2008

Reseach on Cloud Computing Service Oriented Architecture (雲端運算服務導向架構之研究 楊建民 1 蔡鈞華 2 劉俊宏 3)

Chun-Hua Tsai

雲端運算服務導向架構之研究

楊建民[1]　　蔡鈞華[2]　　劉俊宏[3]
[1]政治大學資訊管理系　jmyang@mis.nccu.edu.tw
[2]政治大學資訊管理系　96356027@nccu.edu.tw
[3]政治大學資訊管理系　96356036@nccu.edu.tw

摘要

隨著雲端運算基礎架構的發展，如何結合企業資訊服務與外部雲端運算資源，是當前重要的研究課題。目前市面上一些主要的資服業者，如 Google、IBM 以及 HP 等，大力推動並提供強大雲端運算能力與完善的應用服務環境；然而目前雲端運算架構多需將整體應用與資料放置於外部平台，引發企業導入雲端運算時控制權移轉與資訊安全的疑慮。

本研究係採用 Google 雲端運算環境，結合雲端運算與服務導向架構，發展一個四層式服務平台，其中應用介面層利用開放平台的優勢，降低服務介面開發的負擔。中心控管層連接服務供應方與服務需求方，掌控完整的服務流程，以及使用外部雲端運算與資料中心之資源。服務供應層結合各式應用服務與資料來源，藉由將運算與資料分離的方式，讓服務需求者保持對資料的控管權力，以解決企業導入雲端架構時對資安的疑慮。基礎架構層可連結雲端運算資源，未來可利用各雲端業者所提出的服務架構、運算能力以及儲存資源，並依照個別平台的優勢與特性，充分發揮雲端運算架構的效益。

最後本研究藉由發展數位學習社群網站之應用，以說明本服務平台之可行性與整合外部雲端運算服務的優越性，以及未來研究方向。

關鍵詞：雲端運算、服務導向架構、MVC 架構、Google 運算、數位學習社群

1. 前言

當企業面對營運環境的挑戰，為即時回應改變的需求，系統開發的趨勢轉為重複利用舊有與外部的服務，改以組合式應用服務(Composite Application)模式來滿足需求。這樣的思維已漸漸取代從頭開發全新的系統，事實上在 2003 年已經有許多商業應用服務採用外部資源來完成，也就是利用組合現有資源的方式來滿足系統開發的要求。因此，許多系統開發專案建立了可程式化介面(Programmatic Interfaces)，與舊有系統服務或外部資源進行整合，將異質平台(Heterogeneous)組合而成的應用服務導入到實際交易過程中。在實際的企業營運模式裡，組合並整合舊有與新開發的應用元件(Component)來完成服務，正滿足服務導向架構(Service Oriented Architecture)環境所要求的特性(Natis, 2003)。根據研究報告指出，2008 年服務導向架構將成為普遍的系統開發模式，取代已沿用 40 餘年的整體系統開發(Monolithic Software Architecture)模式(McCoy & Natis,

2003)。

服務導向架構是一種軟體架構，要求軟體元件間關係鬆散，並且提供開放的服務介面讓服務供需雙方溝通，因為當組織面臨問題時，可能需要分散在不同領域下的能力或資源，服務導向架構是將分散的能力或資源將其組織起來，並用來解決組織所面臨問題的一種準則。服務導向架構能夠整合業務流程，在技術上使用安全且標準的服務元件，而這些服務元件能夠因業務上的改變而再利用或重新組合(Bieberstein, Bose, Fiammante, Jones, & Shah, 2005; Pallos, 2001)。

知名網路書城亞馬遜(Amazon.com)於 2002 年推出 AWS(Amazon Web Services)平台服務，容許遠端的網路服務(Web Service)得以在這個平台上運作。亞馬遜提供網路服務給其他的網站或是使用者端的應用程式，這些網頁服務並非直接提供給使用者，而是提供方便的開發介面，由軟體開發者來進行開發後，於 AWS 平台上發佈，供使用者來使用。在 2007 年 6 月，亞馬遜聲稱有超過 33 萬的軟體開發者註冊了 AWS 服務(Amazon.com, 2007) 。

隨著搜尋網站業務的成功，Google 提供許多成功的網頁服務，包括如電子郵件、線上地圖、辦公室應用軟體、社群服務與影片分享等免費服務。有些公司開始外包次要的服務由 Google 提供，將營運重心專注於本身的核心競爭業務，換句話說，企業會讓原本應該是自己經營的業務轉由其它合作伙伴來替他完成(Gray, 2007)。如 SaaS (Software as a Service)的領導廠商 SalesForce.com，宣佈參加 Google Apps 計畫，採用 Google 的 Gmail、Google Docs、Google Talks 與 Google Calendar 等應用服務，導入到現有的網頁版 CRM 應用服務中(Salesforce.com, 2007)。

雲端運算(Cloud Computing)提供者，讓使用者利用它們的基本架構，客戶可以藉此建立特有的應用服務；Google 也利用雲端技術提供應用程式與服務給使用者，並提供方便的開發介面，讓開發者能夠輕易發展出各類型的應用；而 Salesforce.com，則將應用服務建立在各服務平台提供的基礎架構與應用服務之上。(Morrison, 2008)指出企業資訊服務未來與服務領導平台整合的發展趨勢；相關平台業者 Google 宣布加入 IBM、HP、Intel 與 Yahoo 形成聯盟，並宣稱將會投入數以百萬計的資金在雲端計算的相關專案上。然而，雲端運算之發展雖日益完備，然使用者對於資料儲存與運算的安全性仍有疑慮，致使目前之使用尚未能普及，若能加強客戶端對於資料與流程的控管權力，可消除導入雲端運算架構的隱憂。

在雲端運算基礎設施逐漸完善的情況下，如何將企業應用服務與平台發展趨勢相結合，避免重複投資的浪費，並將資源投注於企業核心業務上，應是企業未來應該思考的重點。根據以上的開發思維，本研究目的主要在雲端運算基礎之上發展滿足服務導向架構特色的服務平台，該服務平台可以結合外部網路服務、充分利用現有的資源、降低系

統開發的時間與成本、同時提出分離運算處理與資料，藉此消弭企業對於導入雲端運算架構的隱憂，並有助於雲端服務市集形成。本研究亦將以數位學習社群之應用，說明本服務平台符合服務導向架構之特色與雲端運算之優勢。

2. 文獻探討

2.1. 服務導向架構

服務導向架構是一種軟體架構，強調低耦合性(Loosely-Coupled)、可程式化介面開發與服務供需雙方間的介面呼叫(Interface Calls)。換句話說，不同的元件間將是以獨立作業的模式進行，在組成整體服務的過程中，無需知道其他元件的開發方式。服務導向架構是設計來充分利用現有的資源，滿足快速進入市場的需求，並減少系統開發與維護的成本與風險(Conlon, G.Hale, Lukose, & Strong, 2008; Pallos, 2001)。

