controller 可以對虛擬機器下達指令來達到管理的功能。由圖 6 可知，Controller 可以下指令來要求志願者暫停或執行運算、更新專案計劃、取得目前已連線志工電腦的狀態或數量。Controller 也可以呼叫 BOINC Server 產生工作，並將結果寫入資料庫或移除資料庫的資料。

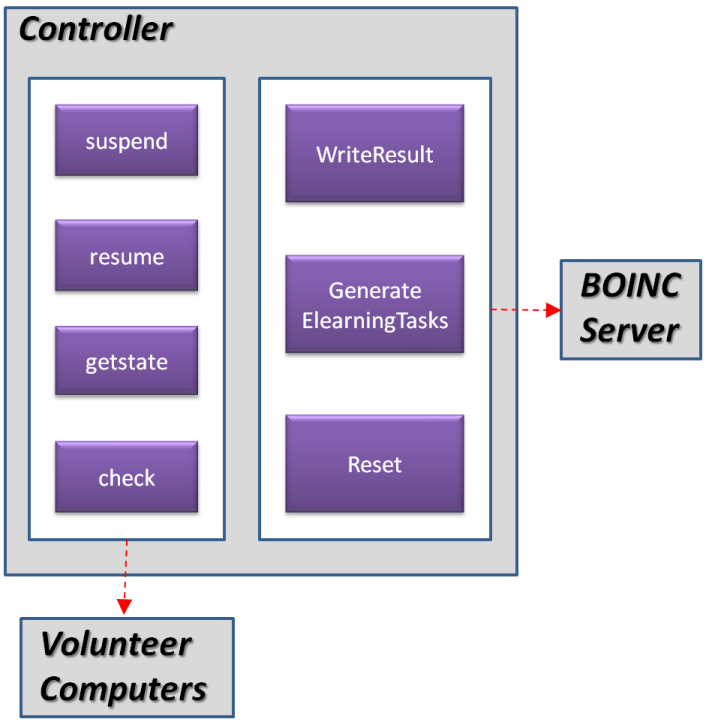

圖 6 Controller 功能

4.3 客戶端

本系統的客戶端主要由 BOINC Manager、Installer 與 Stub 所組成，當要部署一個志願者時，使用者先登入數位學習網站並下載 Installer 用來訂製每一個使用者的帳號。如圖 7，Installer 壓縮檔內含有 Installer 執行檔與 libcurld.dll 檔，libcurld.dll 檔讓 cURL[10]的功能得以實現。

執行 Installer 後會先要求學生輸入在數位學習網站的帳號與密碼，若正確會詢問學生是否要下載虛擬機器，該虛擬機器內已安裝 BOINC Manager、Stub 與 StartStub.bat。StartStub.bat 在虛擬機器系統啟動後便會開始執行，但此時虛擬機器畫面停留在作業系統登入畫面，StartStub.bat 內容指令為等待 BOINC Manager 啟動完畢後再去執行 Stub，以便完成與 Installer 連線。

Installer 與 Stub 連線後，基本上都是由 Installer 傳送指令給 Stub 接收。首先 Installer 傳送查詢帳號指令給 Stub，以便檢查 BOINC 中是否已有該學生帳號，若沒有則傳送建立帳號指令給 Stub，之後再傳送一次檢查指令確定成功建立帳號，最後傳送加入 BOINC 計畫的指令，完成學生擔任志願者的工作，此時若 BOINC 計畫有工作產生，則志願者便會自動抓取工作檔進行運算。

