東海大學資訊工程研究所

碩士論文

指導教授：朱正忠 教授

具服務品質保證之企業雲端計算中介軟體設計與實作

The Design and Implementation of Middleware for Quality of Service Assurance for Enterprise Cloud Computing (QoSAECC-ME)

研究生：李羿杰

中華民國一百零三年一月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 李 羿 杰 所提之論文

具服務品質保證之企業雲端計算中介軟體設計與實作

經本委員會審查，符合碩士學位論文標準。

學位考試委員會
召 集 人 何信瑩 簽章

委 員 楊朝棟

蔡志宏

[illegible]

指 導 教 授 朱正忠 簽章

中華民國 103 年 1 月 3 日

摘要

在企業雲端運算(ECC)中存在諸如多重租約、跨層服務的組合(SaaS, PaaS, IaaS)以及使用者需求與服務層級協議(SLAs)的多重限制等議題，為了確保企業雲端運算的服務品質(QoS)與效率，以中介軟體(Middleware)來支援 ECC 的服務組合與監控就變得相當重要。本論文的目的是設計與開發在 ECC 上可以自動化管理 SaaS、PaaS 與 IaaS 服務提供之資源調監控功能的中介軟體。

本論文設計及實作中介軟體來提供一個簡單、具成本效益以及安全的方式來存取雲端環境的服務，透過中介軟體串接底層的分散式叢集系統、應用軟體、提供服務的平台與要求服務的用戶端使用者。雲端中介軟體的目的在於提供底層技術與周遭環境的完全透明化，而服務導向架構(SOA)承繼基於元件的方法達成分散式處理以增加系統效能與易於管理。除了傳統的 SOA 針對 SaaS 領域外，將更進一步整合作業系統、底層硬體與測試部份。本論文使用代理人技術以監控服務品質的需求與 SLAs 的一致性。提供一個有效且必須的服務給用戶端使用者，中介軟體提供的不僅是支援在複雜的雲端運算環境下軟體的運行，並且必須藉由提供多樣的功能來增進應用軟體的開發。

關鍵字：中介軟體、服務導向架構、代理人技術、企業雲端運算、監控技術

Abstract

Enterprise Cloud Computing (ECC) has the complicated issues of the multi-tenancy,cross layer service composition (SaaS, PaaS, IaaS), multiple constraints from user requirements and service level agreements (SLAs). To assure the quality of service and effectiveness of ECC, a middleware to support the service composition and monitoring in ECC is highly important. The goal of this project is to design and develop a middleware for Enterprise Cloud Computing(ECC) which can automatically manage the resource allocation of services from SaaS, PaaS, and IaaS.

The middleware provides an easy, cost-effective, and secure way to access services from cloud environment, which may involve distributed cluster systems, the application software, and platforms to the end users. The purpose of the middleware for cloud computing is to provide complete transparency of the underlying technology and the surrounding environment. The Service Oriented Architecture (SOA) extends component-based approach to achieve distributed processing, to improve the system efficiency, and easy to manager. In addition to the traditional SOA which is focus on the domain of Software as a Service (SaaS), operating system and infrastructures and even testing part will be integrated. This project employs the agent technology to the monitoring of requested QOS requirements and Service Level Agreements (SLAs).

Keywords: Middleware, SOA, Agent, Enterprise Cloud Computing, Monitoring

目錄

圖目錄

第一章 緒論

1.1.前言

現今許多企業與機構面臨了迅速增加改變的商業需求，IT 產業已由產品導向方式轉變為服務導向，為了解決這些需求，企業必須花費大量的時間與人力在解決這些問題。然而要在如此龐大的系統之下要維持良好的品質相當困難，因此必須導入服務導向架構 Service Oriented Architecture (SOA)的技術讓系統效率更高、更容易維護，SOA 可以提供一個靈活的選擇，經由不同的服務提供商提供的多種服務元件組成企業所需要的系統或軟體，讓系統軟體建置時間大幅縮短，有效地減少開發成本，更可以提高企業的競爭力。

1.2.研究動機

目前於雲端技術研究，如何將 SOA 架構與 Cloud 進行整合，已有學者進行 SaaS 與 SOA 應用研究[1][2]，而在雲端計算中對於平台 PaaS 以及基礎設施 IaaS 中，仍有許多如何提供服務所帶來更複雜的問題；因此本論文目標是提供軟體層、硬體層、作業系統、中介軟體服務與測試驗證等服務透過虛擬化，建立資源池概念方式，將服務資源整合，當這些服務整合之後，產生諸如多重管理、跨層服務的組合(SaaS, PaaS, IaaS)以及使用者需求與服務層級協議(SLAs)的多重限制等議題；因此

為了確保服務品質(QoS)與效率，如何有效利用中介軟體來支援企業雲端運算的服務，提升系統效能管理與系統監測服務可提升企業在雲端導入過程，提高服務的可靠度與穩定度，隨著雲計算的普及，雲端服務已經成為應用程式平台，使用者可以建立新的應用程式組合他人所提供的功能。服務混搭組成的動態分佈，雲服務，提供更複雜的任務，架設大型的互聯網路應用提供了一個有吸引力的方式。雲端服務的混搭極具挑戰性如何找到同時滿足應用程序的資源需求是問題之[3]。

1.3.研究目的

本論文提出進行針對企業所需的雲端服務平台，研究發展可滿足「具服務品質保證之企業雲端計算中介軟體」，利用中介層建立相關服務框架，提供的雲端系統在開發、運行、管理和監控的環境的串聯，透過中介軟體對雲端平台環境進行系統監控與分析，滿足在 SaaS 服務層對於虛擬化系統硬體資源需求，以及有效記錄設計規劃已進行服務的資訊系統狀況，透過即時服務效能監控，滿足企業對於資訊服務的可用性與可靠度，進而達到良好的服務品質與滿足企業用戶需求。

我們透過服務導向架構 Service Oriented Architecture(SOA)的概念，建構企業雲端計算的基礎架構[2]，服務導向架構也在近幾年興起，透過模組化的方式架構系統，實現分散式的處理，並易於管理、降低複

雜度，主要概念是針對企業需求組合成一組軟體元件，藉此讓異質系統整合變得容易，並也提升了系統的敏捷度(Agility)[4][5][6][7]，導入新專案的速度將可加快，從而協助公司的營運，而中介軟體便是服務導向架構中用來串接應用軟體和底層系統不可或缺的角色。但傳統 SOA 著重於軟體層面的服務提供，並未評估量化可確保服務的穩定度與系統的可用性等，因此為了滿足企業對於資訊服務系統服務品質要求，我們結合了服務層級協議 Service Level Agreement(SLA)，並把企業的需求建立不同的模組，而在服務層級協議部份，透過不同需求如：增加了效能或是回應時間等，利用中介層告知基礎層所需增加如硬體資源或作業系統環境等，滿足軟體服務系統可彈性並即時獲得滿足的基礎資源。

在本論文中我們利用中介軟體連接不同應用系統與軟體元件，建構使用者與系統之間的橋樑，最主要的目的在提供完整的後端服務與所處環境的透通性，不但要能支援軟體可以在複雜的計算環境中的執行，而且它必須能讓程式設計師可以容易的開發各種不同的應用程式，提供完整的 API 支援、資料管理。除此之外為了維持良好的服務品質(Quality of Service)，必須設計與服務層級協議 SLA 進行協商的機制，包含協商溝通的參數與交換格式等規格，最後產生一套正式共通的 SLA 文件，為此我們使用協商代理人 Agent 中介服務框架，針對不同

層面(IaaS, PaaS, SaaS)，監控底層伺服器叢集的特徵與負載、資料庫活動、服務或應用程式對於資源的消耗等，動態分析並做出資源調配的處理，去幫助在 SLA 與 Services 之間進行自動化協商一致，確保服務能提供符合預期的品質。

1.4. 章節安排

論文的章節安排分述如下：

第二章 背景知識與相關研究，此章節說明與本論文相關的研究。

第三章 此章說明本論文使用之方法。

第四章 系統設計實作與研究案例。

第五章 結論與未來方向。

第二章 背景知識與相關研究

2.1. Middleware

在雲端運算技術中，Middleware[8][9][10][11][12][13][14]位於服務與伺服器叢集之間，提供管理和服務。本研究所開發的中介軟體之研究著重在雲端運算環境中的系統整合並提供中介前置作業的服務，為應用提供統一的標準化程式介面和協定，將底層硬體隱藏起來、整合作業系統和網路由於這獨有之特性，軟體中介層能夠整合起具安全性普及計算環境的系統，以達到整合及管理中介體之應用程式之發展。如：

1. 一致的應用程式發展之支援。
2. 使用者管理－使用者身份驗證、許可、定制管理。
3. 資源管理－負載均衡、資源監控、故障檢測等。
4. 安全管理－身分驗證、存取授權、安全審計、安全防護等。
5. 映射管理－映射管理、佈署、管理。

