以網路服務為基礎兼具運算協調能力的分享設計

王振生
國立交通大學資訊管理研究所

蔡銘箴
國立交通大學資訊管理研究所

摘要

Computing Power Services (CPS)是一個輕量級以網路服務(Web Services)為基礎的運算能力分享設計，適合在企業內的信賴網路中執行程式的運算，並使用 Web Services 和 Business Process Execution Language(BPEL)提供可圖形化發展環境和流程控制管理的能力。然而，在類似 CPS 這種即時分散式運算環境中，為了達到最佳化的運算效能，協調運算資源是其中重要的關鍵。因此，本研究利用模糊群體決策方法，使 CPS 架構具有服務品質(Quality of Service)的即時運算協調能力，來提升整體架構的執行效能，並使得在資源、條件不同的情形下，可保有相對穩定的運算能力。在本研究中，此方法運用在分析數位浮水印強度的 filter bank 選擇，用以測試執行效能，和未使用協調運算機制相比，在效能及穩定度上，本研究設計皆能加以改善。

關鍵字：Web Services、BPEL、模糊群體決策、分散式運算、服務品質

Web-Services Based Coordination Design for Computing Power Services

Chen-Sheng Wang
Institute of Information Management National Chiao Tung University
Min-Jen Tsai
Institute of Information Management National Chiao Tung University

Abstract

Computing Power Services (CPS) is a lightweight, Web-services-based computation sharing architecture. It can be implemented within the trusty enterprise network which utilizes Web-services and Business Process Execution Language (BPEL) in order to provide a visualized developing environment with workflow management capability. However, the coordination mechanism of the computing resources is a key factor to have the optimized results in a real-time distributed computing architecture like CPS. Therefore, a fuzzy group decision-making module is proposed in this paper for CPS to provide real-time computation coordination and quality of service. This module increases the computing performance efficiently and shows stability in various environments. In this research, the module has been applied to analyze the robustness of digital watermark by filter bank selection. As the result of this study, the performance can be improved in the aspect of speedup and stability.

Keywords: Web Services、Business Process Execution Language (BPEL)、Fuzzy Group Decision-making、Distributed Computation、Quality of Service

壹、前言

隨著資訊系統的運算量以及資料處理量的日益增加，傳統的主從架構(client/server)已經逐漸被網格運算(Grid Computing)或同儕運算(Peer-to-Peer;P2P)等能夠透過網路分享內容或資源的運作模式所取代；儘管這兩種分散式運算模式的應用已日益普及，但是對企業組織在使用這些分散式運算模式時，仍然要面臨相關的安全性、動機性、彈性、相容性和流程管理的議題，因此，(Tsai et al. 2006)提出 Computing Power Services (CPS)架構，CPS 是以服務導向架構(Service-Oriented Architecture; SOA)為基礎，利用 Web Services 鬆散耦合和開放式標準的特性和 Business Process Execution Language (BPEL)的流程管理能力來解決 P2P 在彈性、相容性以及流程管理上的問題。CPS 架構中，包含運算端、運算需求者、協調者，這三個角色分別被定義成扮演服務消費者、服務提供者和服務註冊者，其詳細運作方式將在第二節做介紹。因為 CPS 的目的在提供一個分散式環境，讓各個自願提供硬碟空間資源、運算能力的系統間能獨立運行所指派的運算工作，然後再由 CPS 整合結果。所以，協調運算資源的能力對整體運算效能將是其中重要的關鍵。

在 CPS 的設計中，系統可藉由分配可用記憶體和 CPU 執行時間來執行工作，因此，在不同系統間資源分配的問題，可被視為選擇具有最佳資源系統來協調執行運算工作的決策問題。決策問題在企業和學術中一直是個很重要的研究領域，因此有關決策問題有許多解決方法，例如：層級分析法(Analytic Hierarchy Process, AHP) (Saaty 1980)，它是一種能將定性分析和定量分析相結合，將人的主觀判斷用數量形式表達和處理的系統分析方法。此外(Kacprzyk 1986 ; Chiclana et al. 1998, 2001)提出利用模糊理論的群體決策程序(Resolution Process of GDM;RPGDM)，因為 RPGDM 使用模糊集合來處理不確定資訊和幫助轉換成確定資訊，此種不確定資訊的情況，可應用於 CPS 架構中運算端允許使用者中止或啟動運算的行為。所以本研究選擇 RPGDM 方法來協調運算資源的使用。

在本文以下的內容，第二節文獻探討將針對 CPS 架構及 RPGDM 方法做介紹，第三節實驗執行方法則是解釋如何利用 RPGDM 的方法，在 CPS 架構上執行協調運算資源的問題，並以數位浮水印強度的 filter bank 選擇，來實驗本研究的設計，第四節則探討實驗結果和未使用 RPGDM 方法時的效能及穩定度比較並分析，第五節則是探討協調機制對需求端、運算端，以及權重對運算效能的影響，第六節則是本研究的結論。

貳、文獻探討

由於本研究的主題為分散式運算，且將植基於模糊理論的群體決策程序運用在 CPS 架構中，以協調運算資源達到提升工作運行效能與穩定性，因此本節將對分散式運算、CPS 架構以及模糊群體決策這三者做探討與介紹。

一、分散式運算

目前主要的分散式運算架構有 P2P 和 Grid Computing，P2P Computing 是透過聚集大量的電腦來達到運算處理延伸性，利用中央伺服器將運算問題切成多個運算工作，運算工作由每一個電腦個別完成，工作處理完成後的結果再由中央伺服器收集。Grid Computing 則是另一種概念，它將整個電腦網格中的大量電腦當作是一個單一虛擬電腦來使用(Carlos et al. 2004)。以下可由四個角度區別 P2P Computing 和 Grid Computing 的差異：

1.資源管理方式不同

Grid Computing 的資源管理是一個穩定且層次結構明顯的資源體系，採用一些專屬的軟體與硬體資源，想在網格定義的資源中進行檢索，必須要符合限制的條件。而 P2P Computing 的資源是動態組成，如軟體、書籍、音樂，影片，因此在 P2P Computing 裡檢索相較於 Grid computing 更具有彈性。

2.參與者不同

Grid Computing 的出現是為了處理複雜程度高、數據量龐大的科學運算，主要的精神是將分散在不同地理位置的電腦組織成一個「虛擬的超級電腦」，以實現大規模的、安全的以及協同的資源共享，因此參與 Grid Computing 的節點一般都是高性能電腦節點。

