以 Linux 系統為基礎之插件式雲端桌面架構

王國肇
財團法人國家實驗研究院國家高速網路與計算中心
c00wgt00@nchc.narl.org.tw

摘要

近年來隨著雲端技術的發展，各種應用程式逐漸雲端化，例如線上的文書編輯軟體、電子郵件的收發與雲端儲存空間等，琳瑯滿目，但這些應用大都是設計給一般社會大眾所使用的，而研究與計算導向的雲端服務就比較缺乏，如果是學術界的研究人員，就比較無法立即從雲端技術獲得較大的幫助。本文將探討一個適用於 Linux 系統的插件式雲端桌面架構，實作一個可立即隨插即用的混合式雲端服務，這樣的架構可以讓使用者輕易的將自己沒有雲端使用者介面的 Linux 設備與雲端服務的桌面環境結合，在做最少的更動之下享有雲端桌面的便利性，這個對於非資訊領域的研究者會有幫助，尤其對於 Linux 系統不熟悉的使用者助益更顯著。

關鍵詞：雲端運算、雲端軟體服務。

Abstract

In recent years with the development of cloud technology, more and more cloud-based applications have been developed, such as online text editor, email applications and cloud storage. However, these applications are mostly designed for general public use, and research-oriented cloud computing services are relatively lacking. In this paper, we propose a plugable cloud desktop architecture and implement a plug-and-play service sytem for Linux operating system based on this architecture. This system allows users to add a desktop environment to their Linux OS with minimal changes, and will be helpful for the researchers who are not familiar with Linux systems.
Keywords: cloud

1. 前言

在許多研究領域(如氣象與工程等)本身有非常大量的計算模擬需求，許多科學界的大型模擬軟體都是設計在 Linux 系統下執行的，因此長久以來各領域已經存在許多計算導向之 Linux 工作站或叢集電腦，因為這些設備通常體積龐大、噪音也很高，多半都會放置在專用的機房內，傳統上在使用這些系統都是透過 SSH 連線登入之後，以指令來操作所有要執行的動作，藉此執行科學計算的模擬程式。

但學習 Linux 系統指令與相關的操作技巧的門檻較高，對於非資訊背景的新進研究者而言是一大負擔，甚至有需多需要大量運算之研究者，因為無法突破 Linux 的使用門檻，而導致其研究發展停滯，甚至放棄這類計算導向研究的發展。

近年來資訊科技與網際網路的發展迅速，雲端運算（cloud computing）的概念已經成為現在網路服務的主軸，而所謂的雲端運算即是指藉著網際網路的連線，讓使用者可以隨時隨地使用伺服器上的服務與各種資源，而且透過各種先進的網頁技術（如 HTML5 等），許多以前在本機執行的應用程式，現在都可以透過網頁的方式執行，例如 Google 雲端硬碟服務，即可代替以往的 Office 軟體來製作簡報與試算表、進行文書排版等。

有鑒於此，本文提出一個類似與混合式雲端（hybrid cloud）的系統架構，藉由一些雲端網頁相關技術的整合，建構出一個插件式的雲端桌面，可以讓一般的 Linux 系統在幾乎不需要更動任何設定或安裝任何軟體之下，享有雲端運算的便利性，讓管理者不需要考慮整體系統與軟體更新時的相容性問題，同時也降低了一般科學研究者在使用 Linux 系統時所面臨的指令操控門檻，甚至一般的使用者也可以自行套用這樣的雲端桌面，讓自己的 Linux 系統享有這樣的功能，而不需要尋求技術人員的協助。

2. 文獻探討

在過去有許多雲端桌面的實作，可以讓使用者遠端操控遠端的 Linux 桌面，例如原始的 XWindow[1] 架構本身就有 client 與 server 的設計，可支援遠端顯示的功能，但因為其所需要的網路極大，通常只能在區域網路中使用。

而專門為遠端桌面所設計的 VNC 傳輸協定，雖然大幅降低網路的頻寬需求，但是由於後來顯示器的解析度都非常高，在傳輸較高品質的畫面時，還是會受到網路的頻寬限制，再加上這類的傳輸方式都會需要專用的 client 程式，因此在使用上也不是很方便。而除了 VNC 之外，亦有許多類似的傳輸架構可將遠端電腦的桌面透過網路讓使用者控制，例如 RDP、NX[2]、SPICE[3]等，以上這些都是以影像為基礎的遠端桌面架構，同樣都會受到網路頻寬的影響。

