Object-Oriented Web Services 應用- 解決 JCOM 在大量資料交換效能不佳的問題

黃昱凱
交通大學資工所
d9147424@hotmail.com

唐啟銘
交通大學資工所

袁賢銘
交通大學資工所教授
smyuan@gmail.com

***摘要*—根據統計資料，在全球前 1000 名的企業中，約八成以上都混合使用了 Java 和 Windows 技術，早期為了幫助這些企業解決 java 及 COM +元件之間的溝通的問題，許多中介軟體因而產生(例如 ：bridge2java，J-Integra for COM，Weblogic JCOM…等)。其中，JCOM 因為整合在 Weblogic Server 中，所強調的是提供方便直覺的開發架構，更是被許多程式開發人員所接受。但近年來，隨著企業內的資料量愈來愈大，我們發現以往使用 JCOM 做為異質資料交換系統的執行效率愈來愈差。**

因此我們希望能找到一種架構或方法，即能保留原來 JCOM 的直覺式開發的方便性，讓 COM+的開發人員享受 java 物件導向的優點，又能解決效能的問題。而這篇論文就是探討目前最熱門的 Object-Orient Web Service 架構與 JCOM / DCOM / COM 之間的差異性，以及嘗試用 Object-Oriented Web Service 來解決 JCOM 在大量資料交換時效能不佳的可能性。

***關鍵詞*—Object-Oriented Web Services, Web Services, Java to COM, DCOM, JCOM Bridge**

一、研究背景

長久以來，Java 一直許被多大型企業當作發展企業應用解決方案最佳的分散式平台，尤其是建構分散式伺服器的應用，而當中又以 21 世紀初提出的 Enterprise JavaBeans (EJB)架構最具代表性。

什麼是 EJB 呢?EJB 是實現 3-Tier 架構(將系統切分為使用者介面層(Presentation)、企業邏輯層(Business logic)及資料庫(Database))的一種方法。系統開發人員將企業邏輯程式寫成 EJB 元件，然後 Deploy 到 EJB Container 裡，藉由 EJB 的伺服器端元件架構以及分散式物件的技術(如 CORBA 及以 Java RMI)可以減少 Client 端和 Server 邏輯之的耦合度。Client 端應用程式透過 RMI(Remote Method Invocation)即可呼叫遠端的 EJB，而 EJB 則透過 Connection Pool 連結到資料庫。

但是 JAVA 最讓企業頭痛的部份，在於開發使用者介面應用程式所耗費的時間及資源，遠大於其他開發工具(Visual Basic，Delphi，C++…等)。所以在過去大多數的企業還是偏愛以其他程式語言開發使用者介面，尤其以半導體廠內的系統更是大宗(半導體廠內的系統多為封閉式的系統，無法採用 Web Application 架構)。因此才發展出現目前一般企業常見的架構(以 VB、C++或是 Delphi 呼叫 EJB 的分散架構)。

但 Java 及 Windows Com+是兩種完全不同的平台，為了能讓兩者之間能進行溝通或是資料交換，許多軟體供應商開始提出一些解決方法及架構(例如 ： WebLogic 提供的 JCOM 技術，bridge2java(IBM 提供的基於 Java 本機接口和 COM 技術)，J-Integra for COM…等)。

當時，Bea Weblogic 提出 JCOM 的技術，由於提供開發人員在異質開發平台上直接使用 JAVA 物件結構，並且將其整合於 Weblogic Server 中，提供完整跨平台系統架構，獲得許多大型企業所採用，也因此廣泛被應用在企業中。

近年來，許多企業不斷進行平行或垂直整合，系統與系統之間必須處理的資料量及複雜度也不斷地提高。這使得我們漸漸發現原先使用 JCOM 架構系統的效能似乎愈來愈差，甚至是無法順利運作。

一般而言，大企業若遇到這類的問題時，會有兩種解決辦法：(一) 提昇硬體規格來改善效能。(二) 重新開發新的系統來取代現在的系統。但這二種方式一則需要花費大量的金錢，二則需要耗費大量開發資源及時間。

因此，本篇就是探討是否能找到其他解決辦法，使其能在最短的時間用最少的花費來來改善 JCOM 效能不佳的問題。研究目的

根據 Bea Weblogic 官方對 JCOM 架構的定義，JCOM 其實是一個介於 Microsoft 系列應用程式及 JVM 執行環境的中介軟體。主要是負責將來自於 Microsoft COM+或 JAVA 的 Request 轉換成另一端系統所支援的傳輸協定並傳送至對方，也就是 COM/DCOM 及 Java RMI 之間的資料內容的轉換，這正是為什麼 Microsoft 或 JAVA 開發人員，能直接使對方元件/物件的主要原因。

雖然 JCOM 提供如此直覺化的方便性，但是 JCOM 的缺點在於一旦開發人員大量使用這種技術存取雙方物件/元件資料內容時，可能會因為 JCOM Runtime Engine 負載過重，導致執行效能不佳。