而 P2P Computing 主要參與者大多是普通的個人電腦，電腦性能差異度大。透過大量節點之間的直接連結來共享包括儲存空間、頻寬、運算能力等各種電腦資源。Grid computing 一般則是使用高速網路連接大型主機系統，因此具有嚴格的帳號審查，參與的節點數目少，相反地，一般參與 P2P Computing 的是普通個人電腦，所以節點數目也比 Grid computing 多。

3.可靠度不同

Grid computing 資源大多都集中在部分節點，一般採用集中、分層的結構，因此穩定性較高。而在 P2P Computing 中節點的加入和離開比較無限制，端賴參與者的意願。

4.流程控管能力

Grid Computing 有許多流程控管系統，例如 GridAnt、GSFL、Pegasus、Karajan、GFMS(Zhang et al. 2004；Pichet et al. 2005)。大部分的 Grid Computing 系統在協調他們的服務時，都針對自己系統的需要來設計，因此有個別專用的流程語言，造成不允許透過異質流程引擎執行相同流程步驟的情況。例如 GridAnt 和 Karajan 都採用 Apache Ant 引擎建立，但兩者無法達到互通操作性(Interoperability)。因此，(Pichet et al. 2005)發展一個使用共通標準進行 Grid 的流程控管機制，希望藉由共通標準如 BPEL 的導入達到在不同 Grid Computing 系統中流程的相互應用。至於 P2P Computing 由於節點是隨意組成，因此，目前為止並無流程控管的能力。

二、CPS 架構

CPS 是一個以網路服務為基礎的運算能力分享架構。運算能力分享架構最有名的例子是西元 2001 年由 Intel、牛津大學、英國國家癌症研究基金會和 United Devices, Inc. 共同啟始的篩選抗癌分子的計畫，此計畫至 2005 年已成功吸引超過 300 萬台個人電腦來參與，然而此計畫有安全性、動機、執行效率、相容性等問題 (Loo 2003)。為此，(Tsai et al. 2006)將 CPS 架構運用在企業內可信賴的網路，並利用網路服務鬆散耦合和開放式標準的特性及 BPEL 的流程管理能力，使得 CPS 能對前述問題提供解決方案。

（一）角色定義

由於 Web Services 是以訊息為導向的架構，因此在 CPS 架構中，運算需求者如果不是主動送出要求訊息給提供者，就是提供者輪詢需求者是否需要服務。如果是前者，將變成每個使用者都需要安裝一個網路服務在電腦上，換言之，每一個想提供運算能力的電腦，都需要安裝一個應用程式伺服器，作為網路服務運行的環境，這樣環境對於參與者來說過於複雜，參與的意願亦可能因而減少。因此，本研究採用後者機制，以 Thin Client 的概念讓參與運算端只需下載一個簡單程式來常駐運行。如果以 SOA 來描述的話，提供 CPU 閒置時間的運算端為服務的需求者，運算需求者為服務的提供者，另外為協調多個運算需求者的需求，CPS 架構也定義一個協調者來註冊不同的需求，以便協調運算端去取得工作來運算。根據(Tsai et al. 2006)，三個對應 SOA 角色所負責的工作如下：

1.**協調者**(Coordinator)

協調者扮演仲介和公正第三方的角色，主要的功能是負責維護一份需求者所在位址與需求的對應清單，該份清單是需求者來登記需求而產生。運算者則必須透過協調者，詢問是否有適合的工作。協調者則會根據清單，循序的(round robin)根據需求把需求者的位址通知給運算者。之後，運算者自行和需求者溝通運算工作，協調者只負責撮合。另外，協調者也提供一些如 log、帳號管理之類的管理功能。在整個 CPS 架構中類似 SOA 的 UDDI 角色。不過，UDDI 會因不同的需求、領域或商業考量建立多個 UDDI，當多個 UDDI 使用時，便需要考量到一致性、協調性以及安全性的問題。目前，CPS 是架構在一個組織內部的信賴網路，所以採用一個協調者便可以滿足協調的需求。

2.**運算需求者**(Computing Power Requester)

需求者先根據自己的運算需求以 BPEL 設計整個運算流程，然後需求者跟協調者登記，公告它的需求。爾後，它會指派工作給與它接洽的運算者，來協助自己完成實驗並監控實驗流程的進行。整個工作都在 BPEL 引擎運作，並定義一個非同步網路服務接受工作要求。

3.**運算者**(Computing Unit)

運算者在整個架構中是進行運算的角色。透過運算端程式，運算端會在 CPU 閒置時去詢問協調者是否有適合的工作需求。待協調者回覆需求者的位址後，運算端會透過

需求者端所開放的網路服務跟需求者接洽，之後會從需求者處得到所指派工作。並下載工作所需檔案之後開始運行，完成後會回報給需求者。這個程序會一直循環地進行。

（二）功能描述

(Tsai et al. 2006)為使整個 CPS 架構發展模組化，依功能把它分為五層。分別為運算能力分享層(Power Sharing Layer)、通訊層(Communication Layer)、契約簽訂層(Contract Layer)、發現層(Discovery Layer)及使用者層(User Layer)。其架構可參考圖 1.。

1.運算能力分享層

對應到網路服務架構中的描述層(Description Layer)。主要在描述需求者和運算者之間互動的關係及方式。

2.通訊層

沿襲網路服務的通訊方式，採用 SOAP 協定。

3.契約簽訂層

定義運算端和需求者溝通的一層。CPS 架構將兩者的溝通稱為契約簽訂，並以 XML 定義契約文件。

4.發現層

發現層主要由協調者執行，仲介運算端尋找需求者，猶如 SOA 中 UDDI 的角色。

5.使用者層

使用者層是使用者存取整個系統及架構的介面。分為需求者和運算端兩個部份。運算者端主要就是一些供使用者操控的介面；需求者端則是以 BPEL 實驗流程為主體，可以透過視覺化的介面以及 BPEL 引擎來設計管理實驗流程，提供了一個視覺化且兼具工作流程控管能力的環境。

由於 CPS 架構的目的，在於希望充分利用組織內閒置的電腦資源，並在信賴網路內進行運算工作，因此不計較高性能的運算節點，也不強迫或佔用參與電腦的正常使用，而且允許節點可隨時加入或離開。這些特性正是 P2P Computing 的特色，也是 CPS 架構以 P2P 概念為基礎進行設計的原因，但為解決 P2P 在彈性以及流程管理上的問題，CPS 利用 BPEL 的流程管理能力，並採用基於 BPEL 標準開發的 BPEL 圖形化設計引擎的功能，方便運算需求者設計實驗流程，將需要用到大量運算的實驗很容易地將工作分配出去，達到有效運用運算能力的目的。