後來因為 HTML5 等新興網頁技術的發展，出現了完全以網頁技術實作 VNC client 的 noVNC[4]，以及透過代理伺服器將 VNC 與 RDP 整合的架構

Guacamle[5]，這些都改善傳統上要安裝 client 的問題，但是由於這些架構都是將伺服器上的影像傳輸到使用者端，影像是非常大的資料，因此網路頻寬依然是瓶頸所在。

除了將以影像為基礎的桌面架構之外，亦有許多直接以網頁技術實作的網頁作業系統，在瀏覽器中呈現一個虛擬的桌面，使用者可以藉著這樣的桌面透過網路操作遠端的電腦，例如 eyeOS[6]是以 PHP 撰寫的網頁作業系統，開發者可以依照自己的需求撰寫其中的應用程式，並跟後端伺服器上的程式連結，以達到遠端操控伺服器的功能。

在計算與研究導向的雲端架構上，亦有一些值得探討的實作，例如 RStudio[7]是網頁的方式改寫傳統的 R[8]計算環境介面，透過自行實作的伺服器介接程式，讓使用者可以在網頁上進行 R 程式的編寫與執行，甚至在網頁上即時繪圖。

以上所討論到的相關技術中，其所操控的遠端電腦都是伺服器本身，或是需要額外的繁雜設定才能加入外部的計算主機，本文將針對需要動態加入被操控主機的情況，設計一個插件式的雲端桌面架構。

3. 背景技術

本章節針對本雲端桌面架構中會使用到的各種相關技術分別做介紹，並說明選用這些技術的原因所在。

3.1 Java Servlet

Java Servlet 是一個可以為伺服器增添功能的 Java 類別，它可以被動態載入，以延伸伺服器的功能，但由於它是在 Java 虛擬機器上執行的，相較於其他的技術，它兼具安全性與可攜性的優點，而且他也是一個很普遍且跨平台的標準，當開發者寫好一個 Servlet 時，只要是符合 Servlet 標準的伺服器都可以直接載入使用。

Servlet 的功能類似傳統上的 CGI（例如 PHP 與 Perl 等），但是它與傳統的 CGI 在設計上有明顯的不同，一般的 CGI 收到使用者發出的請求時，會為每一個請求分別建立獨立的行程來處理，雖然有一些改良式的 CGI 會比較有效率的控管行程，但是基本上每一個請求在處理時都會是在不同的行程中來處理，互相之間並沒有交集，而處理完成後，該行程就會結束或是由系統收回，而 Servlet 則不同，它在被伺服器載入之後就直接被放置在伺服器的行程中，當接收到使用者的請求時，會產生不同的執行緒來處理不同的請求，這樣的方式不但更有效率，又具有彈性，而且可以很容易與伺服器溝通，而在處理完使用者的請求之後，它還是會繼續駐留在記憶體中，這個特性也稱為 instance 的持續性（persistence），而也因為這個特性，讓我們可以在 Servlet 上面開啟一個跟外部伺服器的持續性連線，並且保證連線不會因為行程的結束無被迫中斷。

另一方面，由於 Servlet 是以 Java 語言開發的，所以它可以使用 Java API 核心的所有功能，甚至可以使用外部的 Java 函式庫，當需要設計一個功能比較負責雜的伺服器程式時，Servlet 跟傳統 CGI 比較之下會比較占優勢。

3.2 Google Web Toolkit

Google Web Toolkit[9]（後續以 GWT 簡稱之）是 Google 為了發展大型的網頁應用程式而設計的開發工具，其設計的目標主要是為了讓開發者可以在不需要熟悉太多較低階的網頁技術之下，即可建立以網頁為基礎的大型雲端應用程式，其實作方式是透過編譯器將 Java 程式碼編譯成為 JavaScript，讓應用程式開發者可以 Java 語言來設計程式，藉著 Java 語言的物件導向特性，快速建置與維護複雜但高效能的大型應用程式。