以目前的技術而言，只要談到異質資料交換機制，第一個連想到應該是 Web Services。Web Services 主要的優點是利用 XML(text-based) 作為資料交換的內容，而且它是一次性的傳輸，不需要浪費時間處理 COM/DCOM 與 JAVA-RMI 之間的轉換。所以才會有了這樣子的構想，希望用 Web Services 來解決 JCOM 效能不佳的問題。但 Web Services 是以 Text 為傳輸內容，省去了異質平台資料轉換的時間，但必須分別在 Server 及 Client 兩端執行資料格式的轉換(Object <->XML)及透過 HTTP 傳送的資料封包。這兩個步驟是否會產生反效果呢?這正是本篇文章所探討的內容之一。

另外，假設以 Web Server 為中介軟體的架構其執行效能真的比 JCOM 的方法好。但我們要如何將 Web Services 導入即存系統中，卻又能保有 JCOM 提供系統開發人員直覺式使用 JAVA 物件的優點，則是本文中要探討的另一個重點。因此，本研究的目的有下列幾點：

1. Web Services vs. JCOM 的效能比較。

2. 如何以最少的資源，將 Web Services 導入即存的系統，取代 JCOM。

我們知道 Web Services 最重要的一個步驟是產生 WSDL 文件內容，我們可以想像一下，如果能將 Binding 定義成自動透過 SOAP 呼叫 EJB 的方法，並且 EJB Method 的參數及回傳的 Java 物件定義 XML 格式內容，如此一來便能達到使用 XML 來傳來傳遞資料了(圖 1)。

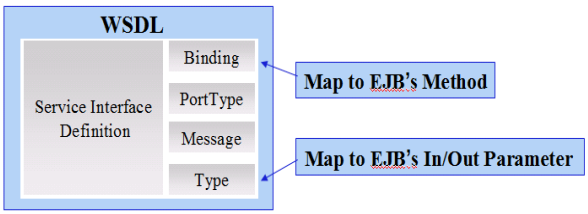


圖 1 Object-Oriented Web Services vs. EJB 的概念

但是 JAVA 及 EJB Interface 物件導向資料結構的複雜度，如果要手動產生出 WSDL 文件中的 Binding (Web Service 提供的 SOAP Method)，及 Types(xml 資料結構)的定義，反而會讓開發人員怯步。所以只能借助 OO-WS 的技術來產生 WSDL 的文件。

一般公司或企業通常將 Web services 應用於電子商務以及 B2B 與 EAI 系統中，而資訊人員而在設計這類型專案時，通常會先專注在 Web services 介面的定義以及溝通方式，接著才依據這 XML 內容來開發 JAVA 或是 COM 的程式。 這是所謂的 Contract-first 的開發流程。反之若開發人員先設計資料交換的方法及物件，就是 Contract Last 的作法。