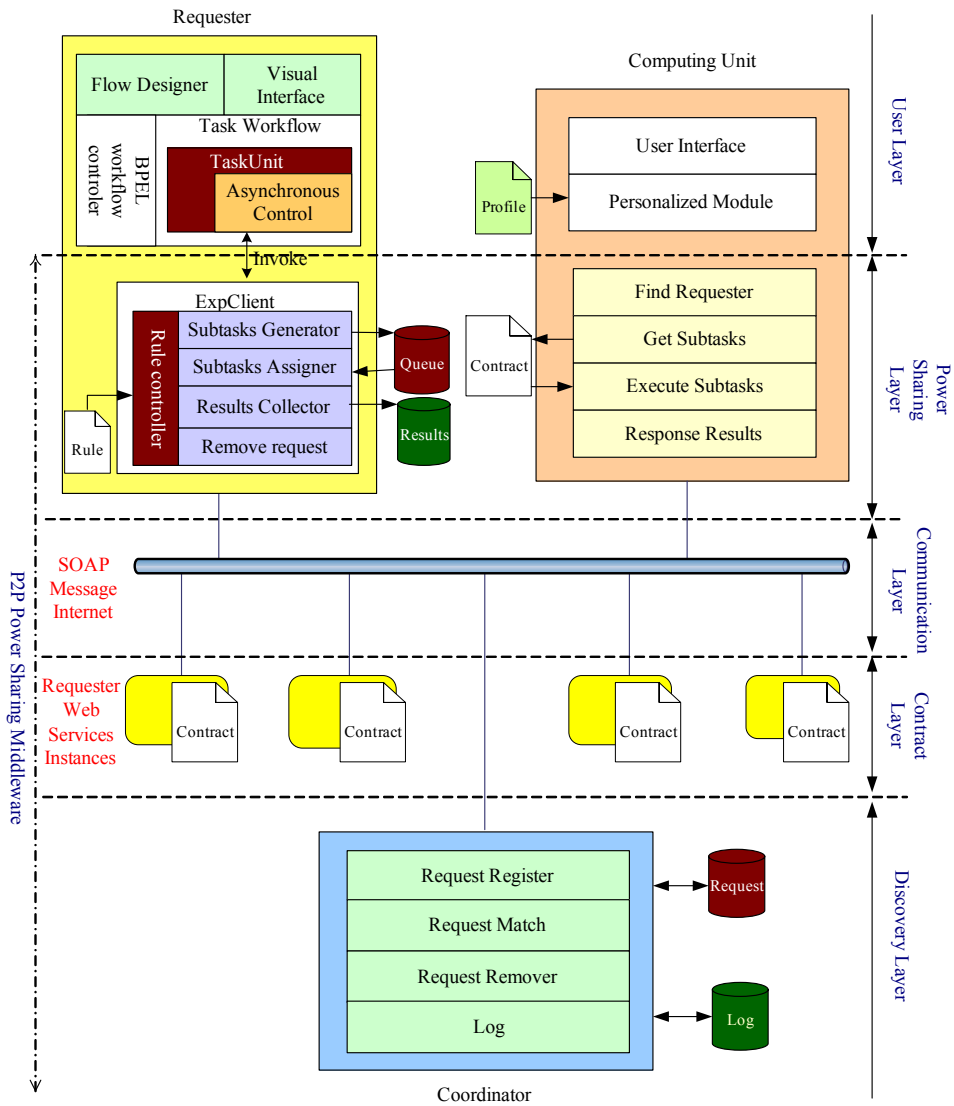

圖 1：CPS 架構圖

三、模糊群體決策

在類似 CPS 的一個分散式環境中，如何協調運算資源的能力對整體運算效能將是重要的關鍵。眾所皆知系統藉由分配可用記憶體和 CPU 執行時間來執行工作，因此，在不同系統間資源分配的問題，可被視為選擇當時具有最佳資源系統來執行運算工作的決策問題。

決策問題在企業和學術中一直是個很重要的研究領域，因此有關決策問題有許多解決方法，例如：層級分析法(Analytic Hierarchy Process；AHP) (Saaty 1980)，它是一種能將定性分析和定量分析相結合，將人的主觀判斷用數量形式表達和處理的系統分析方法。此外(Kacprzyk 1986 ; Chiclana et al. 1998, 2001)提出利用模糊理論的群體決策程序(Resolution Process of GDM;RPGDM)，來處理不確定資訊和幫助轉換成確定資訊，並使用模糊多數(Fuzzy Majority)來決定最佳方案。此種處理不確定資訊的情況，很適合用來描述 CPS 架構中，運算端允許使用者中止或啟動運算的行為，以及電腦運算時隨時變動以致無法準確估算的可用資源，所以本研究選擇 RPGDM 方法來解決“選擇當時具有最佳資源運算端來執行運算工作的決策問題”。

在(Chiclana et al. 1998)中將 RPGDM 分為轉換階段（The transformation phase）、彙整階段（The aggregation phase）及探索階段（The exploitation phase）三階段。

不過在介紹 RPGDM 的過程之前，先定義決策問題。假設存在一個方案有限的集合 $A=\{a_1,a_2,...,a_n\},n\geq 2$，然後有限的專家集合 $E=\{e_1,e_2,...,e_m\},m\geq 2$，在方案集合 A 中評估對每個方案的偏好等級，最後根據這些偏好等級運用決策方法選出最佳方案。底下就針對三階段的執行做深入描述。

（一）轉換階段（The transformation phase）

由於專家間對於問題可能解決方案的喜好意見，會因專家間背景各異，而有不同的喜好程度表示法，所以此階段用來將不同的喜好程度表示法轉換成均一的表示法，以便計算專家意見間的共識性。

根據(Chiclana et al. 1998, 2001) 研究，專家在評估喜好程度時，目前共有四種表示方式，分別為：偏好排序（Preference ordering among alternatives）、模糊偏好關係（Fuzzy preference relation）、多準則偏好關係（Multiplicative preference relation）、效用函數（Utility function）。

因為 RPGDM 採用模糊偏好關係(Fuzzy preference relation)來進行喜好程度的彙整，如果專家以其他三種方法表示喜好程度，就必須轉換成模糊偏好關係表示方法。模糊偏好關係可被定義為：$P_{ij}{}^k, 1\leq i,j\leq n, 1\leq k\leq m$，它描述對 e_k 專家而言，方案 a_i 優先於方案 a_j 的程度。(Chiclana, 1998; Chiclana, 2001) 研究建議如下轉換公式：

