新世代網路運算軟體平台

林盈達、陳枝佑

國立交通大學資訊科學系

新竹市大學路 1001 號

TEL: (03) 5712121 EXT. 56611

EMAIL: ydlin@cis.nctu.edu.tw

摘要

隨著 WWW 的蓬勃發展，在這個表象之下隱藏的是另一波的資訊技術革命，新的網路運算 (Network-Centric Computing) 技術將更加充分發揮 WWW 的威力，改寫電腦軟硬體架構及人們使用電腦的方式。我們可預見的是，將來會有更多的系統以 WebX （ "X" 可能是任何東西）的形式出現。透過網路運算技術使得電腦使用者不再需要在自己的個人電腦上面安裝許多肥大的軟體，相反的，客戶端 (client) 只需要很精簡的作業系統以及瀏覽器 (browser) ，則所需要的不論資料與應用都來自於網路上。在本篇文章中，我們先看看整個軟體平台式怎麼演變到未來的網路運算，接下來就針對網路運算所使用到的 Web 、 Java 以及 Object 技術做一些介紹，同時我們發現目前有兩大勢力在角逐分散式物件技術的地位，也就是 CORBA 以及 ActiveX 架構，我們也將對這兩種技術作一些討論。

一、軟體平台的演變三部曲

在了解網路運算之前，我們須先回顧過去電腦的運算方式。在個人電腦尚未在 1980 年代出現之前，大型主機是唯一的玩家，在那個時代，使用者透過毫無處理與儲存能力的終端機 (terminal) 將工作交給主機，所有的處理與儲存工作都集中交給主機負責，主機與主機之間雖有長距離的網路連線(註：1969 年 Internet 的前身 —— ARPAnet 即已誕生），但只用來作電子郵件及檔案傳輸 (emial 及 ftp)。個人電腦在 1980 年代推出以後，使用者面對的不再只是一個只是鍵盤、顯示器及 RS-232 介面的終端機，而是一個具有處理器、軟硬碟、記憶體的電腦，個人電腦的價格低到每個家庭都有能力擁有，功能也大到讓許多公司捨棄大型主機（或讓許多公司開始願意電腦化），配合這個時期開始成熟的區域網路（如 Ethernet 乙太網路），大部分的處理與儲存工作均可透過網路分散至眾多個人電腦，大型主機只負責處理共同的工作，如公用檔案儲存、電子郵件處理、大型工作處理，甚至由高階個人電腦取代大型主機。

相對於大型主機時代中將處理與儲存工作集中交給主機，個人電腦時代中使用者必須自己安裝各種軟體與管理各種電腦上的資源，才能發揮系統功能的極致，即使是要使用各種主從架構 (client-server)

應用，至少使用者也要在自己的個人電腦上灌用戶端（client）軟體，才能使用某伺服器（server）上的服務，況且還要看使用者的個人電腦是跑那一種作業系統，跑 Win95 或 OS/2 或 UNIX 是需要裝不一樣版本的用戶端軟體。當這些應用軟體愈來愈多、愈來愈複雜的時候，使用者就愈來愈頭大（或愈來愈厲害），同時，軟體開發者也須提供各種版本。

既然如此，我們何不將這些應用軟體全部交給網路上的主機管理？（不管這個主機是大型主機或高階個人電腦）亦即將所有資源放在網路上，由提供該資源的人管理這些東西，而免除使用者須事先安裝（pre-install）該軟體的困擾，使用者要使用該軟體時，即由網路上的主機伺服器"抓"（download）下來在自己機器上執行，而伺服器軟體則在該主機上或該主機所指定的機器上執行。在這種環境下，我們將儲存的工作完全交給網路上的主機，而處理的工作則保持不變，由用戶端及網路上的主機分擔。這種以網路爲中心的運算方式便是網路運算的基本想法，但從網路抓下來的軟體馬上就能在各種不同的作業系統上執行嗎？當然要具備跨平台的特性必須這個軟體本身能在不同平台上執行，由於不同作業系統及機器的可執行檔（executable file）格式及內容不同，顯然我們無法期望下載的可執行檔能跨平台，下載的東西必須接近原始程式碼，並以解譯（interpretation）的方式執行。