二、OO-WS vs. JCOM 差異性實驗及分析

為了能真正了解 WS vs. JCOM 之間的優劣及差異性，我們分別實作了以下幾種資料交換架構的測試。

(一) 以 JCOM - Zero Client 模式為中介軟體的實驗

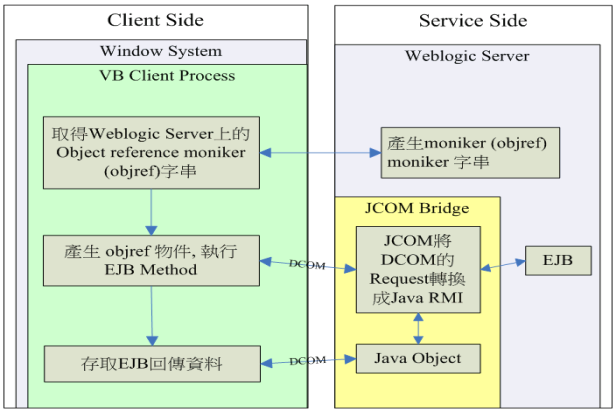


圖 2 Zero Client 資料流程圖

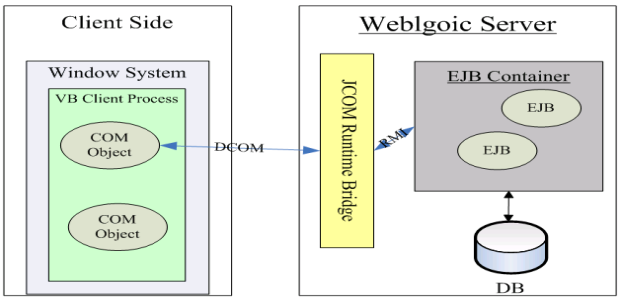


圖 3 Zero Client 的架構圖

(二) 以 JCOM Native Out Process + Late Binding 模式為中介軟體的實驗

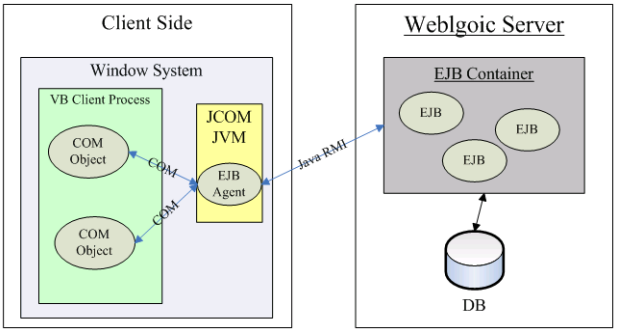


圖 4 JCOM Native Out Process + Late Binding 的架構圖

(三) 以 JCOM Native In Process + Early Binding 模式為中介軟體的實驗

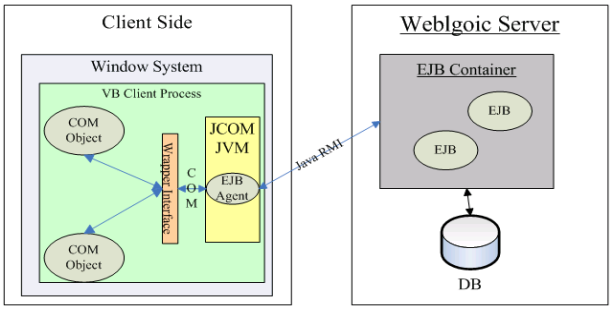


圖 5 JCOM Native In Process + Early Binding 的架構圖

(四) 以 Object-Oriented Web Service 為中介軟體的實驗

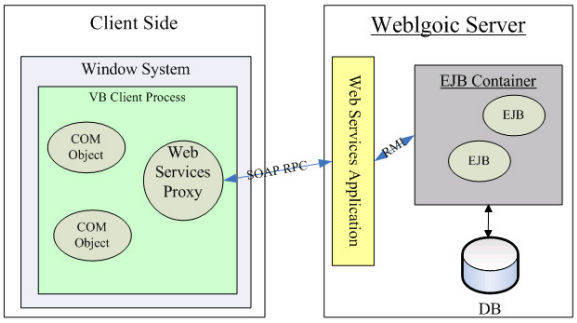


圖 6 Object-Oriented Web Service 的架構圖

該實驗是假設某公司要求開發人員必須開

發一個查詢/修改員工基本資料的介面為範列。軟體架構是以 Windows 系統平台為執行環境，以 Microsoft Visual Basic 6 開發客戶端應用程式透過 Intranet 向遠端 Weblogic Server 上的 EJB 查詢所有公司員工基本資料，並將查詢結果回傳至使用者介面。操作人員可以直接由介面修改員工基本資料後，並儲存至資料庫。

使用者操作介面說明下圖 7 所示。最上方的 Text 為查詢資料筆數輸入畫面，可用來設定查詢資料筆數。下方的 table list 則是呈現的員工資料，使用者可以直接在 table 裡修改資料。中間區域有許多 Button 的，左方 Query 區塊內有四個 buttons，分別透過四種不同的架構查詢資料; 左方的 Upload 區塊內同樣有四個 buttons，則是透過不同的架構回傳資料。左下方的 table 則呈現的是每個架構下查詢資料所需花費的時間:右下方的 table 則呈現每個架構上傳資料所需花費的時間。使用者依據不同的測試架構查詢/修改員工資料時，系統會將所花費的時間記錄在下方的 table 中。

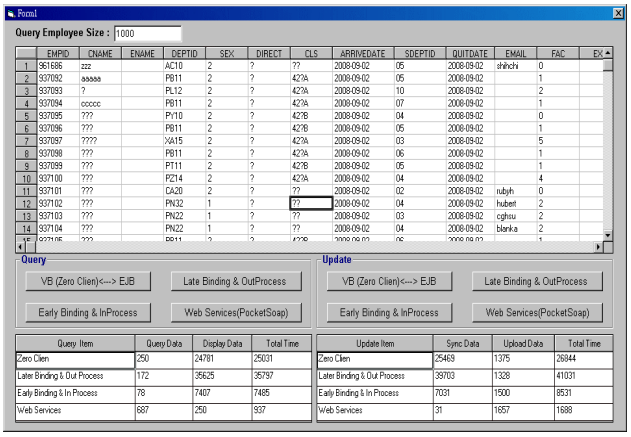
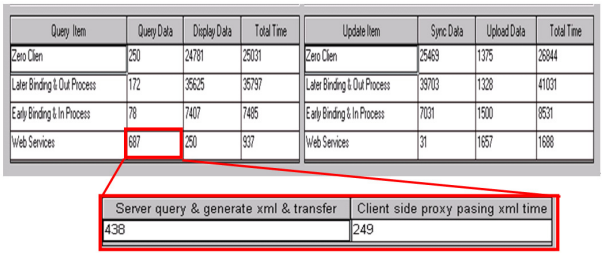