2.2. SOA(Service Oriented Architecture)

SOA 服務導向架構是一種新興的系統架構模型，主要概念是針對學校或企業需求組合而成的一組軟體元件。組合的元素通常包括：軟體元件、服務及流程三個部份。當學校或企業面對外部要求時，流程負責定義外部要求的處理步驟，服務包括特定步驟的所有程式元件，

而軟體元件則負責執行工作的程式[4][15][16][17]。

從分散式元件架構到 SOA 概念上，SOA 如同物件導向、軟體元件等軟體技術一般，運用小的零組件組合成應用系統，但 SOA 強調的是如何將彼此關係鬆散的應用系統功能元件在網路上發行、組合及使用。

SOA 具有下列技術特性：

1. 分散式架構(Distributed)

分散式架構(Distributed)－SOA 的組成元件是由許多分散在網路上的系統組合，可能是區域網路，也可能是來自廣域網路。例如網路服務技術(Web Service)就是運作 HTTP 來相互連結的 SOA。如此的作法，也使得網路服務技術很快地就成為能夠支援所有網際網路系統平台都能使用的技術[18]。

2. 關係鬆散的界面(Loosely Coupled)[6]

傳統的系統主要是將應用系統功能需求切割成相互關聯的零組件：模組、物件或元件，開發者要花費極大的心力了解零組件是如何設計及使用，以確保不會違反零組件連接關係限制。如此一來，若要以不同零組件替換原始設計，就成為一件困難的事。SOA 的作法是以界面標準來組合系統，只要符合界面要求，零組件可以任意替換，大幅提高系統變更的彈性度。

3. 依據開放的標準(Open Standard)

使用開放標準是SOA的核心特色,過去的軟體元件平台如CORBA、DCOM、RMI、J2EE 採用專屬協定作為元件連結的規範，使得不同平台的元件無法相通。SOA 則著重於標準與互動性，將可避免不同平台(.NET Web Service 與 Java Web Service)開發程式間相互整合的困擾。

4. 以流程角度出發(Process Centric)

在建構系統時，首先了解特定工作的流程要求，並將其切割成服務界面(包括輸入與輸出資料格式),如此其他的發展者就可以依據服務界面開發或選擇合適的元件來完成工作。

SOA 能提供的三大優點：

1. 通用性(Near-universal Connectivity)

可減少各個技術整合造成的複雜度以及為了整合而存在的技術量。

2. 再生性(Reusable)

所採用的 MDA(Model Driven Architecture)方式可以增加 Component Reuse 的便利性。

3. 適用性(Adapatability)

可以提供整合起不同語言所開發的各個 Application 之環境。

圖 1 為根據 IBM 所提出的 Open System Standards 的架構。

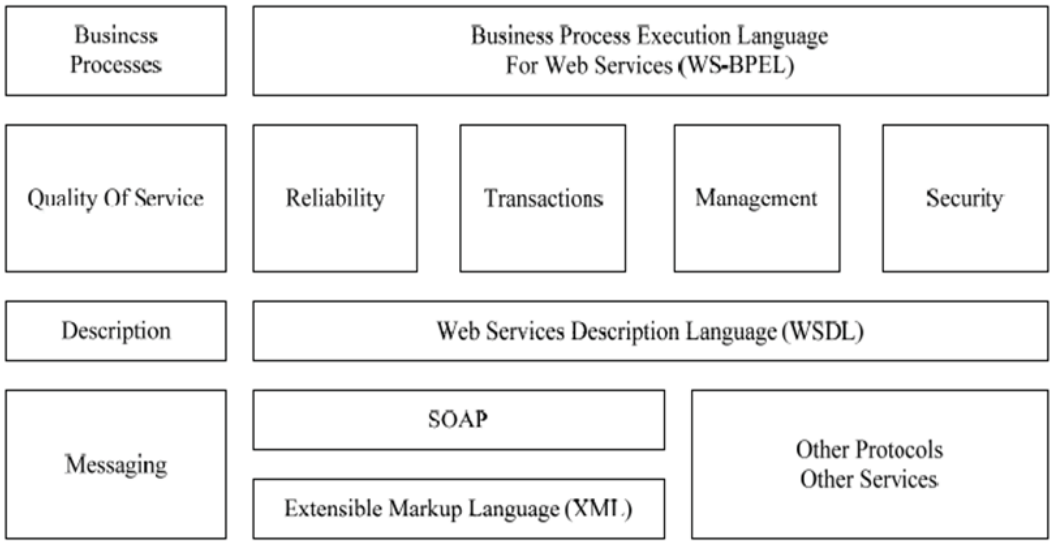

圖 1　Open System Standards 的架構

2.3. Web Service

論文由於牽涉到不同的裝置與網路環境，地區資料集中站(Local Data Collection)[19][20][21]面對所取得不同性質的資料以及使用的各種設備，假如直接與其連接，則需要定義出為數眾多的介面來接收資料，使得此集中站過於複雜，增加系統的難度。BEA Aqua Logic Service Registry 通過一個可設定的 Business Service Console 來支援業務服務生命週期，從而可以針對安全性、可延展性和可靠性，支援並發佈 SOA 資訊和進階功能。對於支援和發佈，它提供了 SOA 資源和發佈精靈之間的對應。對於設計和運行時發現，它提供了安全服務資訊流覽器、變更通知和標準的 UDDI 資料讀取。對於管理，它提供了複製、服務質量(QoS)管理資訊的對應和先進的業務服務分類管理。對於 SOA 治理，它提供了核心和生命週期服務，包括業內最強有力的安全性和批

准流程支援。

A.SOAP(Simple Object Access Protocol)簡易物件通訊協定[22]：一種以 XML 為基礎的通訊協定，其作用是編譯網路服務所需的要求或回應後，再將編譯後的訊息送出到網路，簡單來說就是應用程式和用戶之間傳輸資料的一種機制。SOAP 是一個獨立的訊息，可以獨自運作在不同的作業系統與網路上面，例如在微軟的 Windows 或 Linux 的建構下運作，並可以使用各種不同的通訊方式來做傳輸，例如 SMTP、MIME 或是 HTTP 等。

SOAP 1.2 版中，包含了一個用於簡化網路的工具，這個工具擁有許多之前未有的工具，例如可讓開發者建立管理 SOAP 訊息規則的「處理模型」，以及包含簡易管理大量的 XML 文件檔功能。SOAP 的架構為：Envelope、Header、Body 和 Fault 四個部份；其組織架構是與 XML 的語法相結合應用，換句話說 SOAP 是由 XML 語法所撰寫而成。SOAP 不但可以在不同的網路上運作，更可以在不同的網路間作傳輸，SOAP 可以透過 HTTP 發送訊息，再透過 TCP、MSMQ，最後由 SMTP 收到訊息，途中可以透過四個不同的傳輸點傳達訊息。由此我們可以見到 SOAP 的透通性與實用性，遠比一般的通訊協定更為有彈性，SOAP 之結構圖如圖 2。

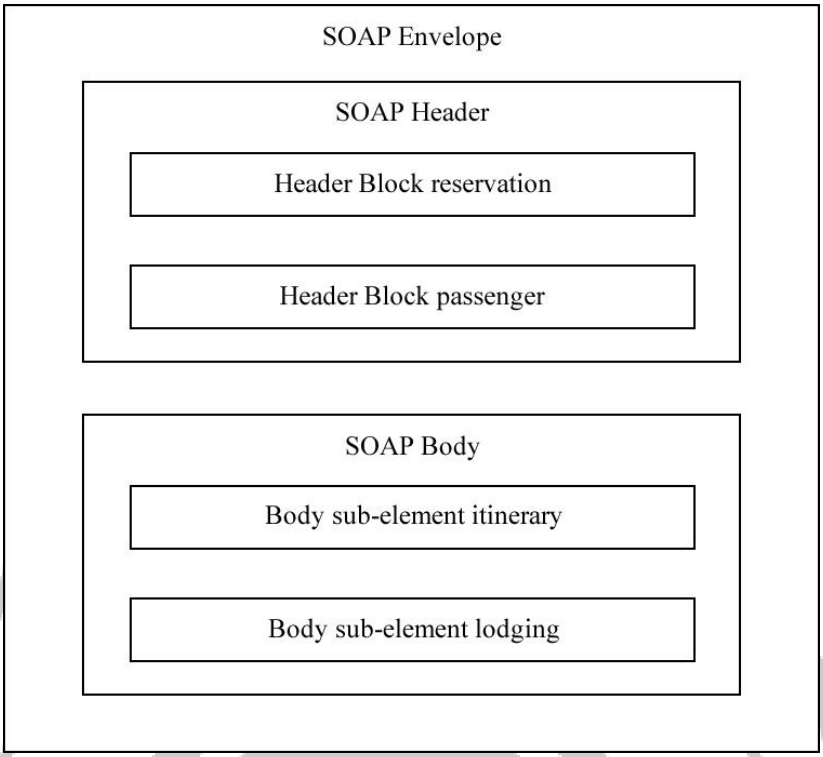

圖 2　SOAP 規範的結構及使用方式

B.WSDL(Web 服務描述語言)[23][24]：WSDL(Web 服務描述語言)是為描述 Web 服務發布的 XML 格式。2.0 版將被作為推薦標準(recommendation)(一種官方標準)，並將被 W3C 組織批准為正式標準。在諸多技術文獻中通常將 Web 服務描述語言簡寫為 WSDL。WSDL 描述 Web 服務的公共介面。這是一個基於 XML 的關於如何與 Web 服務通訊和使用的服務描述，也就是描述與目錄中列出的 Web 服務進行交互時需要綁定的協議和信息格式。通常採用抽象語言描述該服務支持的操作和信息，使用的時候再將實際的網路協議和信息格式綁定給該服務。