我們目前仍處於個人電腦運算時期，尚未真正進入網路運算時間，圖一總結三種運算時期（或軟體平台）的差異。

時期	特性	儲存地點	處理地點
大型主機 （mainframe）	終端機 集中式管理	大型主機	大型主機
個人電腦 （PC）	功能強大的個人電腦 分散式管理	個人電腦	個人電腦
網路 （network）	無磁碟具簡單作業系統的電腦 網路化集中管理 跨平台	網路主機	網路主機 用戶端電腦

圖一：三種運算時期的比較

二、網路運算技術

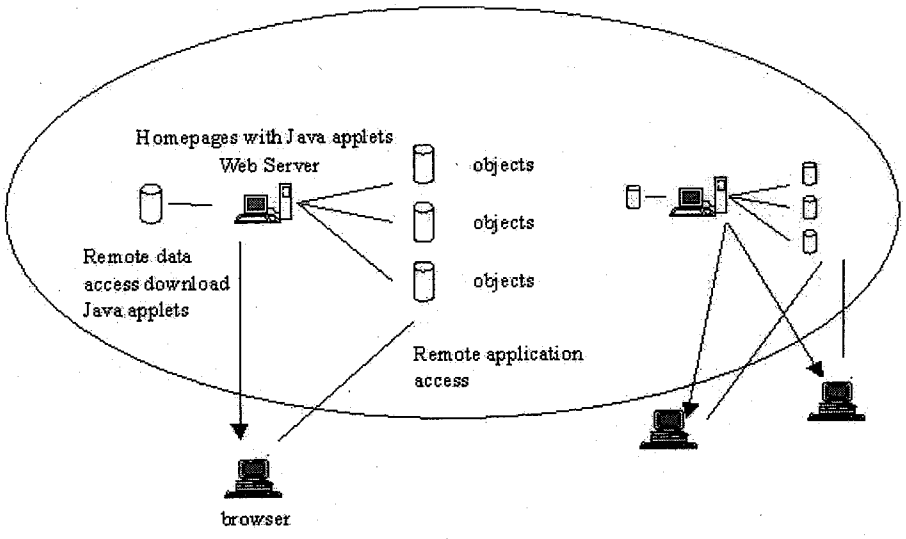
上一節提到了網路運算的概念，但要用那些技術實現這個概念呢？ WWW 提供了相當好的基礎架構，除了多媒體資料與介面，它的超鏈結（hyperlink）將相關網站及文件組織起來，使用者可透過瀏覽器程式（browser），輕易取得資料及應用（remote data access 及 remote application access）。

如上一節所述，網路運算中下載的軟體須具備跨平台的特性，且以解譯的方式執行， Sun MicroSystem 公司所提出的 Java 語言便是用來解決這個問題， Java 程式以 bytecode 儲存及下載至用戶端解譯

執行，它可以 Java Applet 的方式內嵌（embed）於 WWW 的網頁（homepage）中，用戶端的瀏覽器發現網頁中有 Java Applet ，即由網站下載該 Java 程式，並開始執行， Java 程式除了增加網頁的動態功能，最重要的是它的跨平台特性及較小的程式，但它的缺點是解譯速度較慢， Sun MicroSystem 針對這個問題採用了兩種解決之道，其一是 JIT (Just-In-Time) compiler [1]，將下載的 Java 程式即時編譯成用戶端機器的可執行程式然後執行，當然羊毛出在羊身上，這個 JIT compiler 是取決於用戶端平台（機器及作業系統）的種類， Sun MicroSystem 必須提供足夠多的 JIT 版本才能解決大部份用戶的需求，另一個解決之道是提供能直接“執行” Java 程式的處理器，這個處理器即 PicoJava ，它的處理速度是 JIT 的 5 倍，且程式大小爲 JIT 編譯過後可執行程式的 1/3 ， Sun MicroSystem 準備陸續推出 microJava 及 UltraJava 處理器 [2]。