圖 7 JCOM Native In Process + Early Binding 的架構圖

實驗結果分析:

這個小節會依據實驗所得到的結果進行不同角度的分析，以下數據是測試後的結果。首先，我們分別針對每個模型執行查詢了 1000 筆員工資料，並且秀出員工的 14 個屬性資料，也就是會有 1000 * 14 個欄位資格需要呈現。

- 在網路流量，我們分別查得其所需的網路流量如下 :
  Zero Client Mode: 14.1195 / 5.4641 MB
  Native In/Out Mode: 0.1212 MB
  Web Services Mode: 1.0564 MB
- 執行總時間 :



Query Item	Query Data	Display Data	Total Time
Zero Clien	250	24781	25031
Later Binding & Out Process	172	35625	35797
Early Binding & In Process	78	7407	7485
Web Services	687	250	937

Update Item	Sync Data	Upload Data	Total Time
Zero Clien	25469	1375	26844
Later Binding & Out Process	39703	1328	41031
Early Binding & In Process	7031	1500	8531
Web Services	31	1657	1688

Server query & generate xml & transfer	Client side proxy pasing xml time
438	249

圖 8 各模型執行時間

圖 8 中的 Query Data 欄位所表示的是 client 端呼叫 Server 執行 EJB 所需的時間，而 Display Data 表示的是將 EJB 回傳的值顯示在 UI 上所需的時間，Total Time 則是總花費的時間。另外，我們也針對 web services 的 query data 的部份，區分為二段時間(1) Server query & Generate xml & Transfer date.(2) Client Side parsing xml to VB Object.。各模型花費最多時間依序是 JCOM Native Out Process + Late Binding Mode > Zero Client Mode > Native In Process + Early Binding > Web Services Mode。

- Client Side CPU 使用狀況 :

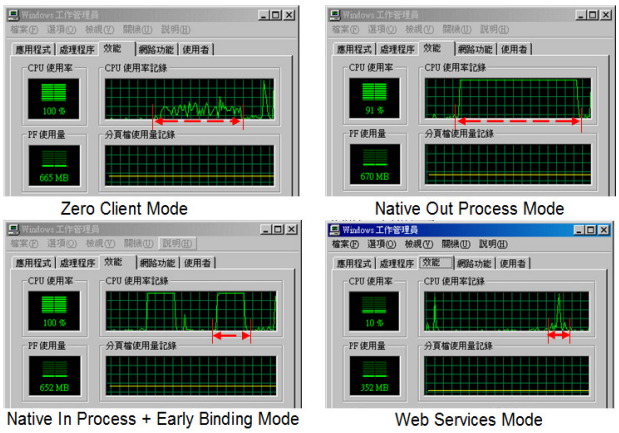


圖 9 各種模式 Client Side 的 CPU 運作狀態

- Weblogic Performance Monitor :

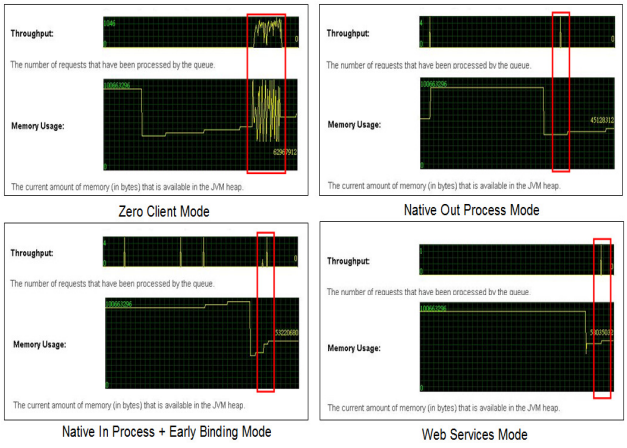


圖 10 各模型的 Weblogic Performance Monitor

各模型的 Weblogic Performance Monitor 的狀態依序是：Zero Client Mode > JCOM Native Out Process + Late Binding Mode ≈ Native In Process + Early Binding ≈ Web Services Mode。

執行效率(速度)分析

由圖 8 的執行結果，我們發現 Zero Client Mode 只花費不到一秒的時間就和 Server 取得物件，但是由於該模式是採用動態取得物件資料，也就是每當取得員工某個屬性資料時，便會透過 DCOM 向 Server 取得，這也證明了為何 Zero Client Mode 花費非常多的時間在呈現資料上。而由圖 8 我們得知四種模型中，執行速度最快的是 Web Services 架構。這個結果證明也驗證了本論文的可行性及目的。圖 11 是每種架構各執行十次的實際執行時間。