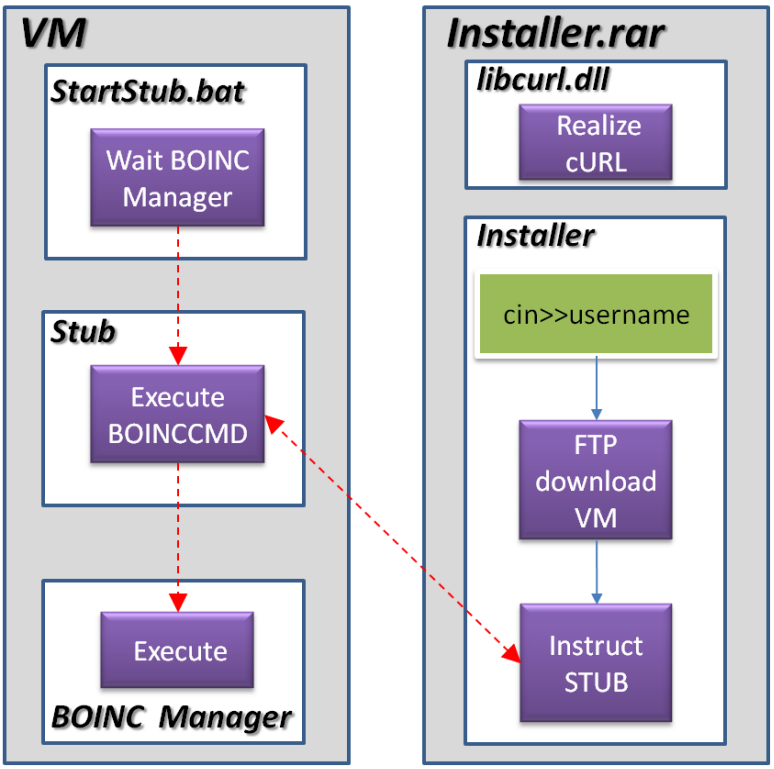
VM
StartStub.bat
Wait BOINC Manager
Stub
Execute BOINCCMD
Execute
BOINC Manager
Installer.rar
libcurl.dll
Realize cURL
Installer
cin>>username
FTP download VM
Instruct STUB

圖 7 客戶端運作模式

第五章　系統實作與討論

5.1　開發環境

BOINC 伺服器使用 Debian 6 作業系統。虛擬機器使用 VMware Workstation 7，其中的作業系統為 Windows XP，映像檔案大小約為 2.4GB，儲存於 FTP 上的壓縮檔大小約為 1.04GB。Installer、Stub 與 Controller 等元件皆以 C++程式語言來開發。元件之間的連線與溝通則使用 socket、cURL 等函式庫。

5.2　實作成果

圖 8 學生下載 Installer

使用者先登入數位學習網站下載 Installer，見圖 8，並執行圖 9 之程序：

1. Installer 要求使用者輸入帳號密碼，透過 cURL 連接到數位學習網站驗證使用者的帳號密碼是否正確。若正確則抓取該帳號的相關資料如 e-mail 地址，若該帳號無效則表示部署志願者失敗。
2. Installer 會下載虛擬機器到志願者的電腦。虛擬機器中已事先安裝 BOINC manager、Stub 與 StartStub.bat，且設定開機後會停留在登入畫面，但虛擬機器仍會去執行 StartStub.bat，等待 BOINC manager 啟動完畢後，StartStub.bat 再去執行 Stub。
3. 虛擬機器啟動後會經由 DHCP 連結到使用者電腦，而 Installer 啟動後會等待虛擬機器內的 Stub 經由 socket 來連線，並等待接收 Installer 傳送的指令。
4. Installer 將傳送指令給 Stub 執行 BOINC 相關的動作，例如使用與數位學習網站相同的帳號去配置志願者身分。