根據學者的分析指出，服務導向架構的優點包含減低系統整合的成本、改善企業回應的敏捷度與彈性、增進資產活用程度，更重要的是增進投資報酬率。從另外一個角度來看，改用服務導向架構架構讓合作伙伴間的系統整合變的更為容易，例如供應商與客戶間的供應鍊系統整合、將 IT 服務進行外包等應用，服務導向架構同時幫助企業實現公用運算(Utility Computing)的概念(Feuerlicht, 2006)。關於服務導向架構的技術特性，請參考表 1。

表 1：服務導向架構技術特性

分散式架構	企業所需資源可能分散於不同領域，需透過服務間良善的溝通機制來連結資源。
關係鬆散的界面	在鬆散的環境下，開發人員可以輕易重複利用現有的服務元件。
開放的標準	開放標準是服務導向架構的核心特色，可避免異質平台整合上的困難。
定義明確的服務合約	服務合約決定每個服務的功能及呼叫使用的方法，提供如何使用服務的導引。
服務可發掘且動態取得	動態載入所需服務，而不需使用者介入，將可充分展現服務導向架構的平台整合特性。

資料來源：(Huang, 2006)

隨著相關技術與基礎設施的成熟，服務導向架構概念也越來越被廣泛採用，成為替代舊有系統開發模式的新解決方案；由於服務導向架構可以滿足未來的應用程式許多嚴苛的需求，如：滿足大量互動的特性(Interaction Style)、快速開發、保持對需求快速改變的服務提供能力、服務品質、彈性與成本效益，這些特性都可以藉由服務導向架構開發模式來加以達成。因此，服務導向架構已成為相當普遍的系統開發模式(Brereton et al.,

1999; McCoy & Natis, 2003)。

根據 W3C 的定義，網路服務(Web service)是一個軟體系統，用以支持網路間不同機器的互動操作。網路服務具有機器可處理的格式(WSDL)，其它異質系統藉由同樣規格的訊息與網路服務溝通(SOAP)，通常來說這些格式、訊息都是以 XML(Extensible Markup Language)等開放標準的格式所發佈。因為網路服務跨平台、分散性與有彈性等優點，許多學者將網路服務視為非常適合發展服務導向架構的解決方案(Conlon et al., 2008; Group, 2004)。

2.2. 網格運算

結合網路服務的網格運算的發展趨勢，可區分為兩個部分，第一是以資訊科技層級的角度出發，所有的服務與網格資源皆由網路服務的開放標準來定義；第二是更高層次的資源共享，如商業服務或商業流程等。配合服務導向架構與網格運算趨勢的發展的新架構思維，改變發佈資料與連接應用服務的方式。服務需求者與提供者間的關係，將透過一個新的基礎架構來連接，而各利益攸關者(stakeholder)專注於本身的核心業務，換言之，利他就是利己會是未來軟體服務產業的重要思維(Zhang & Jeckle, 2004)。

網格運算於 1995 年間提出，其概念源自能源網格的概念，衍生發展成分享計算與儲存的資源，且網格間的介面並不需是標準化的，換言之，資源間將可任何相互流用。以網路服務為中心的網格運算(Grid Computing)，被視為是未來科技發展的潮流；網格運算可以分割工作由多個運算資源來完成，以達到良好的價格效能比。網格運算所使用的運算資源可能處在內部或是外部，並可提供以下的優點：彈性資源配置與管理、安全性、提供不中斷的服務品質與增進資源的利用(Boden, 2004; Zhang & Jeckle, 2004)。

2.3. 雲端運算

雲端運算可以從系統平台與應用服務兩個角度來解釋，從系統面來看，雲端運算系統可以依需求動態新增(Provision)、安裝、設定與移除伺服器的一種平台。雲端運算架構也能包含伺服器提供環境所需的其它計算資源，如：網路儲存群(SANs)、防火牆與其它網路設備等。從應用服務的角度來解釋，雲端應用服務代表利用大量的資料中心(Data Center)與強大的伺服器群來提供服務，並可直接透過網際網路連接，任何使用者只要有適當的連線能力與瀏覽器即可取得服務(Boss, Malladi, Quan, Legregni, & Hall, 2007)。

雲端運算架構可以讓企業投資在軟硬體上的成本更有效益，打破單一系統的最大限制，將運算群視為一個整體，具備自動系統管理、負載平衡與虛擬化技術等特點的一個整體，並可在幾分鐘內完成運算資源的維護，降低系統的複雜度，讓創新的應用更容易實現。無論是 Java、Linux、LAMP 等以堆疊(Stack)方式的應用服務，或是由 Google 所提出的 MapReduce 與 Google File System 都可以利用雲端架構來加以實現，換句話說，將現有的應用服務轉由雲端技術來實現是可行的(Boss et al., 2007)。

雲端運算環境支援網格運算的特性，如快速提供運算資源供網格應用程式運行等。但網格運算著重需要大量運算的工作，由許多小運算單元共同進行，這通常需要數以千計的電腦支援才可能達成。而雲端運算同時也支援非網格運算的環境，如傳統的 Web 架構應用服務，雲端運算架構不僅僅是聚集大量的運算資源，同時也提供了管理的相關機制，包括：部署、重新部署、負載平衡、移除與監控等(Boss et al., 2007)。

根據 Gartner 研究機構的研究，指出雲端運算代表商業模式的革新，影響到電子商務的發展，而各種對於『雲端運算』這個名詞的矛盾與混淆，代表它改變 IT 市場現狀的潛力。Gartner 將雲端運算定義為一種運算的模式，包含可大量擴充 IT 提供相關的技術能力，利用網際網路的相關技術，包裝成一種服務來提供(as a service)給多個外在使用者(Stamford, 2008)。

當雲端運算架構環境持續發展，運算服務變的無所不在，服務交易平台可以讓彼此交換運算資源，同時負責服務供需雙方的搓合與連接，並提供服務發佈、發掘、服務品質控制、計費機制與資訊安全保護等機制，而形成一『全球雲端服務交易市集』(Buyya, Yeo, & Venugopal, 2008)。

3. 研究方法

3.1. MVC 架構

MVC（Model-View-Controller）用於表示一種軟體架構樣式。它把軟體系統分為三個基本部分：模型(Model)、檢視(View)和控制器(Controller)。模型交由資料庫專家進行資料管理與資料庫設計的工作；檢視負責提供圖形介面，是負責與使用者互動的窗口，一般而言由美工人員負責；控制器負責提供系統的應用功能，而這些應用功能將由程式開發人員所提供。MVC 的目的是實現一種動態的系統開發，使對系統的後續修改與擴展簡化，並盡量重複使用已開發的系統元件。透過這個開發樣式可以將系統複雜度簡化，使系統架構更加直覺，專案中各專長的開發人員也可以作適度的分工，讓系統開發的工作能夠更順利與有效率(Reenskaug, 2003)。