1.轉換偏好排序為模糊偏好關係的公式：

$$p_{ij}^k=\frac{1}{2}(1+\frac{o_j^k-o_i^k}{n-1}) \tag{1}$$

其中 o_j^k 表示為 e_k 專家在方案集合中給予方案 a_j 的偏好排序。

2.轉換多準則偏好關係為模糊偏好關係公式：

$$p_{ij}^k=\frac{1}{2}(1+\log_9 a_{ij}^k) \tag{2}$$

其中 a_{ij}^k 表示為 e_k 專家在方案集合中，以比率刻度為 1~9 給予方案 a_i 對方案 a_j 的偏好比率值。

3.**轉換效用函數為模糊偏好關係公式：**

$$p_{ij}^{k}=\frac{(u_i^k)^2}{(u_i^k)^2+(u_j^k)^2} \tag{3}$$

其中 u_i^k 表示為 e_k 專家在方案集合中給予方案 a_i 的效用函數值。

（二）彙整階段（The aggregation phase）

在轉換所有專家的偏好等級為模糊偏好關係之後，此階段的目的在取得一個集體偏好關係 P_{ij}^{c} , $1\le i, j \le n$。RPGDM 使用模糊語意量詞(fuzzy quantifiers)表示模糊多數(Fuzzy majority)觀念，並且用排序加權平均(Ordered Weighted Average; OWA)運算子彙整偏好資訊以得到 P_{ij}^{c}。

多數決概念在傳統上定義為通過決策所需專家人數的門檻值，然而在模糊多數中，它卻是個軟性多數決概念，是運用以語言量化觀點(linguistically quantified propositions)的模糊邏輯基礎來運算(Zadeh 1983 ; Kacprzyk 1986)。

排序加權平均運算子是由 (Yager 1988)提出的一個彙整運算子，利用此運算子所產生的彙整值將介於最小和最大運算子所產生的運算值之間。在使用排序加權平均運算子之前，我們需先定義一個加權向量 $W=\{w_1, w_2, ..., w_m\}, w_i \in [0,1], \sum_{i=1}^{m} w_i = 1$ ，以便能將每個專家對方案集合所產生的模糊偏好關係集合$\{P_{ij}^{1}, P_{ij}^{2}, \ldots, P_{ij}^{m}\}$彙整成 P_{ij}^{c}。一個 m 維的排序加權平均運算子是個包含模糊語意量詞 Q 的公式(4) $F_Q(P_{ij}^{1}, P_{ij}^{2}, ..., P_{ij}^{m}) = \sum_{k=1}^{m} w_k \bullet p_k$，其中 P_k 表示在偏好關係集合$\{P_{ij}^{1}, P_{ij}^{2}, \ldots, P_{ij}^{m}\}$中為第 k 大的值。

如何定義加權向量是使用排序加權平均運算子時的關鍵，在(Yager 1993)中，建議使用一個包含非遞減比例式語意量詞(Non-decreasing proportional quantifier) Q 的運算子如下：

$w_i = Q(\frac{i}{n}) - Q(\frac{i-1}{n}), i = 1, ..., n$ ，其中，$Q(r) = \begin{cases} 0 & r < a \\ \frac{r-a}{b-a} & a \le r \le b \\ 1 & r > b \end{cases}$ $a, b, r \in [0,1]$.

例如，如果使用「多數」模糊語意量詞此時定義 [a,b]=[0.3,0.8]，則一個維度 4 的加權向量將是 $\left[0, \frac{2}{5}, \frac{1}{2}, \frac{1}{10}\right]$.

（三）探索階段（The exploitation phase）

在這個階段中，RPGDM 使用兩個模糊評比(Fuzzy ranking)方法，Quantifier Guided Non-Dominance Degree (QGNDD) 和 Quantifier Guided Dominance Degree (QGDD)來確認方案的優先順序，QGNDD 表示一個方案不被其他方案的模糊多數所壓倒的程度(Dominance degree)。而 QGDD 則是表示一個方案被其他其他方案的模糊多數壓倒的程度。選項 a_i 的 QGNDD 則使用下列公式計算：$QGNDD_i = F_Q$ (1-max$\{P_{ji}^{c} - P_{ij}^{c}, 0\}$,

j=1,…,n, $j \neq i$)。至於選項 a_i 的 QGDD，我們利用下列公式計算：$QGDD_i=F_Q$ (P_{ij}^c , j=1,…,n, $j \neq i$)。

透過這兩個壓倒程度，選擇程序將使用以下的規則去獲得方案的優先順序：

1. 如果存在 UND(Unfuzzy Non-Dominance) 元素 (i.e. QGNDD(a_i)等於 1)，使用 QGDD 所產生的排序以獲得方案優先順序。
2. 否則，使用 QGNDD 來排序獲得方案優先順序。

參、協調運算的運作流程

在 CPS 架構中，運算需求者端會將運算工作需求發布到協調者端，空閒的運算端會主動和運算需求者端進行溝通連結，接著運算需求者端才分配工作給空閒的運算端進行運算。因此本研究在運算需求者端加入運算協調模組採用 RPGDM 來解決“選擇當時具有最佳資源的運算端來執行運算工作的決策問題”。要解決決策問題，首先必須先定義方案集合和專家集合。由於運算協調模組的目的在找出最佳效能的運算端，所以方案集合中會包含所有決策當時可提供運算能力的運算端，而專家集合則定義為包含「CPU 速度」、「CPU 使用率」和「記憶體使用率」，這些參數基本上是影響程序執行時間的關鍵特性。一般來說，擁有上述三項較佳參數值的運算端如果全力運算會有較快的執行時間。然而，CPS 架構是希望在不影響運算端正常運作的情況下，有效運用閒置的運算端來執行需較大運算能力的複雜運算，因此擁有較佳參數值的運算端並不保證當時會有較快的執行時間，同時，各運算端的工作平均執行時間也會影響運算端取得工作數量的優先權。所以除了之前提到的三項專家元素外，「工作平均執行時間」也將包含在專家集合中。綜合以上所言，運算協調模組中的選項集合和專家集合的輸入參數將如下所定義：

專家集合 E={ CPU 速度, CPU 使用率, 記憶體使用率, 工作平均執行時間}

方案集合 A={當時可提供運算能力的運算端}.