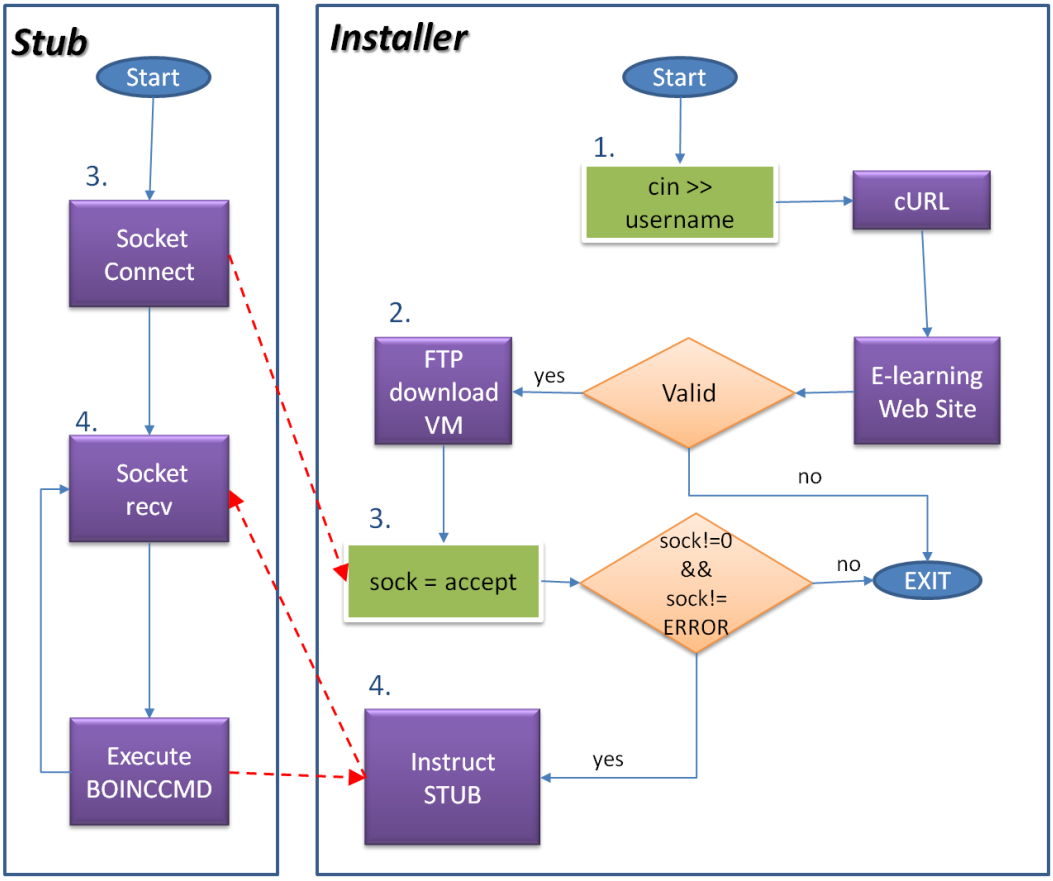

圖 9 Installer 與 Stub 之關係

圖 10 與圖 11 為 Installer 的執行畫面。圖 10 為學生輸入帳號密碼，經數位學習網站驗證確定身份後，詢問是否要下載虛擬機器，確認後便開始從 FTP 主機下載虛擬機器。下載完虛擬機器並解壓縮後，Installer 會自動搜尋 vmware 執行檔與虛擬機器映像檔路徑並執行，如圖 11 會自行啟動虛擬機器，等待 Stub 的連線與接收 Stub 傳送的訊息。

圖 12 為 Stub 的執行畫面。一旦 Stub 與 Installer 連線，Stub 就會收到來自 Installer 傳送的 boinccmd 指令訊息，同時去執行該指令，指令包含了建立 BOINC 帳號、查詢 BOINC 帳號與加入 BOINC 運算