根據 IBM 評估內部入口網站的評估報告指出，使用 MVC 架構存在許多優點，如提供網頁程式的基本框架，讓開發人員可以輕易的加入現有的應用資源；使用保有彈性的開發過程，減輕系統開發時的負擔；藉由控制層連結介面層與模型層的三層架構，讓網頁程式開發人員可以輕易的分工，將服務邏輯交由其它服務提供者負責，轉而專注於如介面與資料的處理等，可見 MVC 設計樣式的可用性受到相當肯定，並足以負擔商用營運環境的嚴苛挑戰(Yan, Leip, & Gupta, 2005)。

以網路服務為基礎的開發模式，相當適合採用 MVC 架構進行實作，由控制器負責流程控制的工作，組合不同的網路服務與模型所定義的資料格式，完成使用者的服務要

求，最後再透過檢視將服務成果呈現給使用者。以設計樣式(Design Pattern)而言，MVC 架構的拓展可視為服務導向架構的理想實作模式，因其強調的動態系統開發模式，讓系統的修改與拓展盡量的簡化，並能將系統適度分工與強調元件重複利用等特性，都與服務導向架構所強調的低耦合性、專注介面開發、充分利用現有資源、減少系統開發時間、低成本、敏捷回應等優點相近。拓展後的 MVC 架構，也可充分利用雲端運算的特性，讓企業投資獲得更大效益、充分利用軟硬體資源、提供強大的服務能力與高品質服務水準(Davis, Demetriou, Gaizauskas, Guo, & Roberts, 2006; Reenskaug, 2003)。

3.2. Google 雲端運算環境

Google 推出了各式各樣憑藉於雲端運算架構下的網路服務，Google Apps 包含：Gmail、GTalk、Google Docs 與 Google Sites 等服務，其所適用的領域包含溝通、協同合作、企業服務解決方案等。Google 同時提供了方便的 API(Application programming interface)供第三方開發者實作，各網路服務提供者能夠輕鬆的憑藉在 Google 強大的運算、儲存資源上，減低系統開發上的難度，如單一使用者登入(Single Sign On, SSO)機制與 Email Migration API 等功能，使得 Google Apps 環境能夠與現有的企業內部服務整合，提升產品開發效率並推出市場，提供使用者更滿意的服務體驗，同時也讓企業本身保有更強的競爭能力。

Google 強調通訊的重要，「Google Apps」可提供各種服務來協助您更容易創建一個社群，輕鬆地為所有使用者部署應用程式，省下更多時間和成本以專注在核心目標上。透過 Google 的代管服務，使用者無須安裝或購買任何軟硬體，可以將導入的成本減到最低。這些服務包括 Gmail、Google 日曆、Google Talk、Google Docs、Google Sites 以及網頁建置服務等等，Google 將這些服務整合在一個網域內，讓以往開發成本很高的使用者管理、協同與溝通機制等，都由 Google 的基礎設施負責提供，讓社群網站架設的成本大幅降低，也促使更多第三方開發人員加入這個平台，共創更大的平台價值(Google.com, 2007)。

Google Sites 是讓使用者快速、即時傳遞訊息的良好工具，使用者可以在 Sites 中討論、添加附檔、結合上述提及的各種 Google 服務，甚至與其他的資訊來源作整合，是故使用者可以輕易的加入影片、照片與線上文件等內容。使用者如編輯文件般建立自己的 Sites，並且保有控制分享的權力，可將 Sites 在個人、團隊或組織中分享，甚至直接發佈到網路上，發佈的內容將可輕易的被 Google 強大的搜尋引擎所找到。

Google Sites 富有彈性的整合特性，讓以往需要花費高成本與冗長開發時間的企業入口網站，能夠在彈指之間做出雛形，並輕易的整合各式資源，讓企業營運所需的各項服務，能夠透過第三方的服務提供者，方便的達成服務需求。本架構採用 Google Sites 作為前端的呈現介面，並且導入其協同合作的強大功能，將 Sites 視為本研究架構的介面(Interface) (Google.com, 2007)。

Google 小工具(Gadget)是以 iFrames 提供的簡單的 HTML 和 JavaScript 小型應用程序，可嵌入網頁和其他應用程序中。根據 Google 官方網頁的解釋，小工具包含易於建構、多平台運行與結合 Google 現有服務、流量的優勢。因為小工具採用 JaveScript 函式庫，讓開發工作的負擔能夠減輕，有助於快速、低成本的建構出可用的服務；小工具也具有可在多個平台上運行的能力，例如 iGoogle 入口網站、Sites 與一般網頁等，都可以輕易的嵌入已完成的小工具；小工具可以有效的利用 Google 所提供的服務與帶來的流量，讓開發者能夠專注於應用服務的實現，將開發的效益最佳化(Google.com, 2008)。

2008 年 4 月，Google 推出『Google App Engine』，讓開發人員可以利用 Google 的基礎環境來運作其網頁應用程式；在雲端架構環境之下，可忽略運算、儲存資源的限制，專注於應用程式本身的開發。該應用程式將可利用 Google 的網頁空間代管(Hosting)、資料庫(Bigtable)、自動負載平衡、完整支援 Google APIs、支援本地端開發等特性，讓開發人員開發出的網頁應用程式，具有可供百萬人次使用的應用能力(Google.com, 2007)。

圖 1：Google Cloud Computing 服務架構示意圖

3.3. 開發工具

MVC 架構將系統架構分為模型、檢視與控制器三個部分，本研究架構與傳統的 MVC 架構不同，將控制器定義為中心控管層，依照服務邏輯進行服務分配的功能；檢視定義為應用介面層，以 Google Sites 搭配 Gadget 的方式進行人機介面設計；而模型定義為服務供應層，提供資料中心的存取與應用服務的提供。

隨著開放原始碼軟體這幾年受到廣泛的接受，在企業環境中應用開發原始碼軟體進行開發的情況也日益普遍，組合商用軟體與開放社群軟體並用的開發模式，成為企業在選擇開發工具時的普遍考量；隨著越來越多的社群與開發資源的投入，開放原始碼的應用較以往更為成熟，也更能負荷商用營運環境的嚴苛挑戰，其中 Apache 專案的成功，囊括了大半網頁伺服器的市場，堪稱為最受歡迎的社群產品之一；而開放原始碼的開放特質，也讓開發人員能有更多的揮灑空間，而不至於受到個別平台的限制(Erdogmus, 2008; Yan et al., 2005)。