為了取得專家集合中各運算端參數的最新數值以應用在 RPGDM 上，本研究修改 CPS 運算端的程式，使運算端定期(例如每兩分鐘)以 XML 訊息格式回報各運算端的最新數值。此 XML 訊息的格式如圖 2 所示。

```
<QoS>
  <ID>
    <IP>140.126.107.250</IP>
  </ID>
  <CPU>
    <Usage>50(%)</Usage>
    <Speed>1000(MHz)</Speed>
  </CPU>
  <Memory>
    <Usage>50(%)</Usage>
    <Physical>256(MB)</Physical>
    <Virtual>512(MB)</Virtual>
  </Memory>
  <Network>
    <RoundTripTime>10(ms)</RoundTripTime>
  </Network>
</QoS>
```

圖 2：XML 訊息格式

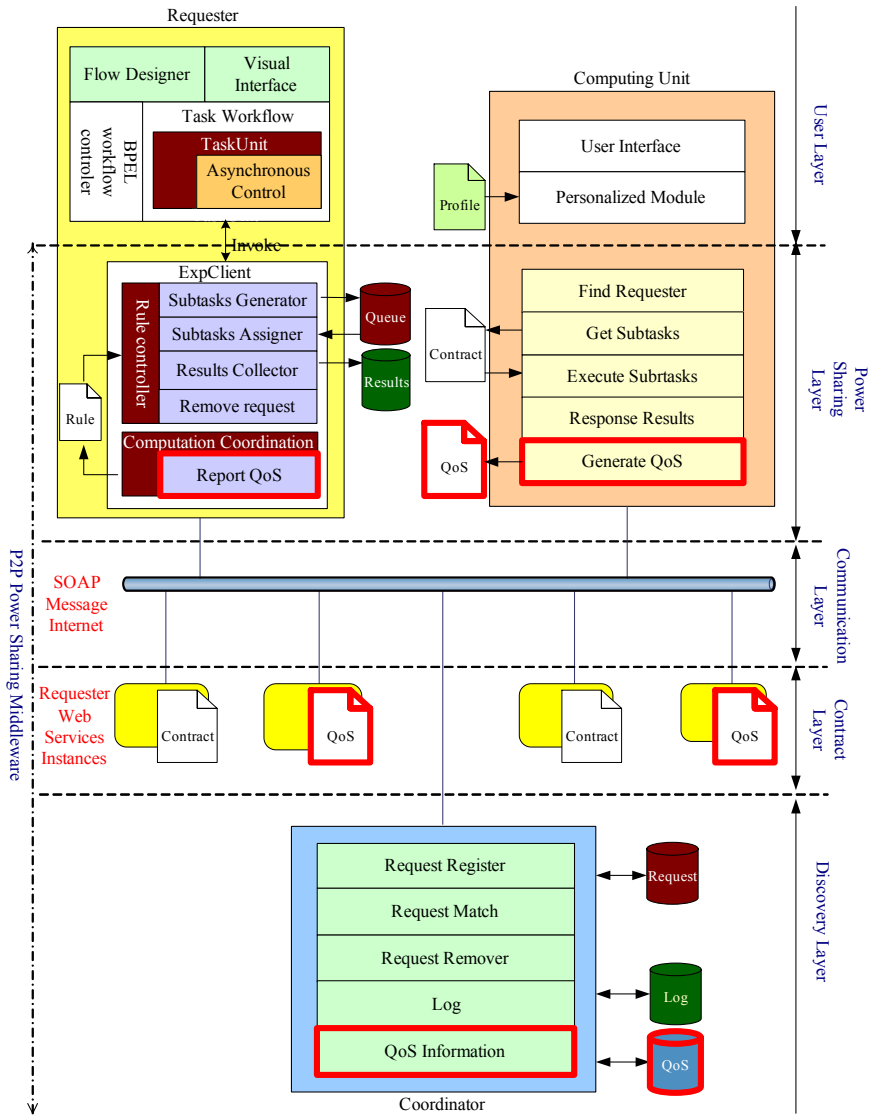

圖 3：具有協調運算功能的 CPS 架構階圖(紅粗線部份是運算協調相關元件)

此 XML 訊息也兼具定時通知運算需求者告知某個運算者存在的功用，可以分派工作給他。由於 CPS 架構模組化為五個階層，為了要整合運算協調模組到此模組架構中，運算能力分享層和溝通層兩者將被修改，以包含運算資源相關的模組和一些訊息的傳遞。其中在運算能力分享層上，運算協調模組則被加入運算需求者端，使其能進行 RPGDM 的運作，而「Generate QoS」模組將加入運算者端，使運算者能回報最新的運算資源數值。定時回報的 XML 訊息定義則加入於溝通層。圖 3 描述加入協調運算能力的 CPS 架構中五階層間的關係，紅粗線部份代表本研究所設計的運算協調相關元件。

在收集運算端資源的最新數值之後，運算協調模組將會使用 RPGDM 決策方法來控制各運算端工作量的分派。基本上來說，運算協調模組會使用佇列來儲存分派給各運算端的工作數量，一旦最快工作執行時間的運算端將佇列中自己所屬的工作執行完畢，RPGDM 將再生成新的工作分派給運算端執行。

各運算端只負責回報監視資源的數值，運算協調模組則負責將這些從運算端取得的數值依各資源類別加以排序，以獲得各資源類別的偏好排序結果。如第二節的說明，運算協調模組中 RPGDM 的運作共分為轉換、彙整、探索三階段。在轉換階段時，利用第二節所提到的公式(1)將之前得到的偏好排序結果轉換成模糊偏好關係。在 RPGDM 的彙整階段中，將使用兩次彙整偏好關係的運算，這樣做的原因是因為最初工作未分派時，尚無法取得各運算端的平均工作執行時間，所以第一次彙整偏好關係的運算，運算協調模組將集合 CPU 速度、CPU 使用率和記憶體使用率的模糊偏好關係，利用第二節所提到的公式(4)以取得彙整偏好關係，此時所使用的 OWA 運算子的權重向量是使用單一語意量詞(Identity qualifier)，因此預設為[1/3, 1/3, 1/3]，表示 CPU 速度、CPU 使用率和記憶體使用率三者比重一致。此權重在幾次子工作(subtask)執行完成後，將透過子工作執行狀況的實驗記錄，計算 CPU 速度、CPU 使用率和記憶體使用率三者和子工作(subtask)執行時間的相關係數，做動態調整。第二次彙整偏好關係的運算中，使用第一次所得到的彙整偏好關係和平均工作執行時間，再進行一次彙整偏好關係的運算，產生最終的彙整偏好關係，第二次整合偏好關係的運算中，同樣使用假設單一語意量詞而將 OWA 運算子權重向量預設為[1/2, 1/2]。當取得最終的彙整偏好關係後，在探索階段中，利用 QGNDD 和 QGDD 兩種模糊評比法，依照第二節第三階段所提的選擇規則來計算各運算端該被分派的工作數量，因為執行速度較慢的運算端會有較低的 QGNDD 或 QGDD 數值，所以本研究讓擁有最低的 QGNDD 或 QGDD 數值的運算端執行一個子工作，而其他運算端該執行的子工作數量，則依照其本身的 QGNDD 或 QGDD 數值為最低 QGNDD 或 QGDD 數值的倍數來決定工作數量。