計畫等。圖 13 為虛擬機器內該學生已加入 BOINC 專案，並可以開始幫忙運算工作。

一般而言，安裝在志工電腦上的虛擬機器是不允許被存取的，因此志願者看不到圖 12 與圖 13 的畫面，只會看到圖 14 的畫面。

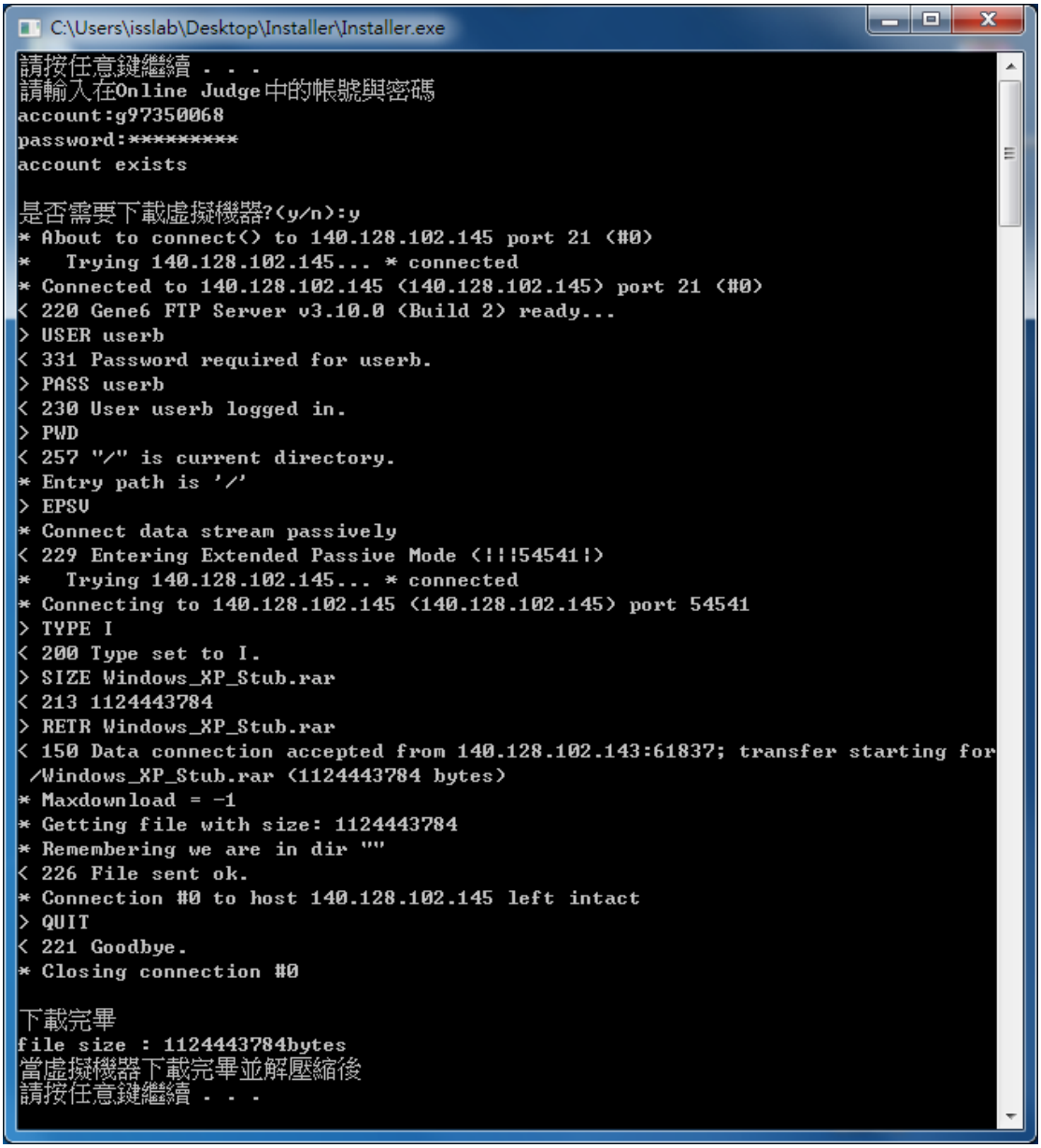

圖 10 Installer 執行畫面-驗證身份與下載虛擬機器

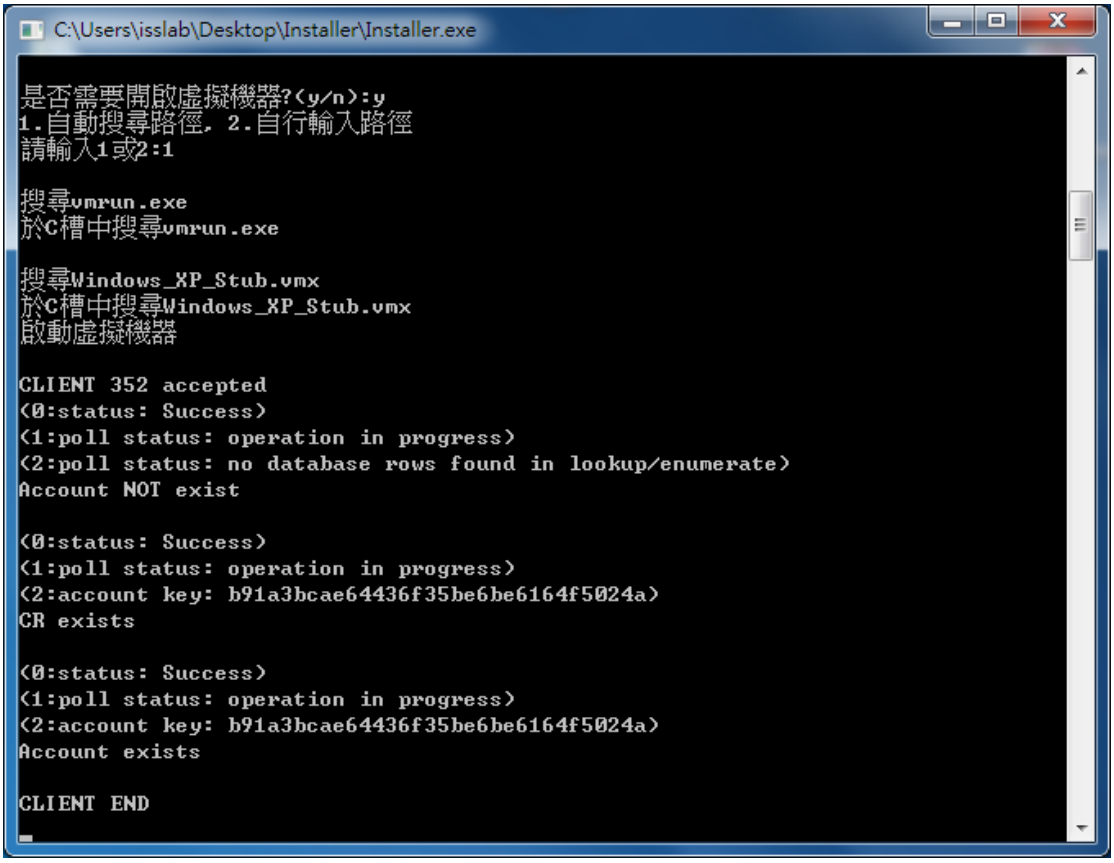

圖 11 Installer 執行畫面-啟動虛擬機器與接收回傳訊息

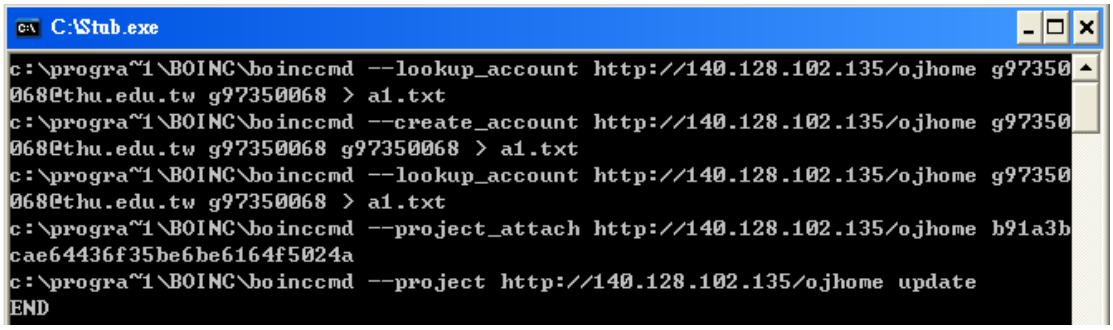