單來說，WSDL 就是 Web Service 的 IDL，Web Service Providers 利

用 WSDL 將其提供服務的相關訊息對外公佈。以下是整合 WSDL、SOAP、及 UDDI 描述的內容和 WSDL、SOAP、及 UDDI 整合的參數，WSDL 的特性如下：

1.描述 Service Providers 所提供的服務名稱，與該服務所能提供的 Operations。

2.描述服務需求者如何與 Web Service 溝通，如何叫用各項 Operations。定義內容包含傳輸協定、參數等。WSDL 使用下列 XML Elements 定義網路上的 Web Service

(A)Type:定義服務要求者與提供者之間，可使用溝通的參數類型。

(B)Message:定義輸入和輸出訊息由哪些參數組成。

(C)Operation:描述該 Web Service 提供的服務項目。

(D)Port Type:描述此 Web Service 所有 port 提供的 Operation 集合。

(E)Port:每一個 port 代表一個 Web Service 的存取點，每個 port 會明確指定 Binding 的方式。

(F)Binding:定義某個特定服務 port，所要求溝通的特定協定如 SOAP、HTTP、MIME 與資料格式定義。

(G)Service:為 WSDL 描述的 Web Service 集合。詳細運作關係如圖 3 所示。

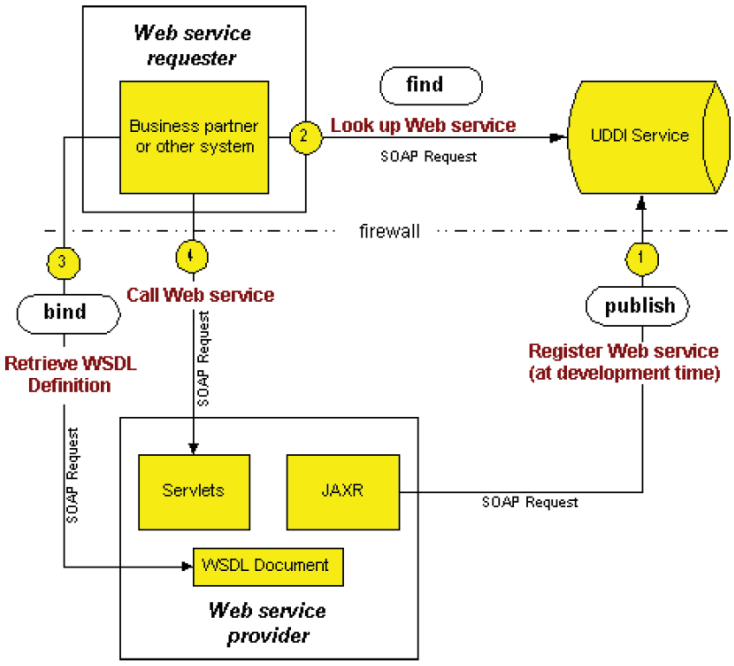

圖 3　How WSDL Works[23]

C. UDDI Services[19]：

通用描述、探索與整合(UDDI)是發佈及尋找網頁服務相關資訊的工業規格。Windows Server 2003 系列包含有「UDDI 服務」，這是個選擇性元件，可提供企業內部或在生意夥伴間使用的 UDDI 功能。UDDI 服務是標準型的 XML 網頁服務，可讓企業程式開發人員直接透過其開發工具及商務應用程式來有效地公佈、發現、共用及重覆使用網頁服務。UDDI 服務建立在 Microsoft .NET Framework 之上，是個可靠且可延展的解決方案，能以企業技術及工具提供簡易的整合。「資訊技術(IT)」管理員可以提昇對標準分類配置、Microsoft SQL Server 及 Active Directory 驗證的原始支援。作為 UDDI 版本 1.0 及 2.0 相容於 API 的服

務，UDDI 服務也包括已轉譯成所有 Windows Server 2003 系列所支援之語言的網頁介面。

UDDI Server 提供處理 UDDI Client 的管理服務要求，包括：

(1)服務註冊

(2)服務查詢

(3)服務變更

(4)WSDL 有效性檢驗 UDDI Client 提供使用者管理服務的瀏覽器操作介面，包括：

(1)服務註冊介面

(2)服務查詢介面

(3)服務變更介面

(4)服務有效性確認介面最後，會將功能需求分配到 2 個子系統，分別是：

(1)UDDI 伺服器子系統(UDDI Server)

(2)UDDI 客戶端系統(UDDI Client)

2.4. 企業雲端服務(Enterprise Cloud Service)

傳統的 IT 企業可以完整的控制並觀看他們的系統與服務，所有的元件與系統都是可以存取、分析、測量具有完整的權限，但轉型成為

雲端架構後，原有的系統可見度(Visibility)與控制權限將無法直接獲取，雲端供應商與企業之間的溝通，必定形成一個議題，可能的戰還包含基礎建設的變更、虛擬化技術的控制、安全性、服務層級協議(Service Level Agreement, SLA)、資料管理、流程的控制、雲端服務的管理，當企業由原本的服務導向架構轉型時，更多的挑戰勢必浮現。圖 4 顯示企業可以將原本的 IT 應用程式用於雲端運算環境中的一些特性與可能性，例如非核心的商業服務、資料的集中計算、計算資源管理、Web 協作應用、集中化的應用程式、安全化的資料與資訊管理、自動化與隨選服務[25][26][27]。

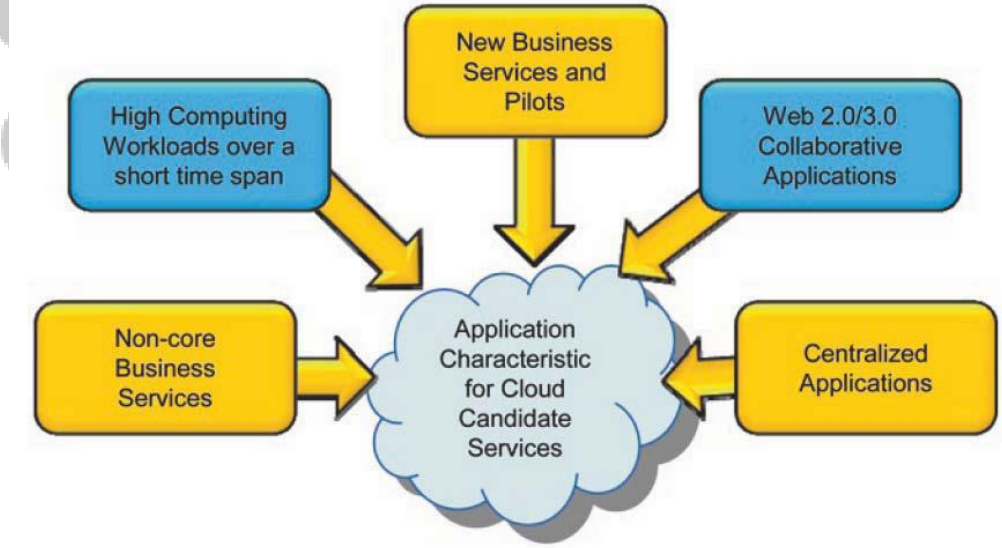

圖 4　企業在雲端運算的應用特性

不同的企業也可能存在不同的雲端服務模式，本論文提供一個雲端監控平台，讓這些企業有的本身就有自己的系統，有些可能已導入

雲端服務，有的還沒，這些可能的服務模式如圖 5 所示，說明如下：

雲端企業服務模式 A：

企業無建置雲端服務中心，透過公有雲方式，連線至企業用戶 B 的雲端系統，進行雲端操作服務。此種服務模式適合中小企業。

雲端企業服務模式 B：

利用雲端服務提供者，透過公有雲方式，提供其他業者連線使用雲端系統。現有此種架構的服務單位如 Google Cloud, Saleforce Cloud, Amazon Cloud。