肆、實驗方法與結果分析

一、實驗方法與環境

為了測試 CPS 系統的效能，本研究使用以離散小波轉換(Discrete Wavelet Transform , DWT)為基礎的浮水印演算法(Tsai 2004)進行測試評估，小波轉換的原理是利用濾波器來過濾和構成數位影像的信號。在濾波器之中，分解濾波器(Analysis filters)用於區別數位影像中的低頻訊號和高頻訊號，而合成濾波器(Synthesis filters)則利用低頻訊號和高頻訊號構成影像。所有以離散小波轉換為基礎的浮水印演算法包含分解(Decomposition)、嵌入(Embedding)、還原(Reconstruction)、偵側(Detection)四個程序(Cox et al. 1997 ; Wang et al. 2002)，然後使用以相依性(Correlation) 統計為基礎的相似性函數(Similarity function)透過比對原始浮水印和被攻擊影像浮水印的計算，來確認數位影像的授權依據。利用這個演算法數位版權能夠被保護，而且所有權資訊能在攻擊之下得以保存。圖 4 是利用離散小波轉換分解 Lena 影像的範例，浮水印已被嵌入在編號 9 的區域。由於濾波器是 DWT 浮水印演算法的重要關鍵，因此需要大量的運算能力從大量的濾波器組中，找尋最適合的濾波器來使用，因此很適合在 CPS 架構下實行。

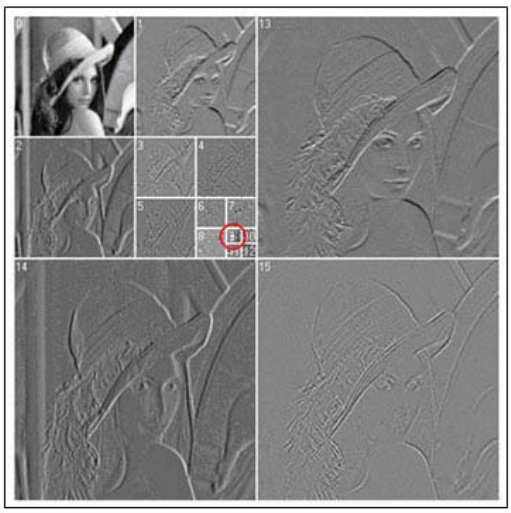

圖 4：使用 DWT 分解 Lena 影像的範例

實際實驗部分，採用了 76177 組濾波器來解浮水印，此步驟相當依賴電腦的運算能力，再加上可把 76177 組濾波器分解成子工作方式進行，因此是一個可批次化的高運算依賴工作，很適合以 BPEL 設計重覆流程作運算處理。在實驗中 76177 組濾波器被切成以 100 個為一組的子工作，所以有 762 組子工作需要被運算。每個子工作的內容是分別使用不同的濾波器，對加入浮水印的 raw 檔以及 JPEG 2000 的圖檔解出浮水印，並計算出對應的相關係數值。

因為 CPS 架構是強調可有效使用低成本的軟體和系統需求來建構，因此設計實驗時，便選擇五台執行環境差異頗大的個人電腦作為運算端，如表 1 所示。所有實驗皆採用這 5 台電腦擔任運算端加入實驗以確保運算端靜態資源環境的一致性。

表 1：實驗中所加入的 5 個運算端列表

運算端	CPU 速度(GHz)	記憶體(MB)
A	P4-2.4	1024
B	P4-3.2	512
C	P3-1.0	256
D	P4-3.2	512
E	P3-0.67	384

二、實驗一

此實驗的目的在比較 CPS 架構沒有使用任何的協調機制(CPS w/o Coordination)、使用在(Tsai et al. 2006) 的協調機制(CPS w/ Coordination)、使用固定權重[1/3,1/3,1/3]的 RPGDM 協調機制(CPS w/ Fuzzy Coordination)和 RPGDM 使用動態權重等四種機制下的運算效能，各機制的特點如表 2 說明。此實驗也假設每個參加實驗的運算端都只專職於此項運算工作，也就是說各運算端都盡量避免有非實驗相關用途的使用。

實驗分別以 50 組、100 組、200 組和 300 組子工作進行測試，每組實驗各進行三次取平均時間，實驗結果如表 3 所示。在 CPS w/o Coordination、CPS w/ Coordination、CPS w/ Fuzzy Coordination(固定權重[1/3,1/3,1/3])和 CPS w/ Fuzzy Coordination(動態權重)四種情況下，由表 3 的數據觀察，四者的工作平均完成時間相互差距大約在 10 秒到 700 秒不等。若以圖 5 來看，大體上來說，其完成工作的效率依序為 CPS w/ Fuzzy Coordination(動態權重)、CPS w/ Fuzzy Coordination(固定權重[1/3,1/3,1/3])、CPS w/ Coordination，最後為 CPS w/o Coordination。綜合以上所言，雖然在各運算端專職運算工作的條件下，採用協調模組並沒有使運算完成時間有大幅的改善，但與採用其他運作機制的情況，CPS w/ Fuzzy Coordination(動態權重)有較佳的表現。

表 2：CPS w/o Coordination、CPS w/ Coordination 和 CPS w/ Fuzzy Coordination 所代表的意義。

CPS w/o Coordination	CPS w/ Coordination	CPS w/ Fuzzy Coordination
在運算分享架構中，不加入任何運算協調機制，每個運算端每次只取一個子工作(subtask)，工作完成後再向需求者端取下一個子工作。	在運算分享架構中，加入簡單的運算協調機制，當某運算端的平均工作執行時間小於 180 秒時(即效能較佳)，便將剩下或需重做的工作分派給此運算端。	在運算分享架構中，加入 RPGDM 運算協調機制，運算需求者端依照當時運算端的綜合效能，動態分派工作數量，此機制將分別使用固定權重和動態權重進行實驗。

表 3：CPS w/o Coordination、CPS w/ Coordination 和 CPS w/ Fuzzy Coordination 的工作完成結果(運算端專職運算工作)

工作平均完成時間 (sec) / Tasks 數量 (組)	CPS w/o Coordination	CPS w/ Coordination	CPS w/ Fuzzy Coordination [1/3,1/3,1/3]	CPS w/ Fuzzy Coordination [動態權重]
50	1752.3	1770	1624.3	1589
100	3108	3052	3151.3	3142
200	6302.3	6158.3	6299	6085.7
300	9635.3	9162	9015	8898.3