在 GWT 的架構中所有的程式都是使用 Java 語言來開發，透過 GWT 編譯器編譯成 JavaScript 後放在瀏覽器中執行，而如果是要在伺服器上執行的程式，則會以一般 Java 的編譯器編譯成 Java Servlet 放在伺服器的 JVM 中執行。為了要使伺服器與瀏覽器之間的資料可以及時的傳遞，GWT 以 AJAX 技術為基礎建立 GWT RPC 的溝通機制，讓 Java 物件得以在伺服器與瀏覽器之間直接傳遞與運用，開發者可以不需要自行處理 JavaScript 與 Java 物件之間的轉換工作。

使用 GWT 所開發出來的程式，最後都可以被包裝成一個標準的 WAR 部署檔案， 其包含該網頁應用程式所有的元件，在部署該程式時，只需要將這個檔案放進符合 Servlet 標準的應用伺服器中即可，因此使用 GWT 所開發出來的應用程式不但可跨平台、跨伺服器，在安裝上更是非常方便。

一般的 GWT 所提供的 GUI 元件很有限，因此出現了許多針對 GWT 所開發的 GUI 函式庫，在開發專業的大型程式時使用這些函式庫可以大幅縮短開發時間，常見的函式庫有 Ext GWT(GXT)[10]、SmartGWT[11]與 GWT-Ext[12]等。

4. 插件式雲端桌面架構

4.1 抽象架構

一般來說，雲端運算的類型可以依據設備提供給使用者使用或管理的層級來分為三大類[13]：

1. 軟體即服務（SaaS）：讓使用者可以不需要安裝任何軟體的情況下，只需要連上網路即可使用安裝於伺服器上的應用軟體，在這樣的架構下所有計算環境的軟硬體設備都是由系統管理者所提供與管理的，方便性最高，但缺乏彈性與自由度，例如一般的 Facebook 即是此類的服務。

2. 平台即服務（PaaS）：提供一個標準化的平台，讓使用者可以在這樣的標準之下，開發自己的應用程式並部署至雲端的伺服器中，這樣的方式讓使用者有較大的彈性可以設計客制化的程式，能夠適用於更多不同的應用領域，例如 Windows Azure 與 Google App Engine。

3. 基礎設施即服務（IaaS）：雲端服務的提供者將處理器、儲存空間、網路等基礎設施都開放給使用者使用與管理，雖然使用者無法實際接觸到實體設備，但是他可以掌控作業系統、儲存設備、網路與應用程式的部署組態，是自由度最高的一種服務模式，但其設定過程也最繁復，使用門檻也最高。

若是以雲端資源的部署來分，則可分為：

1. 私有式雲端（private cloud）：不對外開放，只提供給自己單位內部使用的雲端服務，由單位自行或委外管理。

2. 社群式雲端（community cloud）：只開放給限定的社群使用，由社群內部自行維護。

3. 開放式雲端（public cloud）：開放所有人使用的雲端服務，由雲端服務提供者負責維護。

4. 混合式雲端（hybrid cloud）：結合兩中以上的雲端類型，讓不同的雲端服務得以協同運作提供整合性的服務。

在以往的雲端服務模式中，在概念上不管是哪一種服務模式，所有的實體設備都是由雲端提供者在維護，使用者很難將自己現有的 Linux 系統直接放置在雲端上使用，因此我們設計了一個可以讓使用者將自己的運算資源雲端化的架構（如圖 1 所示），右半邊是傳統的雲端服務架構，但我們將中間層（Middleware）與使用者介面（Presentation）擴充之後，允許使用者將自己的運算資源（圖 1 中的左半部）與其整合，如此一來使用者也可以在雲端上使用自己所提供的運算資源與應用軟體。

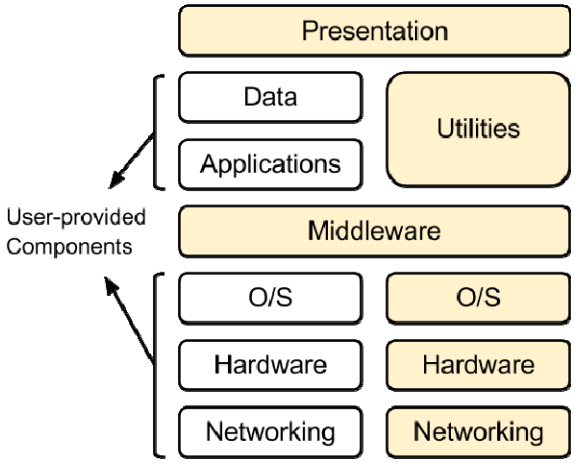

圖 1 插件式雲端桌面抽象架構圖

這樣的模式類似結合私有式與公開式的混合式雲端，但不同的是在這個架構中使用者不需要具備私有雲端的建置技術與相關人力成本，即可得到一個整合性的服務，相對上使用較方便，抽換實體設備的彈性也較大。