雲端企業服務模式 C：

利用雲端服務提供者，以提供私有雲的架構，將雲端服務經由企業內部網路，提供可連線至內部網路的終端用戶如企業內員工，企業之間的合作伙伴等連線使用服務。在私有雲的網路服務中，可彈性提供建構資料管理中心於內部網路或外部網路中。

雲端企業服務模式 D：

整合多重式雲端服務架構，其中規劃包含同時支援多種資料中心連線管理，提供於企業內不同服務系統與用戶可快速連結至企業私有雲或是公有雲，增加服的便捷性，而雲端用戶部份亦可同時支援內部網路與外部網路進行管理，提供適合的服務模式。

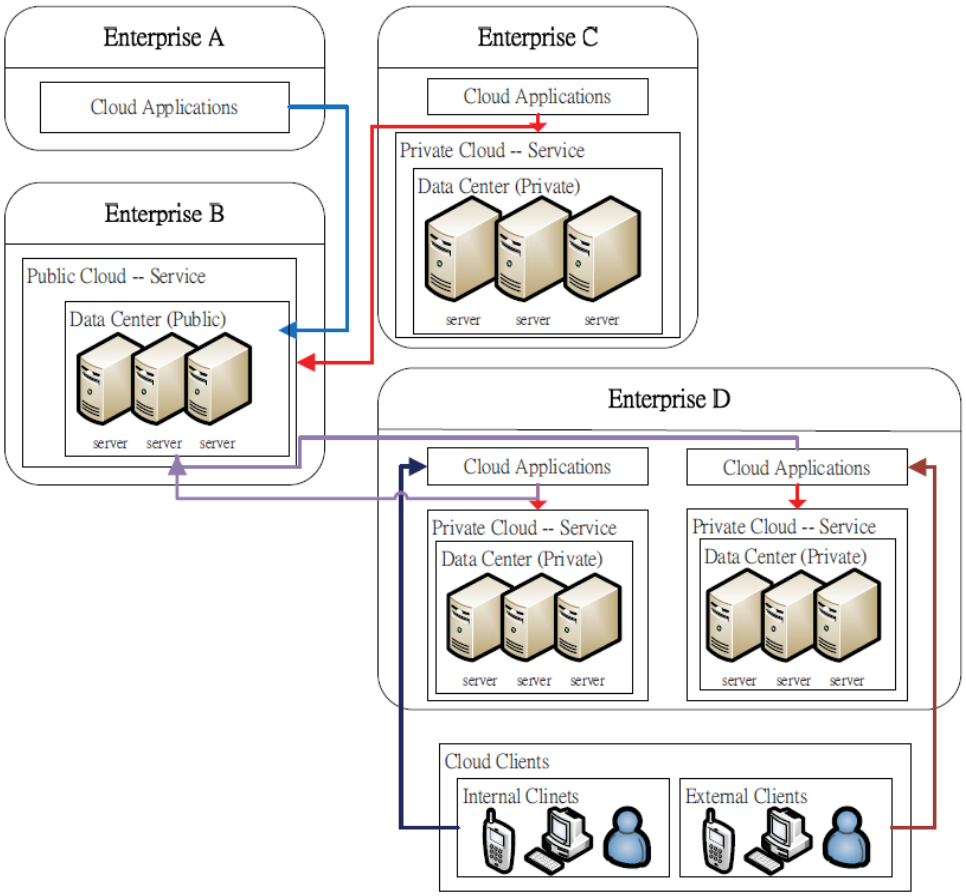

圖 5 企業雲端設計服務模式

2.5. Cloud Computing

雲端的基本概念，是透過網路將龐大的運算處理程式自動分拆成無數個較小的子程式，再由多部伺服器所組成的龐大系統搜尋、運算分析之後將處理結果回傳給使用者。透過這項技術，遠端的服務供應商可以在數秒之內，達成處理數以千萬計甚至億計的資訊，達到和「超級電腦」同樣強大效能的網路服務。它可分析 DNA 結構、基因圖譜定序、解析癌症細胞等高階運算，例如 Skype 以點對點（P2P）方式來共

同組成單一系統；又如 Google 透過 Map Reduce 架構將資料拆成小塊運算後再重組回來，而且 Big Table 技術完全跳脫一般資料庫資料運作方式，以 row 設計儲存又完全的配合 Google 自己的檔案系統(GFS)，以幫助資料快速穿過「雲端」。

雲端運算（Cloud Computing），從本質上來看，它是一種分散式運算(Distributed Computing)[28][29]的新運用，其最基本的概念，是透過網際網路將龐大的運算處理程序(Process)，自動分拆成無數個較小的子程序(Sub process)，再交由多部伺服器(Multi-Server)所組成的龐大系統，透過搜尋與運算分析之後，再將處理結果回傳給使用者端。透過這項技術，網路服務提供者(Service Provider)可以在數秒之內，處理數以千萬計甚至億計的資訊，達到和「超級電腦」同樣強大效能的網路服務。

第三章 研究方法

3.1. 問題定義

在雲端運算技術中，中介軟體位於服務與伺服器叢集之間，如何提供管理和服務即雲端運算結構中的系統，對認證、授權、目錄、安全性等服務進行標準化和操作，為應用提供統一的標準化程式介面和協定,中介軟體層所需處理的項目包含有底層硬體資源資訊記錄配置、作業系統和網路的異質性，利用統一集中記錄與分配管理在雲端服務中的系統效能與資源。因此透過 Middleware 的協助服務，可有效提供 IaaS，PaaS 及 SaaS 之間的協調溝通介面，以及各不同服務之間的資訊交換，並進行服務狀態資訊收集與分析，提供滿足在雲端服務中最需要保證的系統服務品質，達成 QoS 並建立模組化設計，提供企業所需的服務條件限制 SLAs 監控服務。

3.2. 系統架構

我們整合現有的 Open Source 軟體，進行雛形系統的設計，為建構本系統詳細的系統架構圖如圖 6，結合不同功能元件，1.建立 Service Definition Language，定義包含有 Service Requester, Service Provider, Service 提供雲端運算這個較為複雜的環境以及多重契約與限制的定義描述。2.建立 Services Composition，平台中各服務組合元件化，利用不

同元件程序應用開發者可以快速搜尋到開發者所需要的服務構成一個完整的系統。3.確保服務運作狀態，建立 Service Status Collection and Analysis，對應 PaaS 層確認資料存取狀況，並且對 SaaS 確認使用者所需要的服務狀態是否運作。4.Core Component 部份提供 IaaS 基礎架構環境進行資源監控。

3.2.1.具服務品質保證之企業雲端運算中介軟體

我們整合現有的 Open Source 軟體，進行雛形系統的設計，為建構本系統詳細的系統架構圖如圖 6，結合不同功能元件，

1.建立 Service Definition Language，定義包含有 Service Requester, Service Provider, Service 提供雲端運算這個較為複雜的環境以及多重契約與限制的定義描述。

2.建立 Services Composition，平台中各服務組合元件化，利用不同元件程序應用開發者可以快速搜尋到開發者所需要的服務構成一個完整的系統。

3.確保服務運作狀態，建立 Service Status Collection and Analysis，對應 PaaS 層確認資料存取狀況，並且對 SaaS 確認使用者所需要的服務狀態是否運作。