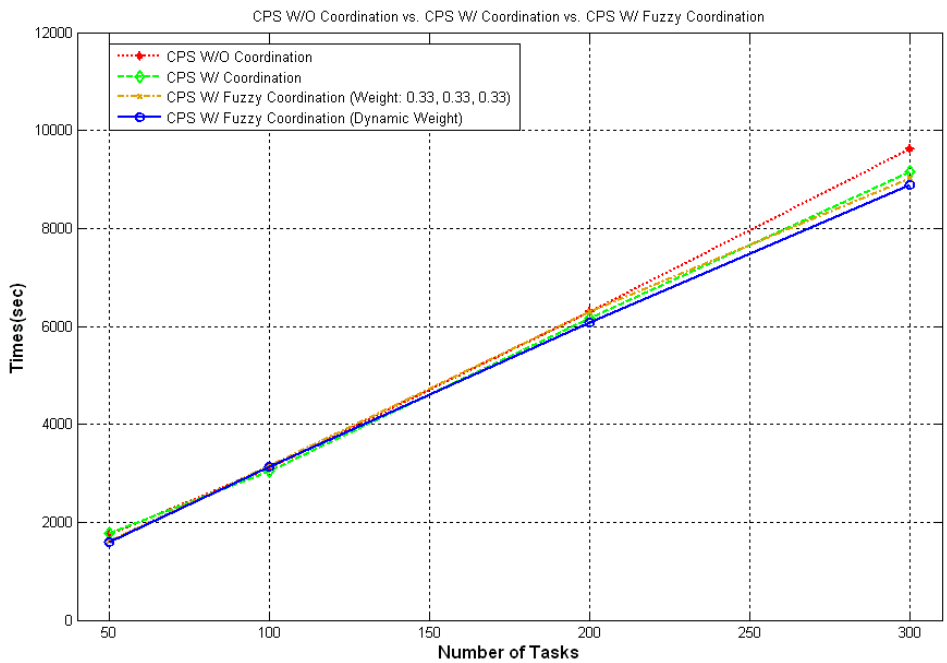

圖 5：CPS 進行小波轉換濾波器估算的總運算時間(50、100、200 和 300 組)

三、實驗二

實驗二的目的也是在比較 CPS w/o Coordination、CPS w/ Coordination、CPS w/ Fuzzy Coordination(固定權重[1/3,1/3,1/3]和[動態權重])四種機制使用下的運算效能。但

實驗時會讓每台加入實驗的電腦，除了開啟運算端處理子工作之外，並以隨機的方式執行 SuperPi 軟體，觀察四種情況工作完成時間的變化情形。SuperPi 是一個計算圓周率的軟體，可指定計算到小數點後的位數，當它計算時會使用大量的 CPU 運算，因此常當作測試系統效能的工具。這樣做的目的是模擬運算端在執行正常工作的影響下，協調機制如何回應處理。因為本研究發現若運算端都只專職於此項運算工作時，效能最好的電腦通常對工作完成時間有決定性的影響，然而在實際應用上，所有網路分享架構的宗旨，都強調是在不影響原有工作下，分享閒置資源。

實驗二分別以 50 組、100 組、200 組和 300 組子工作進行測試，每組實驗各進行三次取平均時間，實驗結果如表 4 所示。在 CPS w/o Coordination、CPS w/ Coordination、CPS w/ Fuzzy Coordination(固定權重[1/3,1/3,1/3])和 CPS w/ Fuzzy Coordination(動態權重)四種情況下，由表 4 的數據觀察，四者的工作平均完成時間差異較明顯，相互差距大約在 50 秒到 2000 秒不等。若以圖 6 來看，大體上來說，雖然 CPS w/ Coordination 和 CPS w/ Fuzzy Coordination(固定權重[1/3,1/3,1/3])兩者的實驗數據非常接近，但可以發現 CPS w/ Fuzzy Coordination(動態權重)在各組情況上，整體比其他三種情況效能要好，所以採用 RPGDM 協調模組，可以在有受到干擾的環境下，保有較佳的工作完成時間。

表 4：CPS w/o Coordination、CPS w/ Coordination 和 CPS w/ Fuzzy Coordination 的工作完成結果(運算端隨機執行 SuperPi)

工作平均完成時間 (sec) / Tasks 數量 (組)	CPS w/o Coordination	CPS w/ Coordination	CPS w/ Fuzzy Coordination [1/3,1/3,1/3]	CPS w/ Fuzzy Coordination [動態權重]
50	2127	2031.7	2043	1994.7
100	4379.3	4192.3	4170	3985
200	8237.7	7853.3	7937.7	7559.3
300	13420	11872.7	11783.3	11257.3

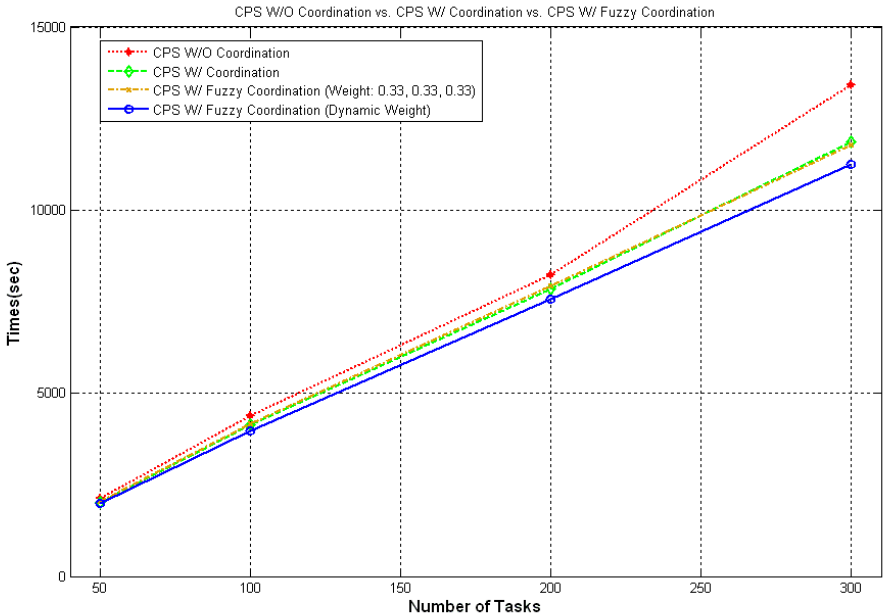

圖 6：CPS 進行小波轉換濾波器估算的總運算時間(50、100、200 和 300 組)