動態語言(Dynamic Language)是程式執行期(Run-Time)保有彈性的一種程式語言，不

同於在編譯期就決定程式執行方式的嚴謹作法，動態語言支援行為控制、執行期繫結與鼓勵快速開發並且回應需求等特色(Erdogmus, 2008)。PHP 是一種內建多樣化函數的動態語言，這些開放程式碼的函數提供了各式功能，可供開發人員快速開發網頁程式，搭配 Apache 網頁伺服器，可以快速生成使用者瀏覽的網頁。PHP 支援在多種平台上執行，完全免費且開放原始碼，讓開發人員能夠有更大的揮灑空間。

有鑑於此，本架構選擇以開放原始碼環境為主要的開發平台，開發環境選用 Linux，搭配 PHP 腳本語言(Scripting Language)與 Apache 網頁伺服器，運用網路服務技術來實作與服務提供商與資料中心的通訊機制，資料交換的格式採用 XML。皆以開放原始碼或標準為基礎進行開發。

4. 雲端運算服務導向架構

4.1. 雲端運算服務導向架構運作機制

本研究之雲端運算服務導向運作機制包括服務提供方、服務需求方與第三方平台，是原生的分散式架構，並保持服務雙方間呈現鬆散的關係；雲端運算服務導向架構平台係根據服務導向架構之分散式架構、關係鬆散的介面、開放的標準、定義明確的服務合約與服務可發掘並動態取得等技術特性所發展。本服務平台將透過開放標準的網路服務來達成溝通的功能，並藉由中心控管層來定義服務合約，決定服務的執行內容與流程；服務提供方與資料中心可以動態加入或移出本平台，一切服務執行的依據將由中心控管層的服務合約來決定，達成服務可發掘與動態取得的特性。

為在雲端運算基礎架構上，滿足服務導向架構的特性，本研究以拓展 MVC 架構建立一個新的系統設計架構，這個架構可分為應用介面層、中心控管層、服務供應層與基礎建設層。應用介面層是服務介面，由 Google 提供的 Gadget 負責扮演服務需求與供應雙方的橋樑；中心控管層決定服務路由，依照服務需求，組合不同的服務供應方與資料；服務供應層結合服務供應方與資料中心，以完成各種服務需求的實際應用功能；基礎建設層採用 Google 提供的應用服務環境，這個環境構築在雲端運算架構之上，服務提供者與消費者共處於雲端之中，必須藉由中心控管層進行服務整合。服務提供者於平台註冊後，即可加入服務市集，成為中心控管層的候選服務；使用者只需透過瀏覽器，即可使用各種應用服務，簡化服務消費流程。本研究架構圖請見圖 2：

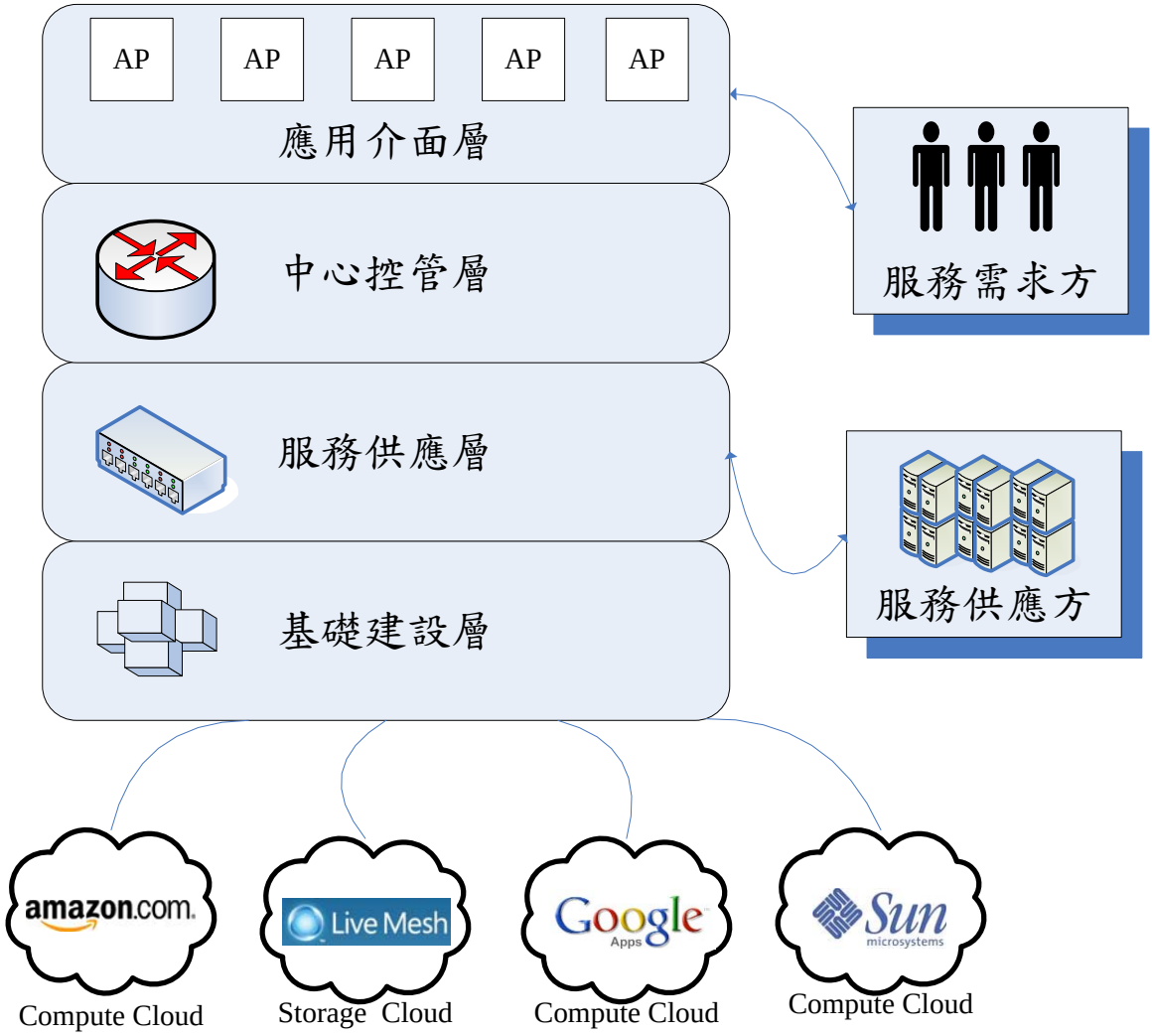

圖 2：雲端運算服務導向架構圖

4.2. 雲端運算服務導向架構平台

4.2.1. 應用介面層

應用介面層負責人機介面的呈現，以往的網頁應用程式開發工作，程式人員需同時負擔應用程式開發與介面美工設計的工作，讓降低專案開發的效率與品質。導入 MVC 架構後，雖然利用樣版(Template)的概念，將畫面與邏輯分開，但介面設計的可攜程度仍然很低，版面必須針對個別專案調整的情況也很常見，導致專案開發人員花費許多力氣在介面開發上，而無法專注於關鍵問題的解決。