4.Core Component 部份提供 IaaS 基礎架構環境進行資源監控。以下分別針對圖中各個主要項目進行描述：

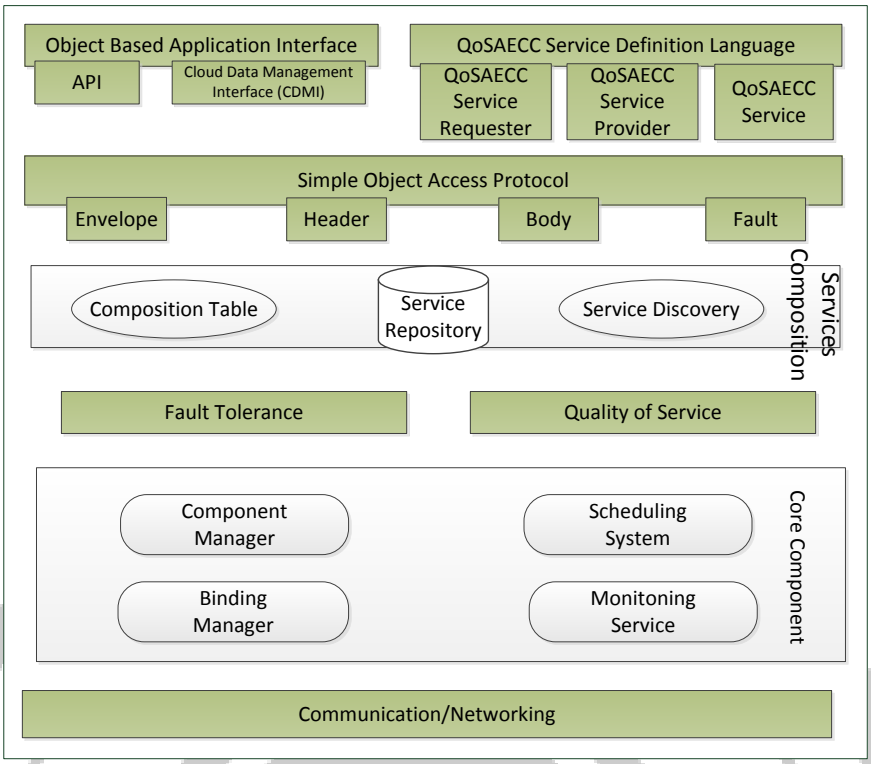

圖 6 具服務品質保證之 ECC 中介軟體架構

3.2.2. QoSAECC Service Definition Language

Web Services 是目前的技術中可以用來實現 SOA 的主要介面標準，Web Service Definition Language 則用來描述 Web 服務的公用介面，是一個基於 XML 來描述如何與 Web 服務通訊和使用的服務描述及進行互動時需要綁定的協議與訊息格式。而本研究中設計之 QoSAECC Service Description Language(QSDL) 是在 Web Service Description Language 的基礎上，擴充對於雲端運算這個較為複雜的環境以及多重契約與限制的定義描述，例如服務提供平台與底層硬體的特徵訊息交換、提供服務的效能需求、服務的可用性、存儲服務的安全需求或行

動裝置支援等，透過 QSDL 的訊息交換完成不同層的溝通、資源的調配與服務及應用程式的佈署。

3.2.3.Services Composition

在 SOA 中是自動透過標準協議和介面把服務做結合，不依賴特定的平台或技術,此外服務的開發,發佈和結合或應用程序的建構在 SOA 都是並行的。在雲端運算中使用 SOA 架構開發應用程序所面臨需要解決的挑戰就是分散式的性質，不僅是服務開發在分散式架構下的不同機器，不同地點，連應用程序開發者，服務代理人，和服務提供商都是各自獨立在不同的地點，因此，透過跨層的 SOA 整合模式來達到跨層資源的整合雲端環境，以滿足使用者需求與 SLAs 的多重限制，提出一套 Service Composition module 提供程序應用開發者可以快速搜尋到開發者所需要的服務，如圖 7 表示。

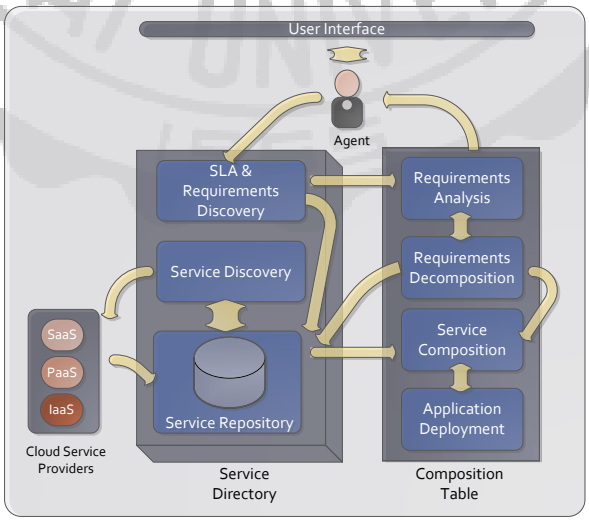

圖 7 Service Composition 示意圖

在圖 7 中，服務供應商可以發佈 SaaS，PaaS，IaaS，和軟體元件，如應用程序模板，使用者介面，Data Schema，Policies，和測試工具等到雲端的一個服務目錄裡。這種雲端聯合服務目錄可以使應用程序開發人員在多個分散式的伺服器上動態的去找尋所需服務並且使用 SOA 和虛擬化技術把這些服務做結合。當使用者要求一個或多個服務時，負責 Service Composition 的 Agent 可迅速地透過服務目錄找到使用者所需的服務。

此架構下具有高度的靈活性，因為 SOA 架構不僅只有軟體服務，還有大量的應用程序元件可以讓人發佈和重用(reuse)，應用程序開發者可以發布 Business Models，Application Templates(workflow structures)，Requirements, Services, Application Interfaces, Testing Tools, Testing Scripts 和 Policies 在一個服務代理的服務目錄下，讓這些服務可以被重用，這種靈活性有助於大規模分散式應用程序的快速發展。

3.2.4.Service Status Collection and Analysis

在圖 8 中，我們提出一個模組 Service Status Collection and Analysis，透過此模組可往下層 IaaS 確認雲端系統環境資源狀態，如:VM 使用狀況，記憶體空間，CPU loading 是否過高等，以確保目前系統資源是否足夠供應服務運作所需要的資源，針對 PaaS 層確認資料存取狀況，並且向上層 SaaS 確認使用者所需要的服務狀態是否運作，是否能夠提供

給使用者做使用，跟Mobile device所需要的互動資訊傳送，如:location，資料傳輸狀況，Mobile device 使用者行為等，最後將結果回傳給 Agent 做自動化分配，以達到服務品質控管。

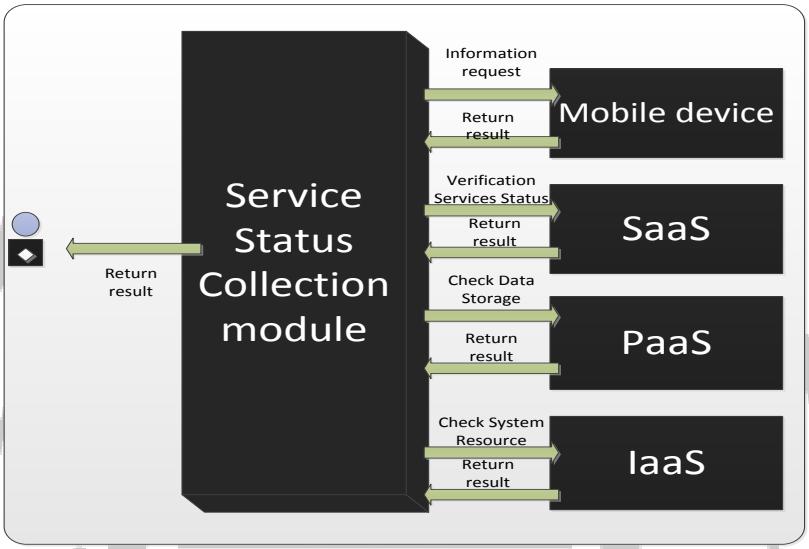

圖 8 服務狀態收集分析

3.2.5.Core Component

部份提供整個中介軟體最主要的功能，讓使用者可以透過中介軟體，呼叫到底層的元件，並存取監控資料。

3.2.6.Component Manager：

根據使用端的要求，尋找適當的元件來完成特定的任務，且根據元件的作用，決定是否下載至使用端執行，並提供元件相關的 API。

- **Binding Manager：**

將使用者端所需要的元件作動態連結，並進行管理。

- **Scheduling System：**

對行程進行排班，有效的處理所有工作。

- **Memory/Resource Monitor：**

監控系統中各種不同資源的狀態(記憶體使用量、CPU 負載、網路流量等)，並且提供統計資料，協助管理系統。

3.2.7.Cloud Service API

API 是雲端運算平台一個很重要的部分，API 是建立雲端運算解決方案的主要機制，它是使用者與雲端溝通的平台，使用者通過各種 API，可以找到自己的設定資料與尋找需求的服務，API 也協調資源與資料庫的存取，讓使用者與平台可以正常運行。 一般來說雲端 API 運作分為四個層次。

1.網路層： 這個層次，開發者可直接撰寫網路格式的要求，若服務以 REST 為基礎，開發者能創造適當的 HTTP 標頭資訊、建立要求內容、並開啟連結服務的 HTTP，REST 服務回覆資料時，會附上 HTTP 回應碼，因為許多 REST 服務直接，故在這個層次撰寫程式碼時相對有效率。

2.特定語言工具箱層： 開發者在這個層次使用特定語言工具箱，以配合 SOAP 或 REST 的要求，雖然開發者仍專注於網路內部的資料格式與結構，許多細節已交由工具箱處理，例如處理回應程式碼、計

算簽章等。

3.特定服務工具箱層： 開發者使用較高層次的工具箱，並與特定服務配合，開發者在此層次能夠專注於企業物件與企業程序，開發者能專注於組織相關的資料與程序，而非網路協定，能更有效的開發應用程式。