在網路運算的環境中， Web 站本身將是各種應用的 SAP (Service Access Point) 服務存取點，而不一定是真正的應用執行點，由於現在大部分的 WWW 應用是資料存取 (data access)，如電子購物、資料查詢瀏覽，而不是應用存取 (application access)，如隨選視訊、視訊會議，所以 Web 站本身並不需要做太多的處理，大部分只是將一個 html 檔案傳給用戶端，若是額外的處理，則以 CGI (Common Gateway Interface) 程式的方式“掛”在 Web 站上執行。但當有愈來愈多的應用是需要 Web 站做大量處理工作時，很顯然地，我們不能將它們掛在 Web 站上，必須將它們放在真正的伺服器上，至於要放在那裡？如何找到它們？及它們將以何種形式存在於網路上？分散式物件導向技術正可以提供答案，在 CORBA (Common Object Request Broker Architecture) 架構中，軟體可被切成許多分散式的物件，物件可透過 ORB (Object Request Broker) 找到其他物件並與其交談，用戶端也可透過 ORB 找到要找的物件並呼叫該物件內的方法 (method)，以執行一件工作。因此，各種應用服務的伺服器軟體便可以 object 形式存放於網路主機上，而不是在 Web 站上，用戶端的 Java 程式便可透過 Java ORB 呼叫伺服器 object ，要求提供服務。

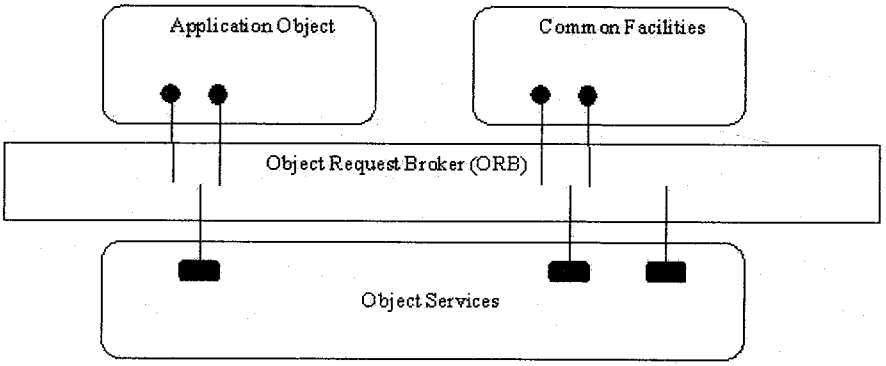
我們已經看到了網路運算所需採用的技術，包括 WWW 、 Java 及 object ，但它們在整個網路運算架構中的位置式如何？圖二總結各個技術的定位。



圖二：網路運算架構圖

三、標準之爭

在第二節中，我們可以發現網路運算技術的成敗，分散式物件技術爲其中關鍵性的因素，在這方面推行最力的爲美國 OMG (Object Management Group) 協會，他們所制定的 CORBA 架構幾乎已經可以被視爲物件導向環境的最終型，也已經有完整的實作產品出現，如 Iona 的 Orbix 應用架構 [10]，甚至在其他類似的架構中也都可以看到它的影子（如微軟公司的 DCOM 技術）。在這一節中，我們先簡述一下 CORBA 的運作，接著我們將看看微軟公司爲因應 Internet 的應用，延伸其原有的 COM 架構而在 1996 年 3 月宣佈的 ActiveX 技術。儘管微軟公司宣佈 COM 一定會符合 CORBA 標準，但是其間還是有些差距，甚至微軟公司可能夾著其目前廣大的市場佔有率強力推銷 ActiveX 技術，成爲分散式物件技術的主流。但是還是先讓我們看看 CORBA 架構吧。