為了讓各種使用者的 Linux 系統皆可立即使用這樣的雲端架構，並且避免相容性的問題，在中間層與使用者 Linux 系統中間的介接部分，我們選擇使用最傳統的 SSH 協定配合 Linux Shell 來將使用者的 Linux 系統與雲端桌面作串接，因為 SSH 是各種 Linux 甚至 UNIX 都通用的協定，歷年來版本變化都不大，相容性問題很小，而且由於它的加密功能可以確保雲端桌面與自己的 Linux 之間的連線非常安全，縱使中間跨越很多外部的網路也不用擔心資料外洩，而在操控使用者的 Linux 系統的部分，則是透過 Linux 的 Shell 來操作，Shell 層級是介於作業系統與應用程式之間，在 Linux 幾乎中所有的應用程式都是透過 Shell 來執行的，所以透過這樣的方式也可以足以操控整個系統進行各種科學運算。

4.2 硬體架構

這裡我們選擇以一台或多台應用伺服器（App Server）與一台或多台資料庫伺服器（Database）來實作這樣的架構（如圖 2 所示），應用伺服器負責主要的網頁呈現與連線的工作，資料庫伺服器則負責使用者帳號與主機的控管。使用者可以將自己計算用的工作站（User's Workstation）連上網路之後，登入應用伺服器並選擇自己工作站的位址，即可透過雲端桌面的介面操控自己的工作站。

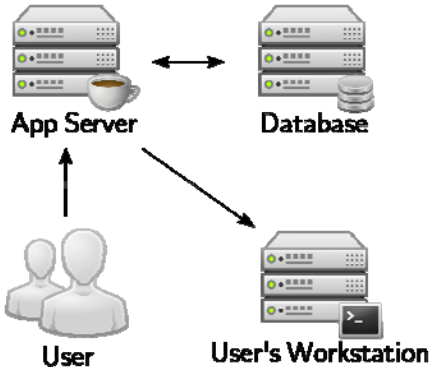

圖 2 硬體架構

由於應用伺服器與使用者工作站之間是使用 SSH 與一般的 Shell 在溝通，而這種架構同樣也適用於其他的 UNIX 相關的系統，例如 FreeBSD 或 Solaris 等，也就是說只要使用者有權限以 SSH 登入的主機，都可以直接套用此桌面來使用。

4.3 軟體架構

本雲端桌面在軟體的架構上主要可分為瀏覽器端的網頁應用程式（Web Application）與應用伺服器端的連線仲介程式（即 Servlet）兩大部分，另外再加上使用者自行提供的 Linux 計算主機。網頁應用程式主要用來產生所有圖形化的使用者介面（GUI），負責將各種 Linux 指令的操作、輸入與輸出轉譯成圖形化介面，讓使用者直接使用很直覺的方式操控後端主機，在應用伺服器端的連線仲介程式部分，其主要的工作是結合資料庫控管所有使用者與主機的連線，另外也負責使用者瀏覽器與外部 Linux 主機之間的資料傳遞等。而使用者的 Linux 主機則負責實際程式的執行與運算。