4.非特定服務工具箱層： 這是最高層次的 API，開發者使用多重雲端運算供應者的共同介面，和層次 3 相同，開發者專注於企業目標及企業程序，但與層次 3 不同的是，層次 4 開發者不需擔心雲端服務項目，使用非特定服務工具箱撰寫的應用程式，應該不需更動或只更動一小部分，就能轉移至不同的雲端供應者。

API 有許多類型，它可以針對不同的使用者或開發者，API 能說明系統運作支援語法，API 也會針對每一個作業，載明應傳送給系統什麼資訊、系統會回傳什麼資訊，以及任何可能發生的錯誤情況。 我們的雲端平台主要是為了精密工業企業而策劃的，使用雲端雖然可以更容易取得需要的資源，並利於各個公司的交流，但對於企業本身存在的伺服器和資料庫與雲端上的伺服器、資料庫的連結方法就成了我們需要探討的問題。

我們預設的雲端 API 有以下幾種：

一般使用者端：針對一般使用者，我們要開發的 API 是非特定服務工

具箱層，提供使用者請求服務的交互介面，使用者透過 Web 瀏覽器進行註冊、登入及所需服務、並且可以設定和管理使用者，使用者可以指定想要的服務，經由我們的API可以自動化挑選出符合需求的組件，使用者可以依需求刪減組件，達成使用者客制化的目標。使用這個方法，使用者可以不用理解低層雲端的各種操作就能方便且迅速的找到想要的服務，且不同組件可以讓不同使用者重覆利用組合服務，具有 Reuse 元件和樣板的功能。

管理系統和佈署工具：此次我們計劃的 API 其中就有一些是處理內部介面和部署，內部介面即為協調雲端基礎架構不同區塊之間的內部程式介面，也可變更雲端架構儲存服務供應者，部署則是指將程式部署在雲端的程式介面，除了雲端運算特殊封裝技術外，還包括 .Net 組件及 EAR/WAR 檔等傳統封裝機制。

服務目錄：使用者在取得相應使用者權限之後，因為雲端平台提供的並不只是單一的服務，依開發者的努力，一個使用者可使用許多種不同的服務，在這目錄 API 上，使用者可以選擇或定制所需服務清單，也可以對已有服務進行取消的操作，在使用者服務介面即展示圖形或清單方式的服務模式。

監控：有部分 API 是用來監控和計算雲端系統資源的使用情況，因為雲端是一個許多使用者一起共同使用的環境，而資源現實上是有限的，

因此資源(例如儲存空間、資料庫分配、硬體等)的分配就顯得相當的重要。在使用者想使用新的資源時，得自動化查尋現有資源分配，以便做出迅速的反應，完成節點同步設定、負載均衡設定和資源監控，確保資源能順利分配給適合的使用者。

儲存與中介軟體：包括所有今日雲端服務支援的功能(如雲端資料庫、雲端訊息佇列及其他中介軟體)，並包括連結、建立與刪除資料庫與表格的 API。

其中雲端資料庫供應者藉由部分限制，讓產品更具彈性，限制查詢大批資料組時佔用大量處理資源的可能性(例如，部分雲端資料庫應禁止圖表之間連結，有些不支援實際資料庫綱目)。這些限制會導致難易變更雲端資料庫供應者，對於使用關係模型建立的應用程式更是如此。

圖 9 為各 API 之間的互相關係圖。

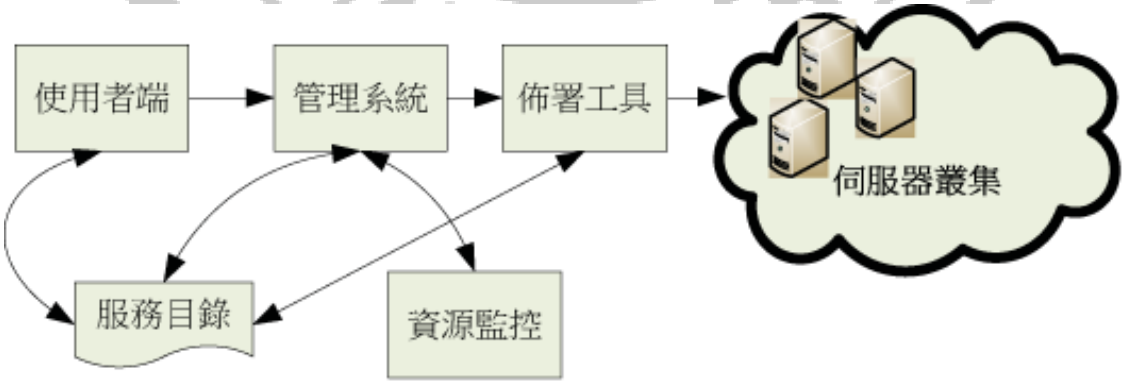

圖 9 API 關係圖

3.2.8.Monitoring Service

當使用者要求使用在雲端運算裡面的某些服務時，我們必須需要讓使用者知道系統能提供多高的資源讓使用者使用此服務，故需要系統的資源監控，隨時掌握目前可使用資源，Middleware 利用 Monitoring 的技術對系統進行監控，根據監控目標的差異設計相對應的代理人(Agent)，適時地將各種狀態回傳給中介軟體進行資料分析，針對不同服務提供對象設計 Agent，主要分為：

1. 底層硬體基礎設施 IaaS 服務專用代理人：收集實體機與虛擬機的資源使用量並動態改變其資源分配以達到效能與節能的平衡，並對系統進行異常行為的偵測及紀錄以協助安全監控。

2. 平台 PaaS 服務專用代理人：提供平台服務間的資料交換與同步，達到資料一致與共用的功能。

3. 軟體應用 SaaS 服務專用代理人：應用服務的控管與整合，協助提供終端裝置如智慧型手機應用程式服務請求交涉等功能。

代理人 Agent 監控架構如下圖 10 所示。

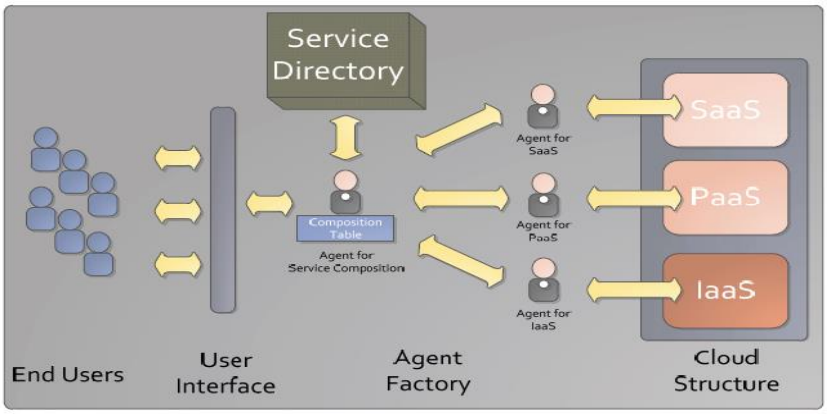

圖 10 Middleware Monitoring with Agents

監控類型可再細分為:

- Server Monitoring:

  為了讓系統管理人員可以更方便管理伺服器，當伺服器發生異常情況，為了維持伺服器之正常運作，系統將自動處理異常狀況，做出相對應的應變措施，並將 log 及實施的應變措施結果回傳給管理者。

- Database Monitoring:

  用來監控資料庫的活動以及分析性能。

- Application Monitoring:

  監控應用程式及進程，它能收集相關資料、運行時間、記憶體使用量和 CPU 時間使用等等，在資源不足或資源過剩時做動態的資源調配處理，並將應用程式類型及資源調配結果寫回資料庫。

- Virtualization Monitoring:

  管理者可以對虛擬主機進行自動化管理，在統一的網管介面中同時對網路、實體主機與虛擬主機進行監控與管理，在資源不足或資源過剩時做動態的資源調配處理，並將記錄當時運算量及其得出的資料

最少搬移量且負載最大平衡的解。

- Service Monitoring:

1.可以指定每次檢查時涵蓋哪些服務是否有執行。

2.當指定的服務狀態並非[執行中] 可以將其[啟動]

3.當任何服務有問題時，會發 email 通知指定的管理者，且通知對象可以多人。 服務的量測通常使用關鍵品質指標(Key Quality Indicators, KQI)和關鍵績效指標(Key Performance Indicators, KPI)，針對明確的面向量測應用程式或是服務的性能，KQIs 可能來自多個來源，包含服務的性能指標或潛在的支持 KPIs，當服務或應用程式是由多種服務所組成，一些不同的 KPIs 可能需要計算某些特定的 KQI 才能確定，KPI 和 KQI 的映對可能很簡單或很複雜，映對可能是正規的，也可能是根據經驗的[30]。圖 11 KPI, KQI, SLA 之間的關係顯示 KPI、KQI 與服務水準協定(SLA)之間的關係，KPIs 可能被定義在 SLAs 中，而 KQIs 從服務層監控的程序中得到，每個 KPI 和 KQI 應包含通知警告與錯誤的門檻值。