圖三 OMA 架構

圖三是著名的 OMA (Object Management Architecture) 架構，這也是 OMG 認爲典型的物件導向網路環境。

Object Services 提供服務給 client 端使用，每一種服務都是一個物件，擁有自己的資料與函數，並提供界面供外界呼叫。 Common Facilities 屬於 client 端的通用程式，這些共通的物件可以讓多個應用程式來使用。 Application Object 則爲應用程式物件，爲構成 client 端的主要部份。關於這些物件如何定義， OMG 提出的是 Core Object Model ，以比較嚴謹的方式來描述物件界面及實作，但是在實際應用時，爲了適應各種平台及程式語言，許多廠商以 IDL (Interface Definition Language) 作爲定義物件界面的方式，物件實作的部份則留給各程式語言發揮。回到網路運算技術的話題上，當我們考慮到跨平台的動機時，似乎 Java 就成爲我們目前實作物件的不二選擇。接著應該注意到整個 CORBA 架構中最重要的部份就是 ORB ，我們可以把 ORB 視爲電信系統中的交換機，或是軟體物件的匯流排，它負責網路上各個物件之間的傳呼與訊息交換，當然，它最重要的作用是在 client 端動態呼叫 server 端時（ client 端事先並不知道要由哪個伺服器提供服務時）將訊息送到伺服器端的 adapter ，再由伺服器決定由哪一個服務物件來接手 [3]。

目前 ORB 機制做得較成功的且應用比較廣泛的有三個產品： Hewlett-Packard 的 Distributed Smalltalk [8]、 IBM 的 Component Broker [9] 以及 DEC 的 ObjectBroker 。其中 DEC 的標準離 CORBA 標準仍有一些差距，然而微軟的 COM 架構卻是架在 DEC 的 ObjectBroker 上，甚至當 DCOM 面世的時候，其中的 SCM (Service Control Management)以及位於 client 端的 proxy 、 server 端的 stub 合起來就在做 ORB 的事情（除了一些關於動態連結上的差異），同時這樣的架構對於微軟原有的 COM 架構衝擊也是最小的，並延伸了 ActiveX 技術在網路世界的舞台。微軟提出 ActiveX 技術原本只是爲了整合 Java 與 COM 技術，但是其真正的受惠者卻是原本使用 C++、 VisualBasic 、 Delphi 等傳統開發 windows 軟體的設計師，因爲他們可以利用已經熟悉的工具來開發出與 Java 程式相當的網路程式。

ActiveX 如同 Java 程式一般被下載到用戶端的瀏覽器，所不同的是 ActiveX 是以可執行程式 (native code) 的形式，所以執行速度快些，但也正因爲如此，它就失去了跨平台的性質，解決方法似乎只有將 ActiveX 技術延伸到各種平台上，微軟承諾要支援 Macintosh 及 Unix 的平台，目前也已經有了測試版， Linux 的版本也即將出現，然而對 Web 網站而言，保留多個平台的同一份程式似乎並不是一個划算的選擇。當然微軟對於這點並不會很在乎，畢竟它目前已有最大的 windows 市場，它也希望將來只有這一個平台。微軟爲了展示其技術的可行性，並吸引廠商爲其開發應用軟體，已推出 NetShow （ VOD 應用的開發工具）及 NetMeeting （ video conferencing 應用的開發工具），此外也的確有許多廠商紛紛投入 ActiveX 的開發工作，以國內而言，趨勢科技的網路掃毒服務就是其中一個例子。

綜觀以上兩者，我們可以發現，以 Java 實作 CORBA 似乎較接近我們所期待的網路運算世界，舉凡跨平台的特性、更完整的分散式物件導向技術都使我們對網路運算的前景感到樂觀。同時微軟提出類似的技術也向網路運算的方向邁進，儘管失去了一些跨平台的功能，但是其目前廣大的市場以及現有廣大的程式設計師群（傳統 windows 程式設計師可以蠻容易進入 ActiveX 的領域）都是它最大的競爭優勢。在這樣的情況下，一些不知道也不想往某一邊靠的廠商開始研究把它們整合的方法，比如說 Novel 就成功的做了由 COM 技術到 CORBA 的軟體轉接器 (bridge)。