圖 12 Stub 執行畫面-接收並執行 boinccmd 指令

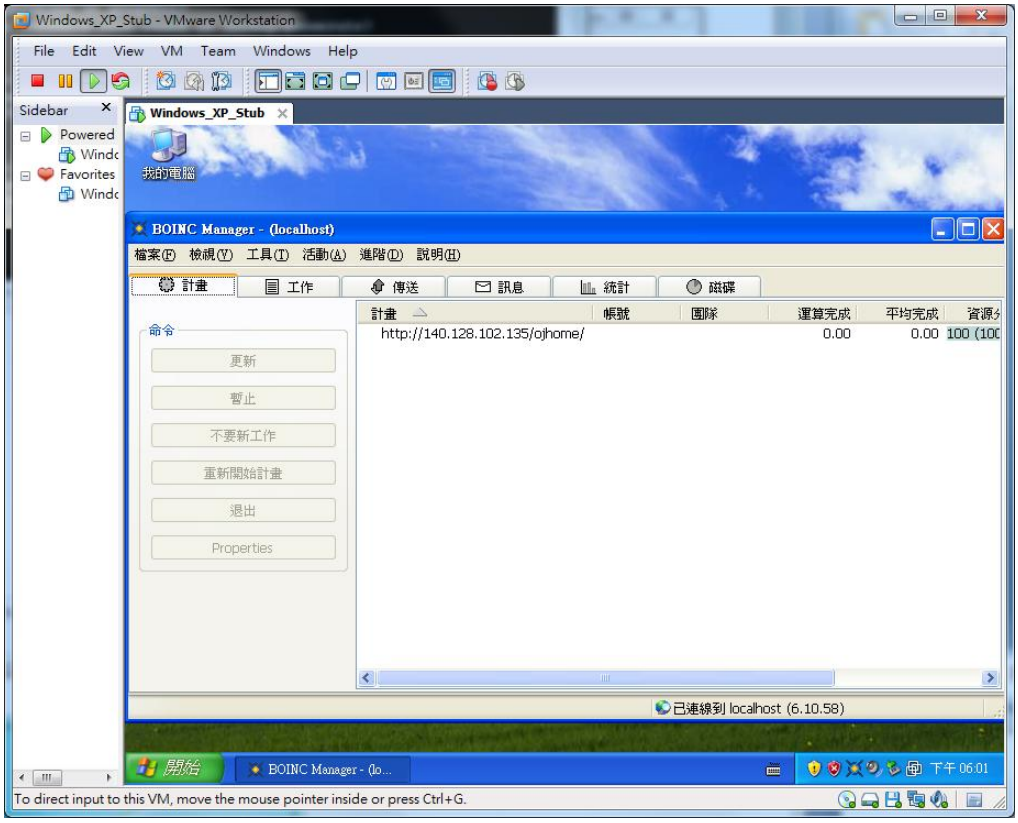

圖 13 成功加入 BOINC 運算計畫

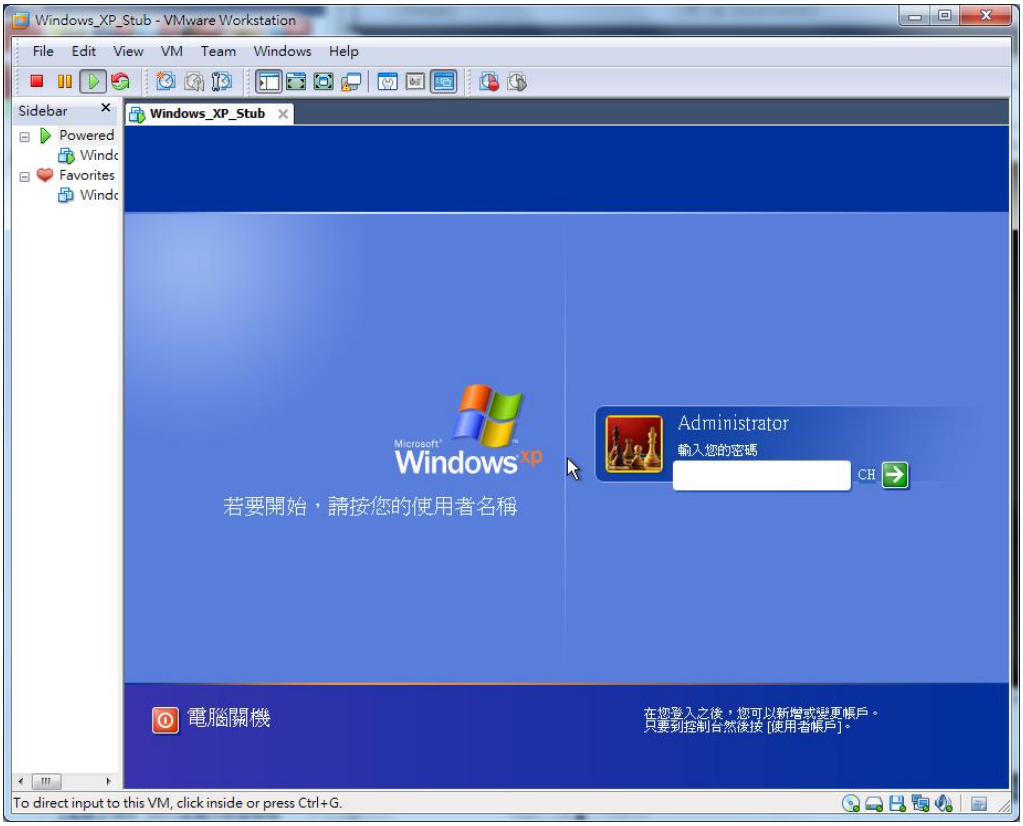

圖 14 登入畫面

若學生成功下載並開啟虛擬機器，虛擬機器內的 Stub 將會實現與 Installer 和 Controller 的連線。當 Stub 與 Controller 連線，Controller 會紀錄目前已連線的 Stub 個數，即已連線之志願者虛擬機器個數，並依照連線的先後順序給予編號，見圖 15。