Gadget 是一種小型應用程序，可嵌入網頁和其他桌面應用程序中。使用 Gadget 的方式嵌入網頁應用程式，可以輕易的將程式部署到各式網頁平台中，無論是 iGoolge、Google Sites 或一般網頁，都比過去減輕許多部署上的困難。這種往平台靠攏的設計思維，隨著各網路平台紛紛開放各式 APIs 後，開始讓許多開發者致力於發展單一功能的網頁應用，無須獨立創建嶄新的平台，而社群也能夠因為開發者能夠輕易的加入開發團隊，讓整個社群更為有活力，並且逐步演變成平台間的領導廠商。

服務需求方需取得 Google 環境的登入帳號，可以使用 iGoogle 的個人入口網站，或是如 Google Apps 的應用社群來註冊使用服務，當使用者需要使用本系統的服務時，只需要將服務需求，透過增加小工具的方式，新增到應用程式介面上，即可開始使用本服務平台所提供的應用服務。使用者無須安裝任何軟硬體，也無須作任何的設定工作，只需要具備有瀏覽器的電腦，連上網際網路即可使用服務，簡單、直覺且方便行動使用。個別服務會切分成單獨的 Gadget，使用者在使用上不容易混淆，可以用看待個別服務的使用觀念來著手進行，降低使用者混淆的程度，提供更友善的使用體驗。

本研究將 Gadget 視為客戶端與中心控管層溝通的介面，程式開發人員無須分心於版面設計與美工，只要專注於服務元件的撰寫，即可讓更多使用者輕易使用開發出來的服務。以應用服務角度的雲端運算定義而言，透過中心控管層決定服務邏輯後，可整合多方服務提供者與資料中心，結合大量服務群與資料群來完成服務的能力，發揮雲端運算架構的平台整合能力，讓服務消費者的消費流程更為簡易。

4.2.2. 中心控管層

中心控管層負責連接服務供應方與服務需求方，當客戶端在應用介面層上使用服務時，客戶端將透過 Gadget 發送需求(Request)到中心控管層的 WebServer，此時 WebServer 將把需求轉送到 Dispatcher 元件，依照服務邏輯的定義，決定此服務應執行的流程與所需相關資料；當決定執行流程與所需資料後，Controller 元件將在服務供應層中透過 WebService、Library 與 SoruceRequest 元件組合資源後，Service Provider 與 Data Center 會依照 Controller 要求的流程完成客戶端要求的服務，最後將服務結果透過 View 元件在客戶端呈現。所有候選服務資源皆為外在第三方伙伴所提供，各種服務與資料來源可以動態的在服務供應層新增或移除，中心控管層負責組合累積的服務資源以完成客戶端的需求。

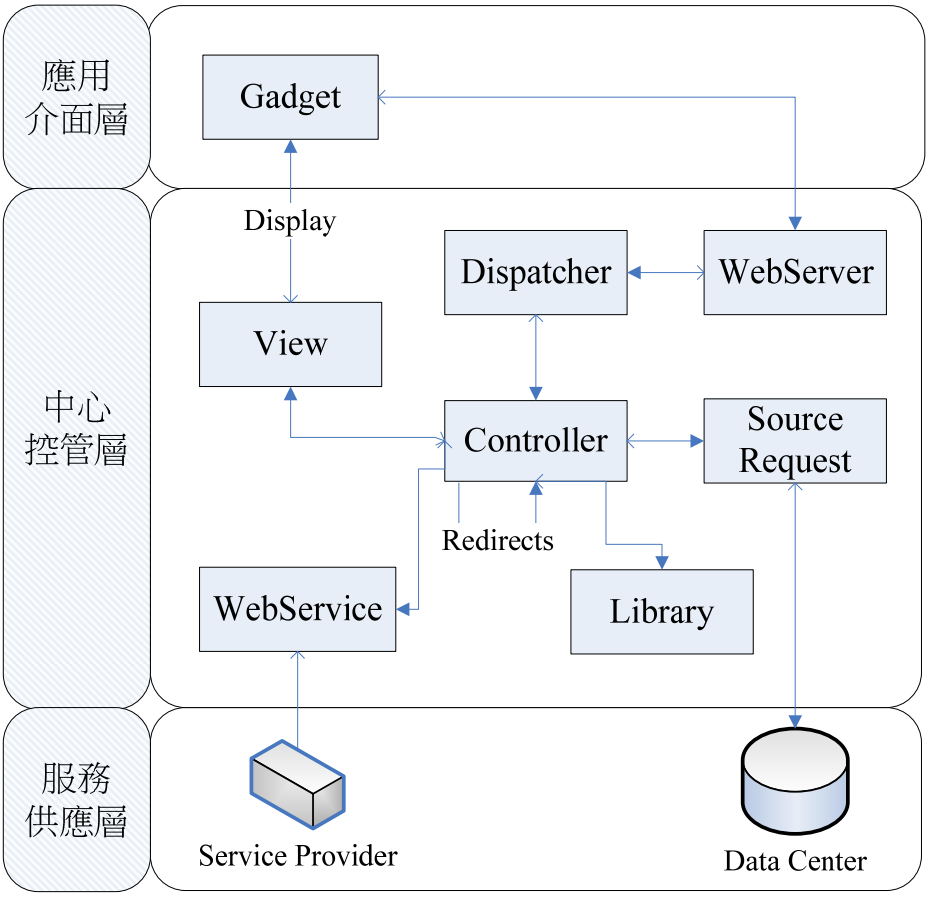

圖 3：雲端運算服務導向架構流程圖

中心控管層負責接收服務需求，定義出適當的流程後，決定結合相對應服務的方式，並且取得所需的應用程式與資料，在完成服務後，回傳給使用者。服務平台的系統架構，可參考下圖：