圖四將這兩種實現網路運算理想的技術做一些大略的比較。

JavaBeans (cross platform)	ActiveX (platform lock) (也許會打破)
Java	C,C++,Basic Java
CORBA	DCOM
任何作業系統	Win32

任何硬體	Intel-based 硬體

圖四： JavaBeans 及 ActiveX 的比較

四、結語

網路運算的概念與架構已成形，勢必完全改變軟體運作方式，以及我們使用電腦的方式。甚至整個網路運算將挑脫電腦範疇，朝向更生活化的方向前進。由於用戶端不須做任何管理與安裝程式，用戶端電腦將被做成一個一個的盒子 (box)，就像家電一樣，並開始進入家庭家電市場。隨著電腦如同家電般深入家庭，更多需要即時性 (real-time) 且屬於 stream-type 的服務將會出現，比如說隨選視訊、視訊會議、線上卡拉 OK 等，這些服務基本上都是需要使用網路大量的頻寬，並且對使用者而言很難忍受因延遲而導致的斷續現象。爲了要提供這樣的服務，分散式物件技術必須要能支援 stream-type 的資料，並且能滿足 QoS (quality of service)要求，而目前 CORBA 架構最弱的部份就在此，所幸 OMG 也已經開始注意到這個問題，我們可以期待在新版的 CORBA 規格中將會解決這個問題 [11]。然而除了物件技術需要改進之外，我們目前所使用的 Internet 也面臨了同樣的瓶頸，無法提供即時性服務、缺乏高速的網路以及缺乏 QoS 的觀念都是爲人所詬病的地方，爲了爲 Internet 找尋下一世紀的出路，在美國已經開始了一個稱之爲 Internet 2 的計畫，植基於 ATM (asymcronous transfer mode) 網路，利用一些新的通訊協定如 RTP (real-time transport protocol)、 RSVP (resource reservation protocol)，並將下一代 IP 協定 (IPv6) 作爲網路層的通訊協定以提供更快、更好的網路品質。相信經過這些技術的改進，網路運算所能提供的資源將會更加多元與方便。

不論將來網路運算技術長什麼樣子，也不論將來主流的分散式物件導向技術爲何，我們已經看到網路運算技術已經是不可被忽視的趨勢，甚至連微軟這個軟體業的龍頭老大都往這個方向前進，「物件」、「網路中心運算」真可稱爲世紀末的軟體革命。

參考文獻

[1] Sun Microelectronics. "Java JIT Compiler Overview," http://www.sun.com/workshop/java/jit/explanation.html. Sun Microelectronics, 1996.

[2] Sun Microelectronics. "Java Processors Intro.," http://www.sun.com/sparc/java/index.html. Sun Microelectronics, 1997.

[3] Object Management Group (OMG). "What Is CORBA????," http://www.omg.org/omg00/wicorba.htm. OMG, 1994.

[4] Sun MicroSystems. "Network-centric computing," http://www.sun.com. Sun MicroSystems, 1996.

[5] IBM Corporation. "Network-centric computing," http://www.as400.ibm.com/complex/mcgif2d.htm. IBM Corporation, 1997.

[6] 周瑞. OLE COM/ActiveX 程式設計. 台北：物澤. 1997.

[7] Chappell, David. Understanding ActiveX and OLE. Microsoft Corporation. 1996.

[8] Hewlett Packard. "HP Introduces CORBA 2.0-Compliant HP Distributed Smalltalk," http://www.hp.com/csopress/95aug07.html. Hewlett Packard, 1995.

[9] IBM Coperation. "IBM: Component Broker Home," http://www.software.ibm.com/ad/cb/. IBM Coperation, 1997.

[10] IONA Technologies, Inc. "Orbix for Windows - White Paper" http://www.iona.ie/Products/Orbix/Windows/WhitePaper.html. IONA Technologies, Inc., 1996.

[11] Object Management Group. "Control and Management of A/V Streams Requests For Proposal," OMG document:telecom/96-08-01, Auguest 1996.