測試項目	取得資料	顯示資料	總時間
Web Services	500	266	766
Web Services	657	250	907
Web Services	516	250	766
Web Services	531	266	797
Web Services	531	297	828
Web Services	547	250	797
Web Services	579	296	875
Web Services	547	328	875
Web Services	500	266	766
Web Services	547	250	797
Early Binding & In Process	140	5719	5859
Early Binding & In Process	94	5656	5750
Early Binding & In Process	78	5484	5562
Early Binding & In Process	94	5437	5531
Early Binding & In Process	94	5484	5578
Early Binding & In Process	94	5297	5391
Early Binding & In Process	78	5328	5406
Early Binding & In Process	78	6719	6797
Early Binding & In Process	93	5297	5390
Early Binding & In Process	78	5469	5547
Early Binding & In Process	79	6890	6969
Later Binding & Out Process	157	26843	27000
Later Binding & Out Process	172	26593	26765
Later Binding & Out Process	141	26937	27078
Later Binding & Out Process	156	27781	27937
Later Binding & Out Process	156	26657	26813
Later Binding & Out Process	125	26422	26547
Later Binding & Out Process	172	26578	26750
Later Binding & Out Process	140	26766	26906
Later Binding & Out Process	141	28078	28219
Later Binding & Out Process	140	27329	27469
Zero Clien	31	18806	18837
Zero Clien	47	21375	21422
Zero Clien	47	19640	19687
Zero Clien	31	19125	19156
Zero Clien	47	19953	20000
Zero Clien	125	18516	18641
Zero Clien	31	19454	19485
Zero Clien	47	19609	19656
Zero Clien	47	18219	18266

圖 11 各架構 10 次實驗結果

至於為何 web services 和 In Process 之間的執行差距會有如此大的差距呢?我們後來分析的結論如下圖 12 所示。若 Client 端 VB 透過 JCOM Bridge 取得某一 JAVA 的資料，必須先透過 COM/DCOM 將 Request 發送到 JCOM Bridge(步驟 1), 然後 JCOM Bridge 將 COM Request 找到對應的 COM Proxy (步驟 2), 再藉由 COM Proxy 找到相關的 JAVA Class(步驟 3), 最後取得 Object value 後再回傳到 client side(步驟 4)。

可想而知，一旦大量透過 JCOM 取得 JAVA 物件的值，就等同於一直重覆步驟 1,2,3,4…，這就是為什麼 JCOM 在處理大量資料交換時效能不佳的因。

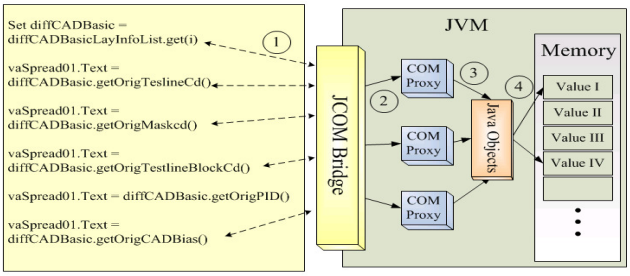


圖 12 VB 透過 JCOM Bridge 取得 JAVA 資料流程圖

另外，讓我們感到意外的是原本以為 Zero Client Mode 所花費的時間應該會是最多，但是結果反而是 Native Out Process + Late Binding 花費的時間最多，而且 CPU 使用度最高。歸究原因可能是因為 JCOM 的架構，分成二個步驟：

1. 利用 JCOM JVM，將 JAVA 資料轉換成 COM 物件。
2. 使用 DCOM/COM 進行資傳輸。

Native 模式雖然將 JAVA 物件載入客戶端，減少 Client <-> Server 之間的來回次數，但在本機端進行物件轉換動作及透過 COM 方式進行資料交換必須使用大量 CPU 時間，反而導致效能不好。反而是 Zero Client Mode 因為將 JAVA <-> COM Object 交換動作，交由 Server 執行，所以速度比較快。

至於 Native In Process + Early Binding 模式和 DOM 模式相比較，雖然同樣是將 JAVA 物件載入 Local Side 來處理，但是它先將 JAVA 物件透過 IDL 方式定義在 COM+ DLL Library 中，並且載入了 JCOM JVM DLL，所以整個資料交換動作都屬於同一 Process，所以執行速度較快。

但就 In Process Mode 和 Web Services 的比較，結果讓人非常驚訝。Web Services 花費的時間，是 In Process 的二分之一不到，若資料量愈大，差距愈明顯。而且 Web Services 所花費的資源，無論是 Server Side 或是 Client Side，都非常少。實驗前我們擔心 Web Services 在處理 XML 的 Parser 及網路流量過重會導致資料交換過慢。但在這個實驗中，XML 處理的速度似乎遠遠超小於 JAVA 與 COM 物間轉換的時間。

為了證明本論文的論點是否真的可行，我們亦針對上傳功能進行測試。該實驗是一次將 1000 筆資料回傳到 server side 進行儲存資料庫動作。結果如同圖 8 右方數值所示，web services 總花費時間也是最少的。