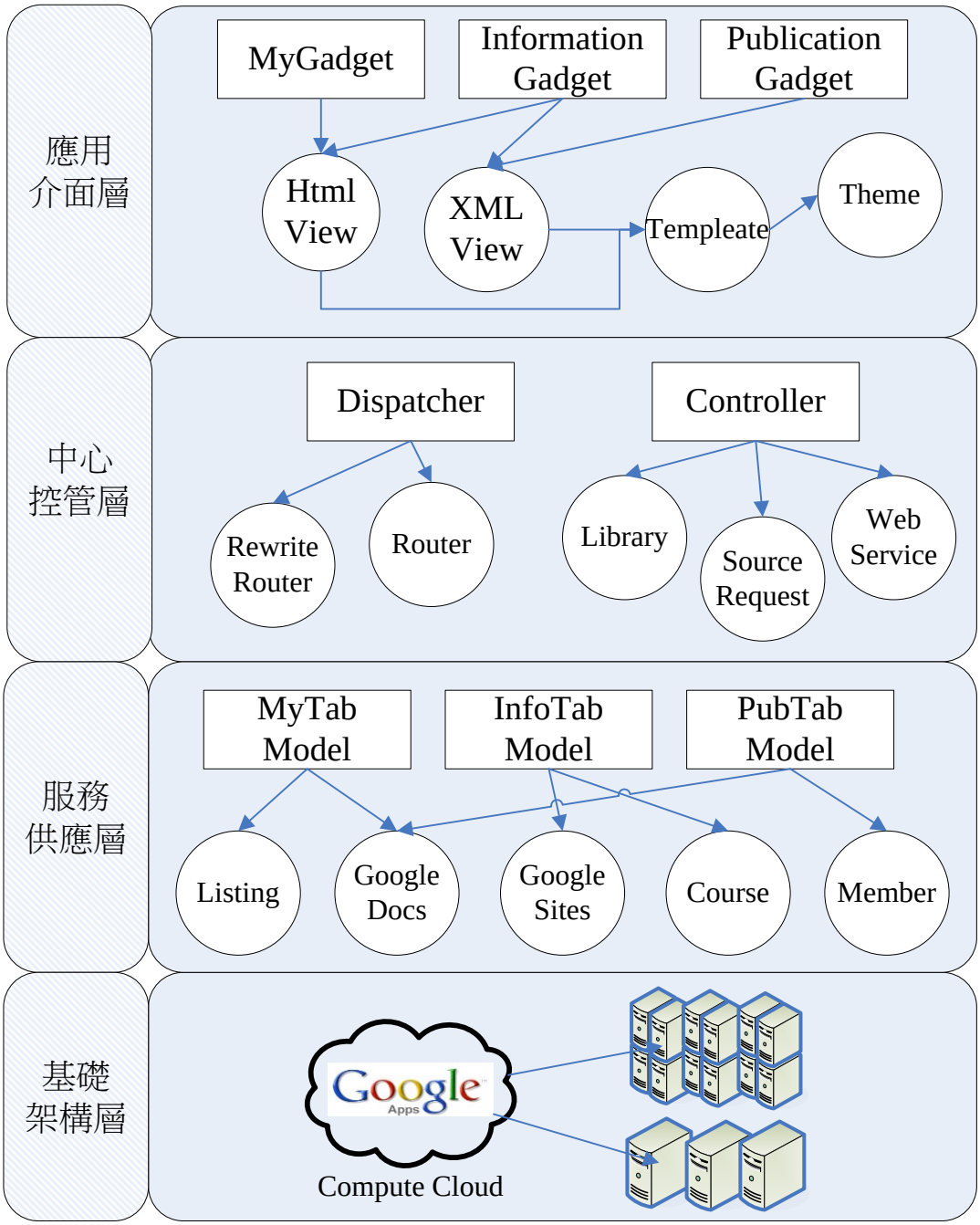

圖 4：服務平台運作機制

本服務平台利用 PHP 動態語言的 Late binding 與 Autoload 功能，設計出服務路由機制，客戶端加入服務 Gadget 後，本服務平台透過 XMLView 元件回傳 XML 格式的服務資訊，讓客戶端可以將小工具加入如 Google Sites 等 Web 服務介面。客戶端使用服務時，Gadget 透過 HTTP Get 發出服務需求，在服務平台接收到服務參數後，透過 Dispatcher 元件的服務路由機制決定對應執行流程，這個動作由 Router 與 RewriteRouter 兩個元件進行，將原本的服務需求重新導向到新的服務流程裡。服務流程的定義在 Controller 元件中進行服務串接，串接的服務存放於 Model 中，Model 的服務實作可以透過 WebService、Library 與 SourceRequest 等元件與實際服務提供者進行連接，並定義出服

務執行內涵，也就是進行實際的執行流程。最後的執行結果會引導(Render)到 HTMLView 元件，產生人機介面所需的 HTML 碼，結合顯示用的 Template 元件後呈現給使用者，換句話說，在程式邏輯的開發過程中，程式邏輯與介面呈現將交由不同階段完成，讓應用程式的開發能夠更有效率。

中心控管層連接服務供應方與服務需求方，將服務需求轉送到服務供應層，並依照服務邏輯定義出服務路由，滿足在服務導向架構對於軟體架構的要求。讓服務供需雙方位於分散式架構中，可發掘並且動態的取得服務，採用開放的標準並且以 Dispatcher 元件定義明確的服務合約，以達成服務流程控管；完成服務的執行流程後，將服務供應層執行完畢的服務結果回傳給應用介面層，善加利用雲端架構優勢以提升服務效率。換言之，中心控管層具備服務導向架構的優點，並能善加利用雲端運算的優勢。

4.2.3. 服務供應層

服務供應層可以輕易結合各式服務與資料來源。在服務部分，包含本地端所內建的函式庫(Library)與透過網路服務與第三方伙伴溝通的各式應用服務；資料來源包含提供存取介面的專業資料管理公司與客戶端的本地端資料庫。本研究之服務平台所提供的應用服務乃是透過不同的應用程式與資料所交互組成的，當匯集更多服務資源後，本服務平台能夠產生更大的加值效益，並隨著雲端運算架構環境持續發展，運算服務變的無所不在，服務供應層可逐步朝向『全球雲端服務交易市集』的特性發展，讓服務供應商間交換運算資源，以充分發揮雲端運算的平台特性。

應用服務提供者與資料中心所組成的服務供應層，皆透過網路服務的方式進行溝通。服務提供者可以是任何平台業者所提供，因為網路服務跨平台的特性，本系統架構可以輕易的與不同類型的網路服務結合；至於資料中心則扮演管理資料的角色，服務提供者並不直接擁有資料，而是必須在取得客戶端授權後，向資料中心索取適當的資料，這可以確保資料的隱私能夠得到最大程度的保障；資料中心也可與客戶端的私有資料庫結合，透過中台的授權機制，仍可在保有資料隱私的情況下，享有雲端運算與本架構所帶來的優點。

服務提供方的連接方式可分為二種，第一是透過網路服務的方式，與服務提供者與資料中心進行聯繫；第二是透過本地端開發的 APIs 函式庫與服務進行聯繫。以透過網路服務來取得服務而言，服務提供者只需要在本平台註冊後，並同時提供網路服務介面，即可成為本平台的候選服務，在使用者發出服務需求後，將可以直接透過本平台進行介接，完成服務需求。於資料中心的連接也透過網路服務進行連接，但概念上有些不同，本平台將資料中心定位為類似目錄服務，當發出要求資料的需求後，資料中心將會發出存取資料相關的連線資訊，例如連接位址、參數等，服務平台將會透過這些資料，進行與資料庫的連線工作，取得資料以完成服務，也就是說，資料中心代表資料儲存的導引者，這個導引的對象，可以是網路上任何一個資料庫，包含專業資料庫代管公司，