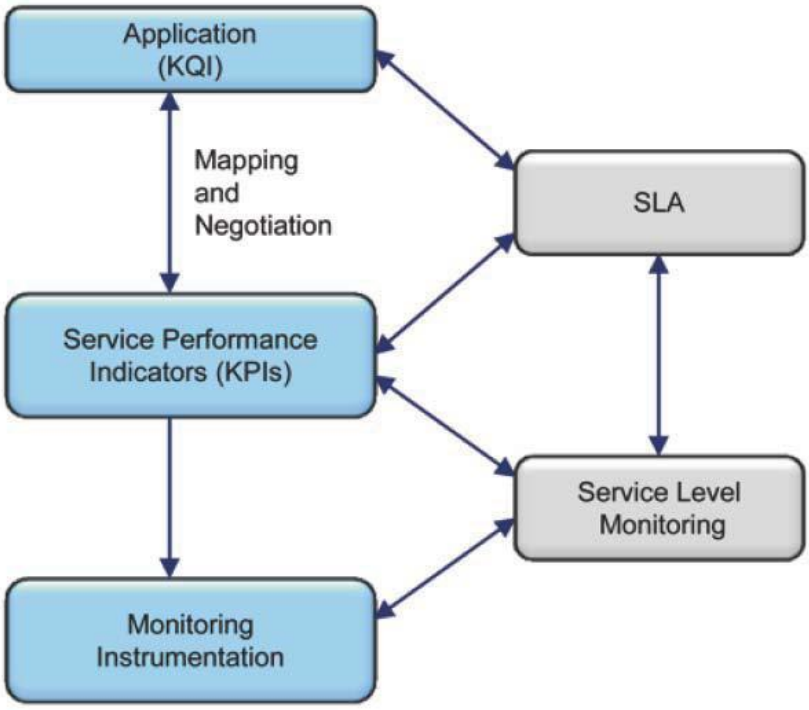

圖 11 KPI, KQI, SLA 之間的關係

- Security Monitoring:

即時監控並針對雲端服務存取行為及資訊內容關連分析，找出可疑的異常使用者，透過使用者信譽評等方式保障雲端服務安全，並在發現異常行為時及時發出資安風險及因應措施等相關警報，虛擬平台惡意行為偵測技術，藉由雲端虛擬平台安全介面，進行系統訊息追跡(Process Tracing)與行為關聯比對(Behavior Inspect)，以偵測 Tenancy 中的惡意為，協助雲端 Infrastructure 系統安全強化。

第四章 系統實作

4.1. 系統環境

由於本論文是實作一個中介軟體必須建立專屬伺服器，與底層硬體設備做串接並提供應用程式介面給其他服做連接，本論文的實作環境如下:

- **Apache 2.2.8**
- **PHP 5.2.6**
- **MySQL 5.0.51b**
- **phpMyAdmin2.10.3**

4.2. 系統成果

4.2.1.服務品質監控規範

本系統可以根據管理者所設定的監控規範，監控系統的狀態限制與服務品質合格率，本監控規範又分為以下兩種規範，根據不同的服務品質監控。

- **系統服務品質監控規範**

系統服務品質監控主要用於監控系統連接資料庫的品質，監控網路傳輸品質與監控系統的回應時間，傳輸品質如檢測封包是否有遺漏需要重新傳送，與傳輸時的系統回應時間是否有符合監控規範所訂定的範圍內，如果沒有符合該監控規範，會於圖 13 服務品質監控頁面以圖表的方式呈現，方便管理者監控系統目前服務品質狀態。

- **功能服務品質監控規範**

功能服務品質監控主要用於監控使用者操作的品質，監控網路傳輸品質與監控系統的回應時間，傳輸品質如檢測封包是否有遺漏需要重新傳送，與傳輸時的系統回應時間是否有符合監控規範所訂定的範圍內，如果沒有符合該監控規範，會於圖 13 服務品質監控頁面以圖表的方式呈現，方便管理者監控系統目前服務品質狀態。

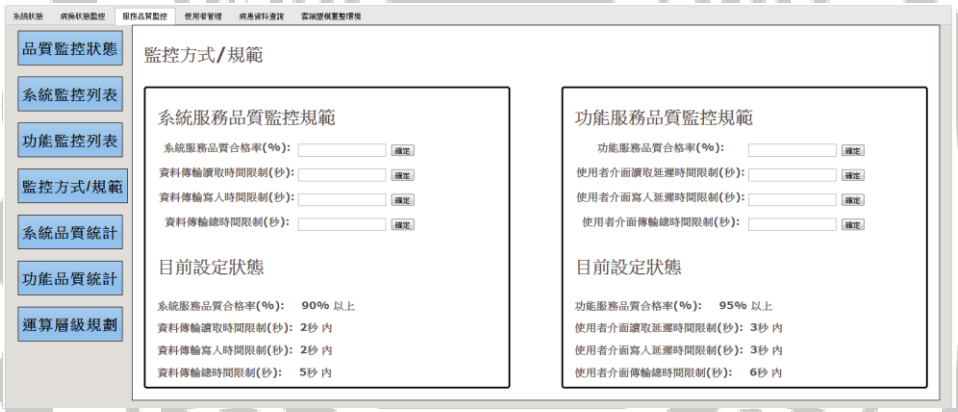

圖 12 服務品質監控規範

4.2.2.服務品質監控

- **系統服務品質監控**

根據監控方式規範之規範監控系統服務品質，顯示系統測試監控的總次數與是否符合標準的次數，根據總次數算出服務品質的合格率是否符合標準，以及資料庫監控總次數與未符合次數。

- **功能服務品質監控**

根據監控方式規範之規範監控功能服務品質，顯示功能操作監控的總次數與是否符合標準的次數，根據總次數算出服務品質的合格率是否符合標準，以及使用者操作總次數與未符合次數。

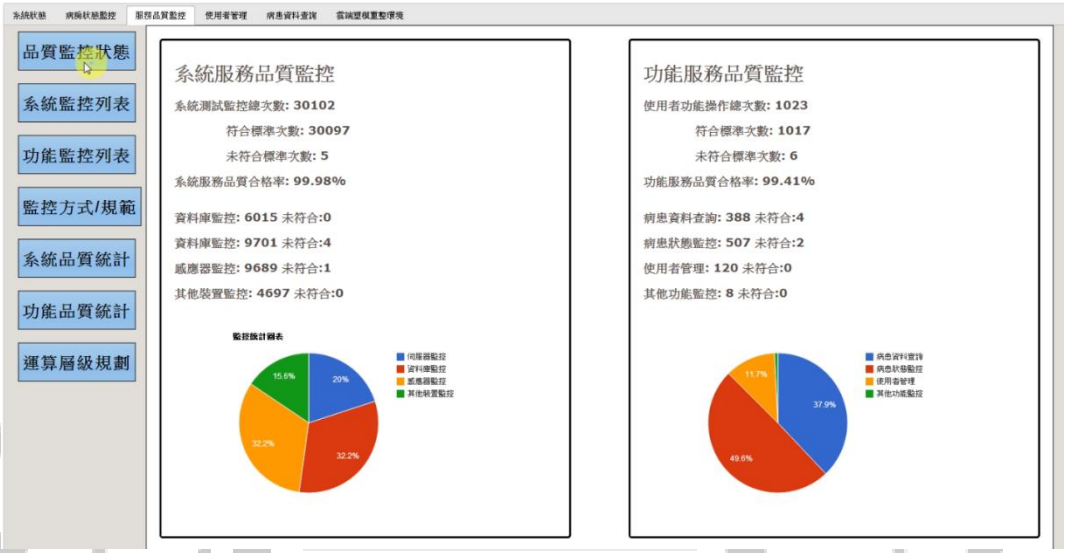

圖 13 服務品質監控

4.2.3. 系統監控列表

系統監控列表是根據圖 13 列出全部測試列表結果，圖 14 標記紅色部分為不符合服務品質的事件，下方會列出該事件詳細清單，如:發生時間、監控狀態、運行時間與錯誤資訊。