另外，由圖 9 及 10 中我們可以看到各模型所花費的 Client Side CPU 的使用度及 Server Side Weblogic Monitor 的 throughput 的使用狀況。Zero Client Mode 的 CPU 使用率雖然大約 50%左右，但是它是將資料轉換的動作交由 Weblogic server 執行，相對的必須花費 server side 的資源。

而 JCOM Native OUT/IN Process，則是會佔用 CPU 100%的使用率。不難想像若 Client Side 若

有其他的應用程式正在執行，必定會使得執行效能變差。而 Web Services 所耗費 Server 及 Client Side 的資源就相對減少許多，這也證明了 Web Services 取得 JCOM 的可行性。

Multi-Users 執行效能分析

以開發的複雜度來看，最簡單的架構毫無疑問的是 Zero Client 架構。但是 Zero Client 的執行效能不好，所以本章節只針對 Web Services Mode 及 Native In Process + Early Binding 進行比較。

通常應用系統不可能只有一位使用，若同一時間有多位使用者執行程式時，Web Services 是否還能保持這麼好的效率呢? 接下來我們將模擬多人同時執行查詢大量資料，並且針對結果進行分析。該實驗只針對 in Process 及Web Services 兩方法進行實驗。實驗過程中，我們準備四台 Client PC，並且於 PC 上安裝了 Weblogic JCOM bridge 及 JDK1.4。

為了放大數值的差異，我們預設是查詢 10,000 的員工資料。在測試時，我們先記錄每台 PC 各別的執行時間，然後再測試 4 台 PC 同時執行所花費的時間.結果如下方 4 個表格。

Web Services-(Single-User)				
PC No.	Server query & Generate XML & Transfer Time	Client side proxy pasing xml time	Display Time	Total Time
No. 1	3141	9094	2109	14344
No. 2	3938	19485	2428	25851
No. 3	3297	20670	4574	28541
No. 4	3766	15964	3324	23054

表 1 Execution Time of Web Services Single User

Web Services-(Multi-User)				
PC No.	Server query & Generate XML & Transfer Time	Client side proxy pasing xml time	Display Time	Total Time
No. 1	6781	13453	2741	22975
No. 2	10938	18470	2280	31688
No. 3	10297	22265	2157	34719
No. 4	8688	8828	4725	22241

表 2 Execution Time of Web Services Multi User

JCOM In Process-(Single-User)			
PC No.	Query EJB & Transfer Time	Display Time	Total Time
No. 1	813	596781	597594
No. 2	953	889203	890156
No. 3	906	1265891	1266797
No. 4	1015	1041382	1042397

表 3 Execution Time of In Process Single User

JCOM In Process-(Multi-User)			
PC No.	Query EJB & Transfer Time	Display Time	Total Time
No. 1	1547	654849	656396
No. 2	2359	907107	909466
No. 3	1672	1499113	1500785
No. 4	2656	1049737	1052393

表 4 Execution Time of In Process Multi User

將上述資料整理後，我們可以得到以下兩個表格圖 13、14，其中圖 13 的左邊是 Web Services Single-User and Multi-User 的 Call EJB + Data Transfer 的平均執行時間，而右邊則是 JCOM in Process 的平均執行時間。由圖中我們可以很清楚看出，Web Services Mode 的 Server 的執行時間， 會因為同一時間執行人數的多寡而產生較大的落差。這是可以理解的結果， 因為 Web service 必須先將資料轉換成 xml format，再將 xml 透過 HTTP protocol 傳送到 client 端。一旦同時間多人使用的話，server 的 loading 就會增加，執行時間也會增加。而 JCOM 的架構，Server 只需負責查詢資料，然後將資料透過 RMI protocol 傳送到 client JCOM-Bridge，JCOM 對 Server 的 loading 遠比 Web Services 小。

看到目前為止，也許大家會開始質疑 Web Service 替代 JCOM 的可行性.但接下來的圖 14 或許可以讓大家放心。圖 14 左邊是 Web Services Single-User and Multi-User 平均的 Total execution time，而右邊則是 JCOM in Process 平均 Total execution time. 雖然 Web services 因為 4 位使用者同時執行而平均執行時間增加了 6

秒，但平均才花費 29 秒左右的時間。而 JCOM in Process 雖然前後只增加 2 秒，但是平約執行時間是 967.2 秒，約 16 分鐘，兩者相差 32 倍. 所以 Web Services 在多人使用的情況下，進行大量資料交換時，仍然是比 JCOM 好。