或是公司內部的資料庫系統等，都可以透過資料中心作一個轉接的動作，而能夠為本服務平台所用。

如圖 5 所示，客戶端將需求交由中心控管層進行處理，第三方服務提供商所提供的應用服務與不同的資料來源，將在中心控管層結合，並且最終完成服務。服務提供商與中心控管層本身並不直接擁有資料，而是透過客戶端的授權，取得服務所需的資料，再透過服務提供者的運算資源完成服務。保有使用者對於服務流程與資料控管的權力，並避免企業在導入雲端架構時，所衍生出的資訊安全與隱私等疑慮。

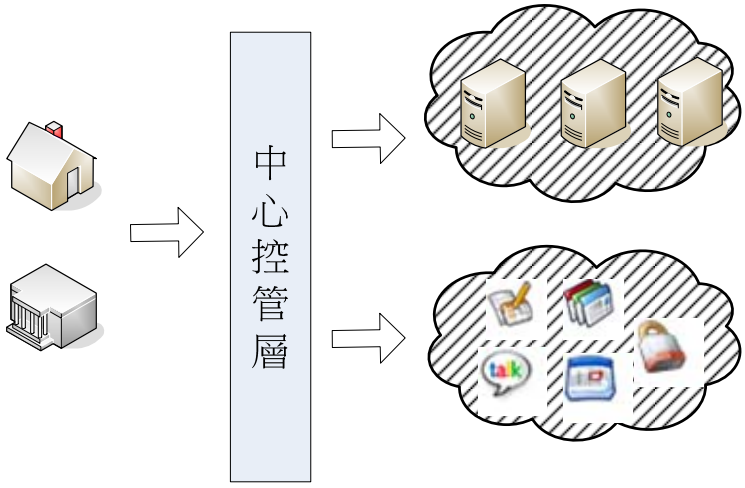

圖 5：應用服務與資料分離示意圖

4.2.4. 基礎建設層

基礎架構層採用 Google 應用服務環境，透過 Gadget 連接本平台所整合的應用服務，並部署於 Google 的應用環境之中。本應用環境同時提供了 Gmail、Google 日曆、Google Talk、Google Docs、Google Sites 以及網頁建置服務等應用程式，讓服務間的溝通、單一窗口登入與協同合作等功能轉由 Google 來提供；搭配 Google App Engine 更可以充分利用 Google 運算資源。

在過去的網頁程式開發，如使用者的帳號管理、登入認證與溝通機制等，都需要花費一定的開發成本才能提供，但是這往往跟應用程式本身的功能沒有太大的關係，也就容易造成開發資源的浪費，Google Apps 同時完成了會員登入管理與相關的溝通機制，讓開發人員能夠省去繁複的認證機制，將應用程式的開發單純到功能本身，減輕應用程式開發的困難度，並縮短開發時間。網頁程式常常使用到的 Email 與即時通訊等機制，Google Apps 也提供了相對應的資源，開發人員不需要再分心開發這些基礎服務，而能專心於程式功能本身的開發工作，讓開發工作的進行更有效率。

IBM、HP、Intel、Yahoo、Sun 與 Microsoft 等公司也相繼投入雲端運算相關基礎設施的研發工作，透過本服務平台，可以輕易的利用各陣營所提出的基礎架構與運算資

源，並在基礎建設層中進行連接；亦即，本研究以 Google 的基礎應用環境為例，示範與雲端平台業者共生的整合營運模式，在相關雲端基礎設施益發茁壯的情況下，本研究之服務平台可扮演雲端運算服務市集平台的角色，結合不同服務供應方與雲端基礎建設供應方，提供強大的服務能力，並且充分發揮雲端運算與服務導向架構的優勢。

4.2.5. 應用範例說明

本研究可應用於發展數位學習(E-Learning)社群網站，藉由本研究之服務平台，發展大學教學社群網站，包含社群網路的互動機制、教師評分與學員表現等，以說明本研究所提出之架構符合憑藉在雲端運算基礎架構上的服務導向架構原則。依照前述章節所提出的架構，系統分為應用介面層、中心控管層、服務供應層與基礎建設層，以下將分別由各層別角度說明應用範例。

社群成員可依本身需求在應用介面層，新增所需服務，例如使用者新增了『Information Gadget』以取得學習平台中各學員修課主題與課程資訊等資料。使用者使用此服務時，將透過 Gadget 對中心控管層發出需求，依照 Dispatcher 元件對 Information 服務的服務邏輯，將需要結合 Google Docs、Google sites 等應用功能，再配合學員資料與課程資訊等資料來源，以完成服務內容。當確定服務邏輯後，Controller 結合處於服務供應層中的 Google 應用環境相關服務，並透過 SourceRequest 取得相關資料來源，以這個應用而言，學員、課程與應用程式間三者的結構與應用，將在 Library 中進行實作，換言之，也就是透過 InfoTab Model 來完成實際的服務執行工作。而後的服務執行結果，透過 HtmlView 元件，依照 Template 定義的呈現模式，將結果回傳給服務 Gadget，使用者不需知道實際執行服務的地點等資訊，只需專注於服務內容即可，符合雲端運算架構的概念。（請參考圖 4）

應用介面層是服務介面，本研究採用 Google 提供的 Gadget 負責扮演應用服務與客戶端間的橋樑。Gadget 可以輕易的嵌在各式網頁中，本研究利用 Sites 作為使用者的入口網站(Portal)，使用者可以在自己的 Sites 中，新增由本平台提供的各項 Gadget，此時的頁面扮演如服務儀表版般的角色，讓花費在開發應用介面的負擔能夠減輕，將開發重心著重在應用服務上。當數位學習社群服務平台加入管理個人資訊的 My Gadget 與發表個人意見的 Publication Gadget 之後，一個數位學習社群網站的雛形已經可以運作，並且應用於實際的教學環境之中；與以往的開發經驗相比，結合本服務平台與 Google 提供的應用服務環境，可以節省大量的建置時間與降低開發成本，並善用可用的服務資源以提升服務能力與品質，將發展重心轉為更有價值的社群經營。

5. 結論

本研究在雲端運算基礎之上發展滿足服務導向架構特色的服務平台，以解決雲端運算所衍生出內外部資源有效配置的問題，與企業導入上的資安疑慮。本研究結合雲端運

算與服務導向架構，發展出一個四層式服務平台，包含服務介面層、中心控管層、服務供應層與基礎架構層。

應用介面層專注於人機介面的呈現，以 Gadget 負責畫面呈現，將版面與程式邏輯分開，充分利用開放介面的優勢，降低服務介面開發的負擔，讓開發者能專注於關鍵問題的解決。