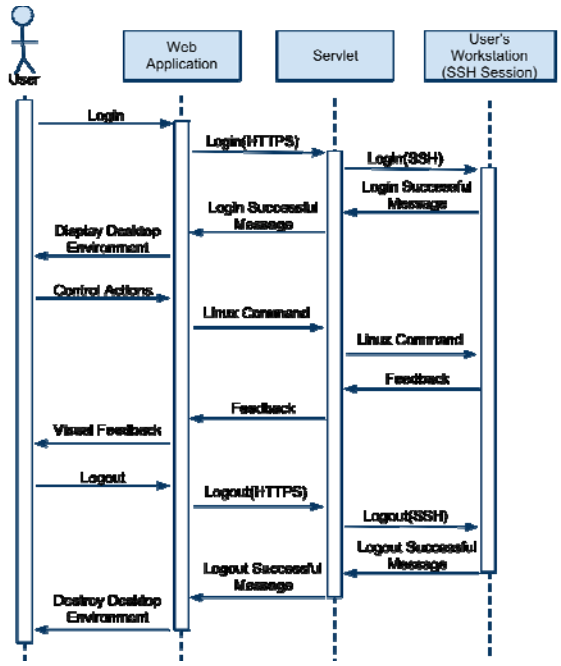

圖 3 軟體架構循序圖

使用者在開啟瀏覽器之後，先以網頁的方式登入應用伺服器（如圖 3 所示），其中的 Servlet 會依照資料庫中的帳號與主機權限控管所有的使用者與計算主機的連線，如果使用者與主機皆在被允許使用的名單之內，即可依據使用者所輸入的 Linux 主機之帳號密碼來開啟一個 SSH 連線，在開啟與外部 Linux 主機之間的連線之後，使用者就可以開始使用桌面來操控該 Linux 主機了，在圖形化的桌面環境中，使用者可以使用滑鼠點選欲執行的程式，此時網頁應用程式會將圖形的操作轉譯成實際的 Linux 指令傳送至應用伺服器端，由 Servlet 轉送至外部的 Linux 主機上執行，隨後再將執行完成後的輸出透過 Servlet 送回使用者端的桌面環境中，轉換成圖形化的輸出後再顯示給使用者。在這樣的情況下，由於在網路上傳輸的資料都不是實際的桌面影像，因此其反應速度會比較快，不會受到網路頻寬的限制。另一方面，所有應用伺服器與瀏覽器之間的資料傳輸都是透過 GWT RPC 協定，所有的 Java 物件都會自動在必要時做轉換，讓瀏覽器的 JavaScript 與伺服器上的 Java 程式可以直接溝通，在開發上也會更有效率。

由於 Servlet 的持續性，當與外部的 SSH 連線建立之後，即可將此連線保存在 Servlet 之中，每一次使用者下達指令時都可以使用同一個 SSH 連線而不需要重新建立一個新的，這樣可以提高整體的執行效率，甚至在使用者端暫時關閉瀏覽器之後再打開，同樣可以使用同一個 SSH 連線繼續操控 Linux 主機。

在安全性上，我們使用資料庫控管使用者與主機，避免被網路上的有心人士當成跳板進而攻擊外部的主機，另外在連線上瀏覽器與應用伺服器之間使用 HTTPS 加密連線，而應用伺服器與外部的 Linux 之間則是使用 SSH 協定，因此在所有的傳輸過程資料都是加密的狀態，可確保一定的安全性。

這樣的架構除了執行簡單的指令之外，亦可進行檔案的傳輸與類似本機操作的各種動作，例如下載檔案的動作我們就可以直接在 SSH 連線中將 Linux 主機上的檔案內容直接傾印出來送給 Servlet 之後，再加上適當的 HTTP 標頭，即可直接傳送給瀏覽器，進行檔案下載的動作，這樣的方式也可以讓使用者在不需要安裝任何 FTP 軟體之下，下載任何 Linux 主機上的檔案。

除了直接下載檔案之外，亦可依據需求修改 HTTP 標頭，讓一些瀏覽器直接支援的檔案格式（例如圖檔、PDF 檔等）可以直接在瀏覽器中的桌面上顯示，這樣使用上會更直覺，更接近本機使用的操作習慣。

除了直接操作 Linux 主機的動作之外，其實有很多桌面操作的動作是可以在瀏覽器端完成的，例如編輯簡單的文字檔案時，使用者可以開啟遠端 Linux 主機中的文字檔，將其內容下載至瀏覽器內，使用雲端桌面上的文字編輯器進行編輯，此時所有的編輯動作（例如打字等）都可以不需要透過網路，由瀏覽器本身直接反應給使用者，直到需要存檔時再透過網路將檔案內容傳送至 Linux 主機上並進行儲存，這樣的方式可以讓檔案在編輯時完全不會受到網路延遲的影響，跟使用本機的原生文字編輯器幾乎相同，只有在開啟檔案或儲存檔案時才會有影響。

5. 實作

實作上我們使用 GWT 的 Eclipse 整合環境開發，並使用 JSch[14]與 Connector/J[15]兩套 Java 函式庫來建立 SSH 與 MySQL 的連線，配合 GXT 函式庫後來實作整個插件式雲端桌面。