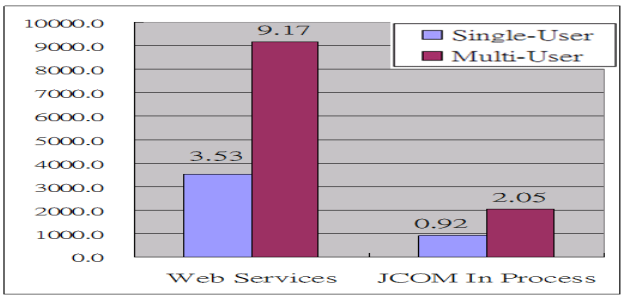


圖 13 100M bandwidth execution time

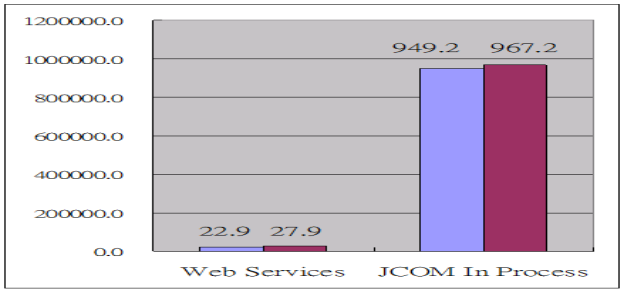


圖 14 100M bandwidth execution time

三、結論

首先，本篇論文的研究的並非強調 Web Services 可以完全取代 JCOM/DOM 的架構。在許多情況下 JCOM/DCOM 仍是有其優勢。例如若須要透過網路執行遠端應用程式，並且不需要進行大量資料往返的系統時，Zero Client Mode 就比 Web Service 來的方便許多。

而本研究目的是希望以 Object-Oriented Web Service 架構，解決 JCOM/DCOM 在處理大量資料交換時效能不佳的問題，並且提出一套真正方便、有效率的實作方法，可減少企業導入 Web Services 可能會遇到問題。在本篇論文的實驗結果中，可以很明顯的看出來，Web Service 的執行效能在大量資料交換時，的確比 JCOM 來的好。

過去幾年，所以許多大型企業為了解決 JCOM/DCOM 效能的問題，情願花多上 2~3 倍的人力及時間去開發新的 Web 化系統，也不願導入 Web Services。我們歸究其主要的原因有二:

1. WSDL 的定義太複雜以及需要花費許多時間在處理 XML 往返資料上。許多企業資訊人員一談到要將 Object-Oriented (物件導向)的程式改成用 XML Schema 結構表示時，往往花費許多腦力，以致不想使用，甚至完全排斥它。

2. 資訊安全的問題。

但是就目前的技術成熟度而言，這些都已不再是問題。藉由本篇論文的實驗結果，證明了 Web Service 的執行效能在大量資料交換時，的確比 JCOM 好。並且證明了 Web Services 只要是有好的輔助工具，它不再是如此的複雜難以使用。

而 Web Service 安全上的疑慮，雖然這不在本篇討論的範圍內，但目前也已經有許多解決方法來防止資料外洩(例如:HTTPS、XML 資料壓縮、資料加密、身份驗證…等)，甚至是將 Web Application 的安全管理機制套用在 Web Services 上都是可行的。

五、參考文獻

[1]. 林章鈞，「開啟 EAI 技術新頁的 Web Services」，資訊與電腦，pp. 30-33, 09/2002。

[2]. 張思源，「企業應用 Web 服務之策略」，財團法人資訊工業策進會，資訊與電腦雜誌出版, 2002

[3]. 楊仁達，「物件導向技術與標準」，資訊與電腦，pp. 136-141，05/1996

[4]. 簡西村, "Web Services 應用與發展」，資訊與電腦，pp. 73-77，08/2002

[5]. 簡西村，「網路服務化：Web Services 技術與應用"，資訊與電腦，pp.86-92，10/2002

[6]. 簡瑞炤, "以 XML 標準透過 Web Service 達到異質資料庫資料交換機制", 亞洲大學資訊工程學系碩士論文, 2006

[7]. 劉遠威、黃雯汝, "Web Services 帶來整合革命", 資訊與電腦, 261 期 p:19-25, 2002.04

[8]. 戚玉樑、彭淑芸、張琪瑩、賴德優，Web Services 探索與應用，全華科技圖書，民國九十二年

[9]. 戚玉樑，網際服務技術導論，全華科技圖書，民國九十三年

[10]. B. Medjahed et al., "Infrastructure for E-Government Web Services", Internet Computing, IEEE CNF, vol. 7, no. 1, pp.58 - 65, 2003

[11]. Binder, W. Hulaas, J. Moret, P., "A Quantitative Evaluation of the Contribution of Native Code to Java Workloads", Workload Characterization 2006 IEEE International Symposium on, IEEE CNF, pp.201 - 209, 25-27 Oct. 2006

[12]. Brenner M. R., Unmehhopa M. R. "Service-Oriented Architectureand Web Services Penetration in Next-Generation Networks", Bell LabsTechnical Journal, Vol.12, No.2, pp.147-160. (2007)

[13]. Curbera, F. et al., "Unraveling the Web Service Web. An Introductionto SOAP, WSDL, and UDDI", IEEE Internet Computing, vol. 6, no. 2, pp. 86-93, Mar./Apr. 2002