中心控管層係基於服務導向架構之設計，能連接服務供應方與服務需求方，將服務需求轉送到服務供應層，並依照服務邏輯定出服務路由，可使用外部雲端運算與資料儲存之資源，並達成服務流程控管的需求；在完成服務的執行流程後，將服務供應層執行完畢的服務結果回傳給應用介面層。中心控管層具備服務導向架構的優點，並能善加利用雲端運算的優勢。

服務供應層結合各式應用服務與資料來源，透過內建函式庫與透過網路服務與第三方伙伴溝通，包含資料中心與服務提供者。服務供應層藉由將運算與資料分離的方式，讓服務需求者保持對資料的控管權力，以解決企業導入雲端架構時對於資安的疑慮；在匯集越多服務資源後，能產生更大的平台加值效益。

基礎架構層可充分利用各雲端業者所提出的服務架構、運算以及儲存資源。本研究係採用 Google 應用服務環境，透過 Gadget 連接本平台所整合的應用服務，並部署於 Google 的雲端運算環境之中。基礎架構層未來可導入更多雲端平台業者，依照個別平台的優勢與特性，充分發揮雲端運算架構的效益。

最後本研究藉由數位學習社群網站之應用，以說明本服務平台之可行性，以及整合外部雲端運算服務的優越性。未來研究我們將探討如何將本服務平台發展成雲端運算架構交易市集。

參考文獻

1. Amazon.com. (2007). Amazon.com Announces Third Quarter Sales up 41% Year over Year - Raises Financial Guidance - Expects Record Holiday Season. from http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=1066358&highlight=Amazon%20Web%20Services
2. Bieberstein, N., Bose, S., Fiammante, M., Jones, K., & Shah, R. (2005). Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap. *IBM Press Publishing*.
3. Boden, T. (2004). The grid enterprise — structuring the agile business of the future. *BT Technology Journal, 22*(1), 107-117.
4. Boss, G., Malladi, P., Quan, D., Legregni, L., & Hall, H. (2007). Cloud Computing. from http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/hipods/Cloud_computing_

wp_final_8Oct.pdf

5. Brereton, P., Budgen, D., Bennnett, K., Munro, M., Layzell, P., MaCaulay, L., et al. (1999). The Future Of Software. *Communications of the ACM, 42*(12), 78-85.
6. Buyya, R., Yeo, C. S., & Venugopal, S. (2008). Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*.
7. Conlon, S. J., G.Hale, J., Lukose, S., & Strong, J. (2008). Information Extraction Agents For Service-Oriented Architecture using Web Service Systems: A Framework. *Journal of Computer Information Systems, 48*(3).
8. Davis, N., Demetriou, G., Gaizauskas, R., Guo, Y., & Roberts, I. (2006). Web Service Architectures for Text Mining: An Exploration of the Issues via an E-Science Demonstrator. *International Journal of Web Services Research, 3*(4), 95-114.
9. Erdogmus, H. (2008). So Many Languages, So Little Time. *IEEE Software, 25*(1), 4-7.
10. Feuerlicht, G. (2006). Enterprise SOA: What are the Benefits and Challenges? *Systems Integration*.
11. Google.com. (2007). Google Apps. from http://www.google.com/apps/intl/en/business/index.html
12. Google.com. (2008). What are Gadgets? , from http://code.google.com/apis/gadgets/
13. Gray, P. (2007). SOA - Moving From Doing it Yourself to Getting Others to Do it for You. *Information Systems Management, 24*(1).
14. Group, W. C. W. (2004). Web Services Architecture. from http://www.w3.org/TR/ws-arch/#whatis
15. Huang, M. F. (2006). *A Study of Multi-Business Service Oriented Architecture.* National Chengchi University, Taipei.
16. McCoy, D. W., & Natis, Y. V. (2003). Service-Oriented Architecture: Mainstream Straight Ahead. Retrieved August, 2008, from http://www.gartner.com/pages/story.php.id.3586.s.8.jsp
17. Morrison, S. (Aug 20, 2008.). Cloud Computing' Makes Gains. *Wall Street Journal*,
18. Natis, Y. V. (2003). Service-Oriented Architecture Scenario. Retrieved August, 2008, from http://www.gartner.com/DisplayDocument?doc_cd=114358
19. Pallos, M. (2001). Service-Oriented Architecture: A Primer. *eAI Journal*.
20. Reenskaug, T. (2003). The Model-View-Controller (MVC)Its Past and Present from http://folk.uio.no/trygver/2003/javazone-jaoo/MVC_pattern.pdf
21. Salesforce.com. (2007). Salesforce.com and Google Form Strategic Global Alliance. from http://www.salesforce.com/company/news-press/press-releases/2007/06/070605.jsp
22. Stamford. (2008). Gartner Says Cloud Computing Will Be as Influential as E-business. Retrieved August, 2008, from http://www.gartner.com/it/page.jsp?id=707508
23. Yan, N., Leip, D., & Gupta, K. (2005). The use of open-source software in the IBM

corporate portal. *IBM Systems Journal, 44*(2), 419-425.

24. Zhang, L.-J., & Jeckle, M. (2004). Convergence of Web Services and Grid Computing. *International Journal of Web Services Research, 1*(3).

A Study of Cloud Computing Service Oriented Architecture

Jiann-Min,Yang[1] Chun-Hua,Tasi[2] Chun-Hung,Liu[2]
[1] Dept.of Management Information Systems, National Chengchi University
jmyang@mis.nccu.edu.tw
[2] Dept.of Management Information Systems, National Chengchi University
96356027@nccu.edu.tw
[3] Dept.of Management Information Systems, National Chengchi University
96356036@nccu.edu.tw

Abstract

Cloud Computing heralds an evolution of business, this creates a research question on how to integrate enterprise information service and external Cloud Computing resource. Information service big players like Google, IBM and HP, providing powerful Cloud Computing ability and application environment. However, when enterprise adopt Cloud Computing architectures proposed recently, and transfer applications and data to external platform arouses the problems of management authority shifting and information security.

We proposed a 4-layer service platform that incorporates Google Cloud Computing environment and Service Oriented Architecture in a sense to provide new paradigm of system development. Service Interface Layer utilizes the advantage of open platform and reduces the burden of interface development. Control Center Layer connect service providers and service requesters, while handling service flow and using the resource of external cloud computing and data center. Service Provider Layer reserves the management authority and ownership of the data resources. Infrastructure Layer assemble service framework, computing ability and storage resource that are provided by Cloud Computing providers. We believed that with the 4-layers service platform proposed in this paper, one can leverage the advantages and merits of Cloud Computing architecture.

Finally, by developing an E-learning community website based on proposed platform, we are able to justify the service platform's feasibility and its advantage with the use of cloud computing.

Keywords: Cloud Computing、Service Oriented Architecture、MVC framework、Google computing、E-learning community

.