Controller 僅能由管理者來操作使用，目前功能主要是針對志工電腦中虛擬機器的 BOINC manager 與 Stub 做管理，是一個能夠擴充 BOINC 功能的管理介面。

管理者可以執行 suspend 指定虛擬機器的編號讓志工電腦暫停運算；執行 resume 讓志工電腦從暫停運算狀態回到運算狀態；執行 getstate 讓管理者能夠知道志工電腦的帳號等狀態，如圖 16；執行 check 可以檢查目前已連線的志工電腦個數，如圖 17。

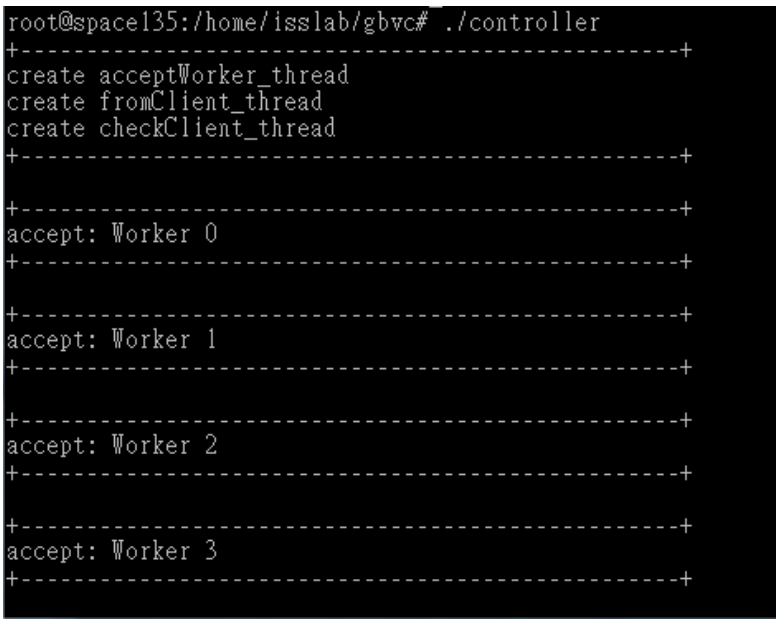

圖 15 Controller 執行畫面-接受 Stub 連線