圖 14 服務品質監控列表

4.2.4.服務品質統計

服務品質統計會列出每個月服務品質統計，呈現方式以表格與長條圖的方式顯示，根據不同的顏色區分不同年度的資訊，方便管理者管理與維護系統品質。

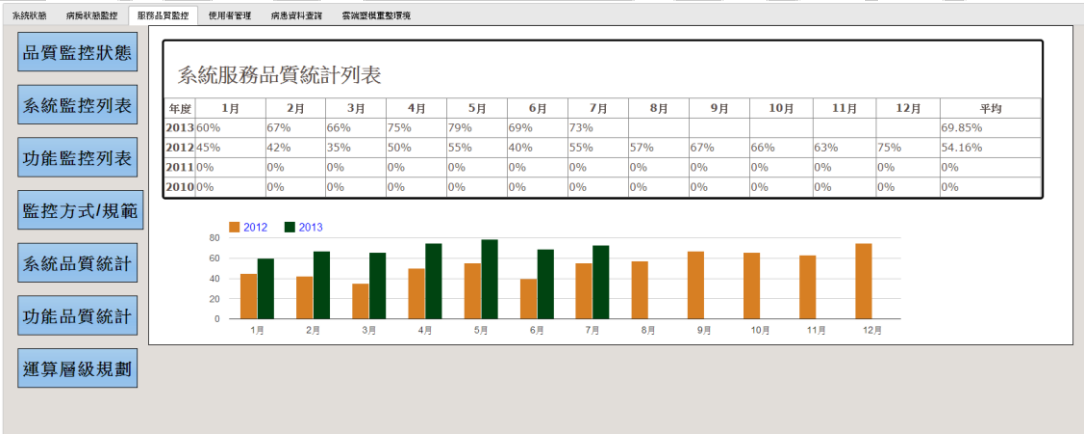

圖 15 服務品質統計列表

4.2.5.運算層級規劃

運算層級規劃可以列出每個時段運算的尖峰時間，以及資料庫使

用者操作寫入回應的平均時間，根據圖表的結果讓管理者參考每個時段的運算量。

圖 16 服務品質運算層級規劃

TUNGHAI UNIVERSITY 1955

第五章 結論與未來工作

企業雲端運算(Enterprise Cloud Computing)是 IT 服務產業的新趨勢，因為可以降低成本並提供服務的品質(QoS)。然而，把傳統企業運算服務轉移至雲端環境有很多的阻礙，比如如何確保服務的品質，包含安全性、可用性、資料隱私等等。從多重租約、跨層服務組合(SaaS、PaaS、IaaS)、使用者需求和服務層級協議(SLAs)的多重限制等問題讓等讓企業雲端運算(ECC)議題變得十分複雜。

為了克服企業雲端運算的困難，本論文提出一套技術去確保服務的品質和 IaaS、PaaS、SaaS 方面的運算成效。由於企業雲端運算的高複雜性，本論文中建構包含 SaaS、PaaS、IaaS 與服務的正規化塑模方法，以確保所提供的服務品質協議可以滿足多重品質保證因素，例如安全性、效能、時效性、傳輸速度和可靠性，可同時對大量客戶需要的運算各種需求做系統化分析。未來希望可以加入自動化分析模組與測試，可以達成自動化測試以及監控執行階段系統是否符合需求與品質服務協議等相關限制要求，並將平台移植到其他企業進行實際運作測試。

參考文獻

[1] A.B. Nassif and M.A.M. Capretz, “Moving from SaaS Applications towards SOA Services,” in Proc. SERVICES, 2010, pp.187-188.

[2] Junjie Jing, Jian Zhang, “Research on Open SaaS Software Architecture Based on SOA” appears in: Computational Intelligence and Design (ISCID), 2010, pp.44 – 147.

[3] Yee Ming Chen, Yi Jen Peng, “A QoS aware services mashup model for cloud computing applications,” JIEM, 2012-5(2): 457-472.

[4] Chao-Tung Yang, Shih-Chi Yu, Chung-Che Lai, Jung-Chun Liu and William C. Chu, “Implementation of a SOA-Based Service Deployment Platform with Portal,” International Conference, FGCN 2010.

[5] Krogdahl, P., Luef, G., Steindl, C., "Service-oriented agility: an initial analysis for the use of agile methods for SOA development", IEEE International Conference on Services Computing, 2005.

[6] Schelp, J., Aier, S., "SOA and EA - Sustainable Contributions for Increasing Corporate Agility", The 42nd Hawaii International Conference onSystem Sciences, 2009. HICSS '09.

[7] Chao-Tung Yang, Tsui-Ting Chen, Keng-Yi Chou, C.W. Chu.” Design and Implementation of an Information Service for Cross-grid Computing Environments,” The 12th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS 2008), Kunming, China, pp.99-105, Oct. 21-23, 2008.

[8] Roy H. Campbell · Mirko Montanari · Reza Farivar, “A middleware for assured clouds”, Journal of Internet Services and Applications,

SpringerLink 2011.

[9] Hoh P. In, ChangHwa Kim, Unil Yun and Stephen S. Yau “Q-MAR: A QoS Resource Conflict Identification Model for Situation-Aware Middleware” Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS’03) 2003.

[10] Wohlstadter, E., Tai, S., Mikalsen, T., Diament, J., Rouvellou, I., "A Service-oriented Middleware for Runtime Web Services Interoperability", International Conference on Web Services, 2006. ICWS '06.

[11] Tambe, S., Dabholkar, A., Gokhale, A., "Fault-Tolerance for Component-Based Systems - An Automated Middleware Specialization Approach", IEEE International Symposium onObject/Component/Service-Oriented Real-Time Distributed Computing, 2009. ISORC '09.

[12] Coulson, G., Grace, P., Blair, G., Mathy, L., Duce, D., Cooper, C., Yeung, W.K., Cai, W., "Towards a component-based middleware framework for configurable and reconfigurable grid computing", The 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004.

[13] Ajith Ranabahu and Michael Maximilien, "A Best Practice Model for Cloud Middleware Systems", The International Conference on Object Oriented Programming, Systems, Languages and Applications, 2009. OOPSLA 2009.

[14] Zhaoyu Liu, Dichao Peng, "A Security-Supportive Middleware Architecture for Pervasive Computing", The 2nd IEEE International

Symposium on Dependable, Autonomic and Secure Computing.

[15]Haresh Luthria, Fethi Rabhi, and Michael Briers “Investigating the Potential of Service Oriented Architectures to Realize Dynamic Capabilities” IEEE Asia-Pacific Services Computing Conference. 2007.

[16]Jiehan Zhou, Daniel Pakkala1, Juho Perälä, Eila Niemelä, and Jukka Riekki, Mika Ylianttila, “Dependency-aware Service Oriented Architecture and Service Composition” IEEE International Conference on Web Services (ICWS 2007).

[17]Rob High, Jr.，Stephen Kinder and Steve Graha, “IBM’s SOA Foundation：An Architectural Introduction and Overview” SOA Foundation Architecture Whitepaper v1.01 2005.

[18]Cloud Computing Use Case Discussion Group, "Cloud Computing Use Cases White Paper".

[19]John Buford, Alan Brown, Mario Kolberg “Meta Service Discovery” Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW’06).

[20]Chen Yanping, Li Zengzhi, Wang Li, Yang Huaizhou, "Service-Cloud Model of Composed Web Services", Third International Conference on Information Technology and Applications, 2005. ICITA 2005.

[21]Jianqiang Hu, Changguo Guo, Peng Zou, "WSCF: a framework for Web service-based application supporting environment", IEEE International Conference on Web Services, 2005. ICWS 2005.

Proceedings. 2005.

[22] James Staten, Simon Yates, Walid Saleh , “How Large Enterprises Approach IT Infrastructure Consolidation”, 2007.

[23] Xiaoying Bai, Wenli Dong and Wei-Tek Tsai, Yinong Chen, “WSDL-Based Automatic Test Case Generation for Web Services Testing”, Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE’05).

[24] Brock, M., Goscinski, A., "Offering clusters from clouds using WSDL and stateful web services", IEEE Asia-Pacific Services Computing Conference, 2009. APSCC 2009.

[25] Chang, William Y, Abu-Amara, Hosame, Sanford, Jessica Feng, "Transforming Enterprise Cloud Services" 1st Edition, 2010, Springer.

[26] Lakshmanan, G., Pande, M.: "How the Cloud stretches the SOA scope", MSDN architecture center. Arch J, http://msdn.microsoft.com/en-us/architecture/aa699420.aspx

[27] Longji Tang; Jing Dong; Yajing Zhao; Liang-Jie Zhang; "Enterprise Cloud Service Architecture", IEEE 3rd International Conference on Cloud Computing (CLOUD 2010), 2010.

[28] Andrews, G.R. Foundations of Multithreaded, Parallel, and Distributed Programming, Addison–Wesley, ISBN 0-201-35752-6, (2000).

[29] Sanjeev, A.; Boaz, B. Computational Complexity – A Modern Approach, Cambridge, ISBN 978-0-521-42426-4, (2009).

[30] “The Open Group: SLA Management Handbook”, vol. 4. The Open Group, Berkshire, 2004.