圖 4 為雲端桌面之登入畫面，在使用者輸入帳號、密碼與登入主機的位址之後，系統可以根據這些資訊做控管，如果是被允許使用的使用者與主

機，則為其開啟遠端的 SSH 連線，並顯示桌面畫面。

目前我們已經在這個桌面環境中實作了幾個簡單應用程式（如圖 5 所示），例如文字編輯器可以讓使用者開啟遠端 Linux 主機上的文字檔案，編輯簡單的文字內容，編輯完成後再儲存回遠端的 Linux 主機中。而檔案總管則是將 Linux 主機中的目錄結構以樹狀的結構呈現出來，使用者可以很方便的瀏覽各目錄中的檔案，在檔案上按下滑鼠右鍵即可開啟選單，使用者可以下載後儲存在自己的硬碟（如圖 6 所示），而如果該檔案的檔案類型剛好是瀏覽器支援的類型，則可以針對該類型在雲端桌面環境中實作一個檔案檢視器，然後藉由該檢視器直接開啟，例如顯示遠端 Linux 主機的圖檔就是常見的例子（如圖 7 所示）。

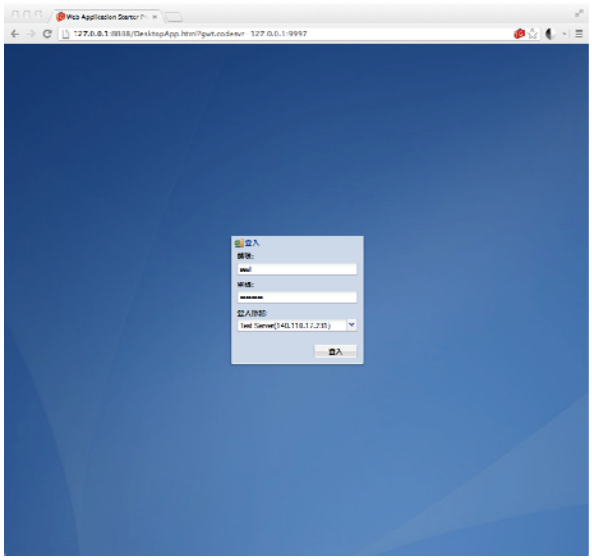

圖 4 登入畫面

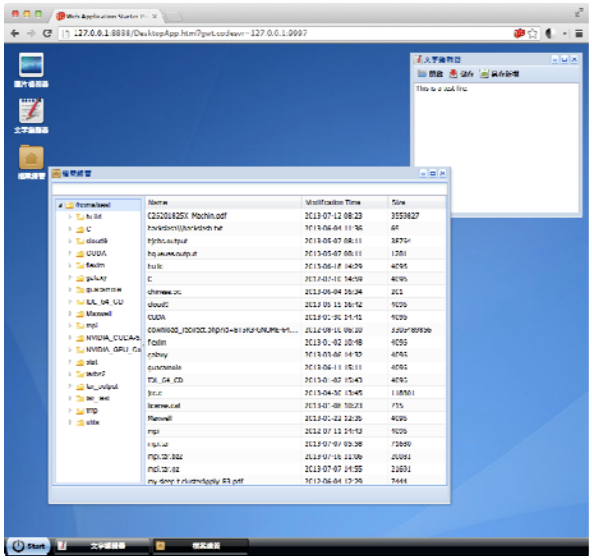

圖 5 檔案總管與文字編輯器

這些雲端桌面上的應用程式在底層都是靠著 Linux 指令在操控遠端的 Linux 主機，因此只要是一般的 UNIX-like 的系統，都可以直接套用這樣的架構來使用，伺服器不需要針對個別的系統作設定，更不需要在這些 Linux 或 UNIX 系統上安裝任何軟體或作任何變更。再者，由於所有的程式皆是在 GWT 的架構下整合開發，因此當開發完成之後，所有的元件皆可包裝成一個 WAR 部署檔，這樣的方式可以讓系統管理者非常容易安裝，而且具有高度可攜性，甚至要讓使用者自行安裝在自己的 Linux 系統中都可以，彈性非常大。