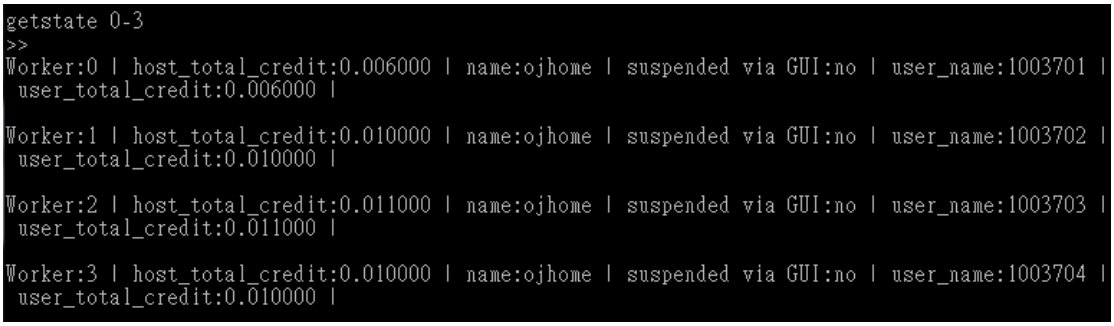

圖 16 Controller 執行畫面-執行 getstate 指令獲取 worker 狀態

圖 17 Controller 執行畫面-執行 check 指令檢視連線的志工電腦個數

5.3 討論

在部署志願者方面，由於虛擬機器的壓縮檔容量達 1GB，使用者下載及解壓縮檔案很耗時間，雖然網路速度快慢是一個因素，但未來仍需致力於虛擬機器容量的縮減。虛擬機器只需包含志願運算平台的必需元件如 BOINC manager，甚至連視窗或瀏覽器等介面都可省去，以減少虛擬機器的檔案大小。