[14]. Chi Zhang;Guang-Jun Huang;Jian Wu, 等，「分佈式組件與 Web 服務集成技術研究」，微電子學與計算機，23 卷 3 期，03/2006

[15]. Davis, A., Du Zhang, "A comparative study of DCOM and SOAP", Multimedia Software Engineering Proceedings. Fourth International Symposium on , IEEE CNF, pp.48 - 55 , Dec. 2002

[16]. Donglai Zhang, Coddington, P., Wendelborn, A., "Binary Data Transfer Performance over High-Latency Networks Using Web Service Attachments", e-Science and Grid Computing, IEEE CNF, pp.:261 - 269, 10-13 Dec. 2007

[17]. Eric Newcomer 著，深探網路服務，黃義焜譯，培生教育出版，台北，民國九十一年。

[18]. Goth, G., "News: data-driven enterprise - slouching toward the semantic Web", Distributed Systems Online, IEEE JNL , Volume 7, Issue 3, March 2006

[19]. Gustavo Alonso, Fabio Casati, Harumi Kuno, et al., "Web Services Concepts, Architectures and Applications", 2004

[20]. He Guo; Chunyan Guo; Feng Chen; Hongji Yang;, "Wrapping Client-Server Application to Web Services for Internet Computing", Parallel and Distributed Computing, Applications and Technologies, IEEE CNF, pp.366 - 370, 05-08 Dec. 2005

[21]. Jia-Rui Chen;Guo-Yong Cai，「基於擴

展 WSDL 變異的 Web 服務測試方法」，計算機應用，pp.1725-1728，27 卷 7 期，07/2007

[22]. Jiang, J.; Systa, T., "UML-based modeling and validity checking of Web service descriptions", Digital Object Identifier, Web Services, 2005. ICWS 2005. Proceedings., IEEE CNF, page:460, 11-15 July 2005

[23]. K. Gottschalk, S. Graham, H. Kreger, et al., "Introduction to Web Services Architecture," IBM System Journal, vol. 41, no. 2, pp.170-177, 2002

[24]. Lin, J., Taso, H., and Chu, Y., "Object-Oriented Analysis and Designof Web-Based Information Systems", IEEE Engineering of ComputerBased Systems, 2001, pp. 68-75.

[25]. Magdalenic, Ivan; Vrdoljak, Boris; Skocir, Zoran; "Towards Dynamic Web Service Generation on Demand", Software in Telecommunications and Computer Networks, IEEE CNF, pp.276 - 280 , Sept. 29 2006-Oct. 1 2006

[26]. Martin Tsenov.WAN communication using SOAP protocol[J]. Interbational Conference on Computer Systems and Technologies:e-Learning. 2003，406～410.

[27]. Newcomer E., Lomow G. (2007). "Understanding SOA with WebServices." Pearson Education, Inc.

[28]. Parsa, S., Ghods, L., "A new approach to wrap legacy programs into web services", Computer and Information Technology, IEEE CNF, pp.442 - 447, 24-27 Dec. 2008

[29]. Paul Muschamp. An introduction to Web Services [J].BT Technology Journal.2004,22(1).

[30]. Ren-Jin Zhang, Bin Liu，「Application of Component in Web Services Application Integration」，廣西師範大學學報(自然科學版), pp.144-147, 25 卷 4 期，12/2007

[31]. Roy, J., Ramanujan, A., "Understanding Web Services", IT Professional, Vol. 3, pp. 69-73, 2001

[32]. SeokHyun Yoon, DongJoon Kim, SangYong Han, "WS-QDL containing static, dynamic, and statistical factors of Web services quality", IEEE CNF, pp.808 - 809 6-9 July 2004

[33]. Stal, M., "Web Services：Beyond Component-Based Computing," CACM, Vol. 45, pp.71-76. 2002

[34]. Steve Vinoski , "Web services Integration Models", IEEE Internet Computing, pp.89-91, May/June 2002.

[35]. Vassilis Kapsalis,onstantinos Charatsis,Manos Georgoudakis, Efstratios Nikoloutsos,George Papadopoulos.A SOAP-based system for the provision of e-services[J].2004,（26）:527～541.

[36]. Web Services Conceptual Architecture 1.0, IBM Software Group, 2001

[37]. Wei Yu;Ming Cai，「基于 Web 服務的遠程數據訪問」，江南大學學報(自然科學

版)，pp.536-538，5 卷 5 期，10/2006

[38]. Zhang, C. Jacobsen, H.-A., "Refactoring middleware with aspects", Parallel and Distributed System, IEEE CNF, pp.1058 - 1073, Nov. 2003

[39]. Zhuang-Ye Liu；Zheng Yao，"基于 Web 服務的教師管理系統的設計與實現"，中國科學院研究生院學報，pp.127-131, 26 卷 1 期, 01/01/20