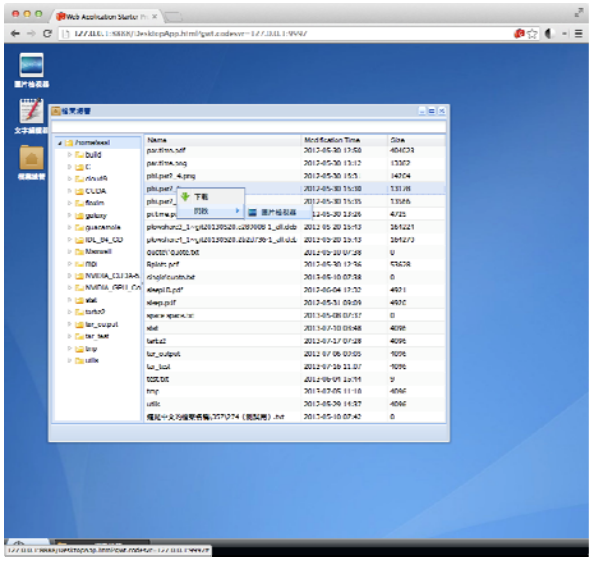

圖 6 檔案總管

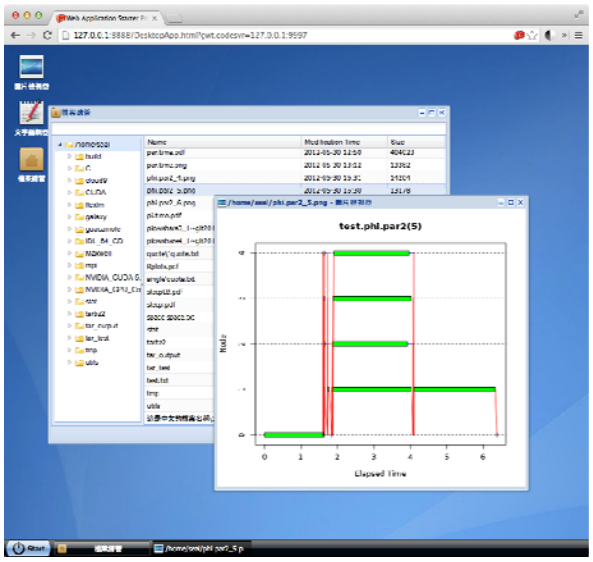

圖 7 圖片檢視器

6. 結論與未來展望

本研究提出一個插件式的雲端桌面架構，可結合任何的 UNIX 或 Linux 系統，讓一般 Linux 使用者立即可以使用雲端化的桌面，並且不需要更動任

何系統上的設定或安裝任何軟體。除此之外，這樣的架構也可以作為簡單檔案傳輸工具，進行檔案下載的動作，對於原本使用 Linux 指令模式從事科學研究的使用者有很大的幫助。

由於本架構尚在研究階段，因此實作出來的應用程式不多，但基本上因為所有的 Linux 系統上的應用程式都是透過 Shell 來執行的，因此理論上大部分的應用程式都可以在此架構下實作出來，而有 3D 繪圖需求的程式也可以配合新的 WebGL 網頁技術來呈現，在這方面尚有很大的發展空間。

參考文獻

[1] Chris Tyler, “X Power Tools”, O'Reilly Media, 2007

[2] NoMachine, http://www.nomachine.com/

[3] SPICE Protocol, http://spice-space.org/docs/spice_protocol.pdf

[4] VNC Client using HTML5 (Websocket, Canvas). http://kanaka.github.com/noVNC/

[5] HTML5 Client-less Remote Desktop. http://guac-dev.org/

[6] eyeOS: The unified workspace, http://www.eyeos.com/

[7] RStudio (2012). RStudio: Integrated development environment for R (Version 0.96.122) [Computer software]. Boston, MA. Retrieved May 20, 2012.

[8] R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing; 2013.

[9] Google Web Toolkit, http://www.gwtproject.org/

[10] Sencha GXT: Application Framework for Google Web Toolkit, http://www.sencha.com/products/gxt

[11] Smart GWT: GWT API's for SmartClient, https://code.google.com/p/smartgwt/

[12] GWT-Ext Widget Library, https://code.google.com/p/gwt-ext/

[13] "The NIST Definition of Cloud Computing". National Institute of Standards and Technology. Retrieved 24 July 2011.

[14] JSch: Java Secure Channel, http://www.jcraft.com/jsch/

[15] MySQL Connectors, http://www.mysql.com/products/connector/

[16] WebGL(Web Graphics Library), http://en.wikipedia.org/wiki/WebGL