第六章　結論

本論文實現了自動部署虛擬機器的技術，志願者使用 Installer 獲得數位學習網站中的帳號，並透過虛擬機器中 Stub 與 Installer 的連線來傳達指令給 BOINC Manager 執行，促發該虛擬機器進行 BOINC 帳號的建立、驗證與加入運算專案的動作，不僅減少學生擔任志願者時所要執行的額外動作，更增加了學生加入志願運算的意願。此外，管理者可以透過 Controller 來管理志工電腦，或讀取志工電腦的各種狀態，如帳號、是否已啟動運算、貢獻度等資訊。

虛擬機器的檔案大小大約為 2G 左右，在正常的使用情形下往往可以成長到 5G 以上，部署容量如此龐大的虛擬機器需要數個小時以上。因此，虛擬機器的精簡化將是往後重要的課題。

參考文獻

[1] 主從式架構 - wiki

http://zh.wikipedia.org/wiki/主從式架構, 2012

[2] UVa Online Judge

http://uva.onlinejudge.org/,2012

[3] David P. Anderson, “BOINC: A System for Public-Resource Computing and Storage” 5th IEEE/ACM International Workshop on Grid Computing, 2004.

[4] 伺服器虛擬化技術簡介

http://www.cc.ntu.edu.tw/chinese/epaper/0004/20080320_4012.htm,2012

[5] Hypervisor

http://www.dotblogs.com.tw/nel/archive/2010/06/01/15542.aspx,2012

[6] Berkeley Open Infrastructure for Network Computing (BOINC)

http://boinc.berkeley.edu/,2012

[7] BOINC - wiki

http://zh.wikipedia.org/wiki/BOINC,2012

[8] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D.Werthimer, ”SETI@home: An experiment in public-resource computing”. Communications of the ACM, November 2002

[9] Sangho Yi1, Emmanuel Jeannot1, Derrick Kondo1, David P. Anderson, “Towards Real-Time, Volunteer Distributed Computing”, 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2011.

[10] cURL and libcurl

http://curl.cs.pu.edu.tw/,2012

[11] Trilce Estrada, Michela Taufer, Kevin Reed, David P. Anderson, “EmBOINC: An emulator for performance analysis of BOINC projects”, IEEE International Symposium on Parallel&Distributed Processing , 2009.

[12] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Neri and O. Lodygensky, “Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid.”, FGCS Future Generation Computer Science, 2004.

[13] A. Chien, B. Calder, S. Elbert, and K. Bhatia, “Entropia: architecture and performance of an enterprise desktop grid system.”, Journal of Parallel and Distributed Computing, 2003

[14] David P. Anderson, Kevin Reed, “Celebrating Diversity in Volunteer Computing”, Hawaii International Conference on System Sciences, 2009

[15] Virtual machine http://en.wikipedia.org/wiki/Virtual_machine, 2012.

[16] Diogo Ferreira, Filipe Araujo, Patricio Domingues, “libboincexec: A generic virtualization approach for the BOINC middleware”, IEEE International Parallel & Distributed Processing Symposium, 2011.

[17] V. Pande et al., “Atomistic Protein Folding Simula-tions on the Submillisecond Time Scale Using World-wide Distributed Computing”, Biopolymers, 2003.

[18] D. Stainforth et al., ”Climateprediction.net: Design Principles for Public-Resource Modeling Research”, International Conference on Parallel and Distributed Computing Systems, 2002.

[19] Predictor@home
http://predictor.scripps.edu
[20] Klaus-Dieter Schewe et al, “A Conceptual View of Web-Based E-Learning Systems”, Education and Information Technologies, 2005.