四、實驗三

實驗三的目的，在衡量 CPS w/o Coordination、CPS w/ Coordination 和 CPS w/ Fuzzy Coordination 三種機制個別使用時，整體運算處理工作的穩定程度。此模擬以 300 組子工作各進行 5 次實驗，而且讓每台加入實驗的電腦，除了開啟運算端處理子工作之外，並以隨機的方式執行 SuperPi 軟體，用來模擬使用者是正常使用電腦的情況下，觀察三種機制使用時，子工作完成時間的變化情形。另外，此實驗在 CPS w/ Fuzzy Coordination 機制使用時，加入運算協調的權重變動，以觀察權重對穩定度的影響。因此除了有最初固定權重[1/3,1/3,1/3]之外，還分別使用固定權重[0.4,0.5,0.1]和動態調整權重方式模擬。

固定權重[0.4,0.5,0.1]是利用過去的實驗記錄資料，以 CPU 使用率和記憶體使用率為兩個自變數，來計算對因變數(子工作執行時間)的判定係數，其代表意義是子工作執行時間，能夠用 CPU 使用率和記憶體使用率兩變數解釋其變動的程度。本研究利用 Minitab 統計軟體針對過去的實驗記錄做計算，求得迴歸方程式為 C1 = 10.3 + 0.648 C2 + 0.795 C3，其中 C1, C2, C3 分別代表子工作執行時間，CPU 使用率及記憶體使用率；而判定係數(R-Squared)求得為 57%，也就是說 CPU 使用率和記憶體使用率可解釋 57%(約近 60%)的子工作執行時間變動情形。

此外，因 CPU 速度為常數，無法使用相關分析計算，但運算工作非常依賴 CPU 速度，所以假設 CPU 速度解釋剩下的 40%變動，再加上 Minitab 計算出 CPU 使用率和記憶體使用率對於子工作執行時間的相關係數分別為 0.711 及 0.235，依大約比例將權重調為[0.5, 0.1]，因此，CPU 速度、CPU 使用率和記憶體使用率的權重被設為[0.4,0.5,0.1]進行實驗。

至於權重動態調整，則進一步將判定係數的計算加入協調機制中，以便能隨時依據實驗記錄結果進行權重調整。

實驗結果如表 5 所示，表 5 括弧內的數字是與該機制平均時間的差異。將表 5 的數據製成圖 7，由圖 7 可以看出 CPS w/ Fuzzy Coordination 在不同權重使用下，工作完成時間差異幅度均比 CPS w/o Coordination 或 CPS w/ Coordination 的工作完成時間差異幅度小，顯示運作情況較為穩定。由此可見，採用 Fuzzy 運算協調能在運算端電腦正常使用時，依舊能夠維持穩定的運作。

此外，由圖 7 也可以看出 CPS w/ Fuzzy Coordination 機制，如果使用依據實驗結果歷史記錄計算所得的固定權重[0.4,0.5,0.1]，或更進一步將權重設計成能隨時依據實驗記錄進行的權重動態調整，其運作的穩定度均更勝於 CPS w/ Fuzzy Coordination(固定權重[1/3,1/3,1/3])，也就是說根據運算端之前的運算記錄所作的權重改變，對運作穩定具有影響性。圖 8 是權重動態調整機制和其他兩種機制各運算 20 次實驗結果，結果進一步證實 CPS w/ Fuzzy Coordination 加上權重動態調整機制，可享有整體服務品質最高的穩定性。

表 5：CPS w/o Coordination、CPS w/ Coordination 和 CPS w/ Fuzzy Coordination 的 300 組工作完成結果

完成時間 (sec) / 機制	1	2	3	4	5	平均完成時間
CPS w/o Coordination	19102 (5472.8)	11731 (-1898.2)	11712 (-1917.2)	14177 (547.8)	11424 (-2205.2)	13629.2
CPS w/ Coordination	12132 (94.8)	11393 (-644.2)	13288 (1250.8)	11824 (-213.2)	11549 (-488.2)	12037.2
CPS w/Fuzzy Coordination [1/3,1/3,1/3]	12209 (180.2)	12585 (556.2)	12247 (218.2)	11387 (-641.8)	11716 (-312.8)	12028.8
CPS w/Fuzzy Coordination [0.4,0.5,0.1]	11799 (-105.4)	12227 (322.6)	11388 (-516.4)	11956 (51.6)	12152 (247.6)	11904.4
CPS w/Fuzzy Coordination [動態權重]	11753 (35.8)	11924 (135.2)	11704 (-84.8)	11692 (-96.8)	11871 (82.2)	11788.8

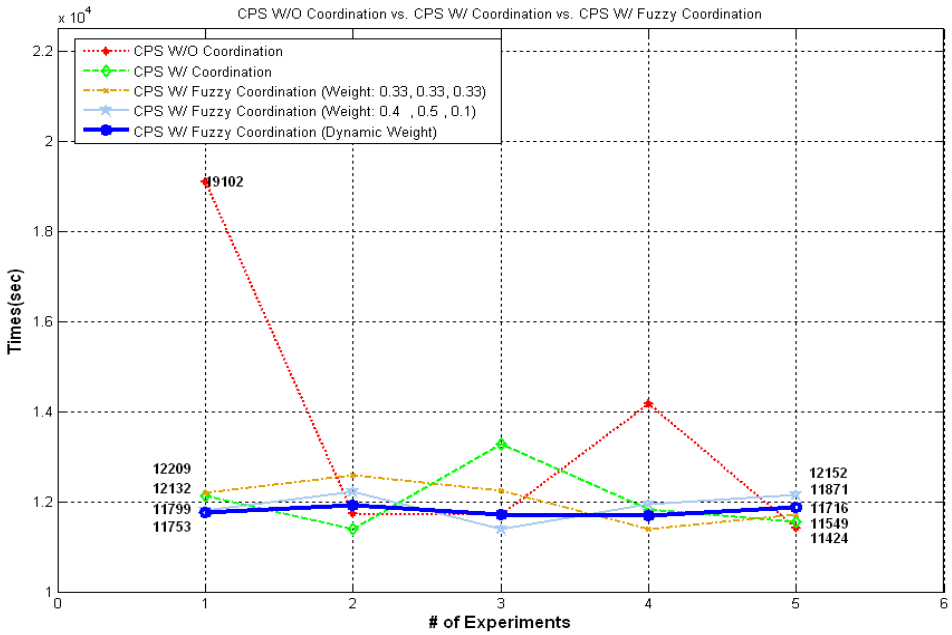

圖 7：各種機制執行 5 次 300 組的實驗結果

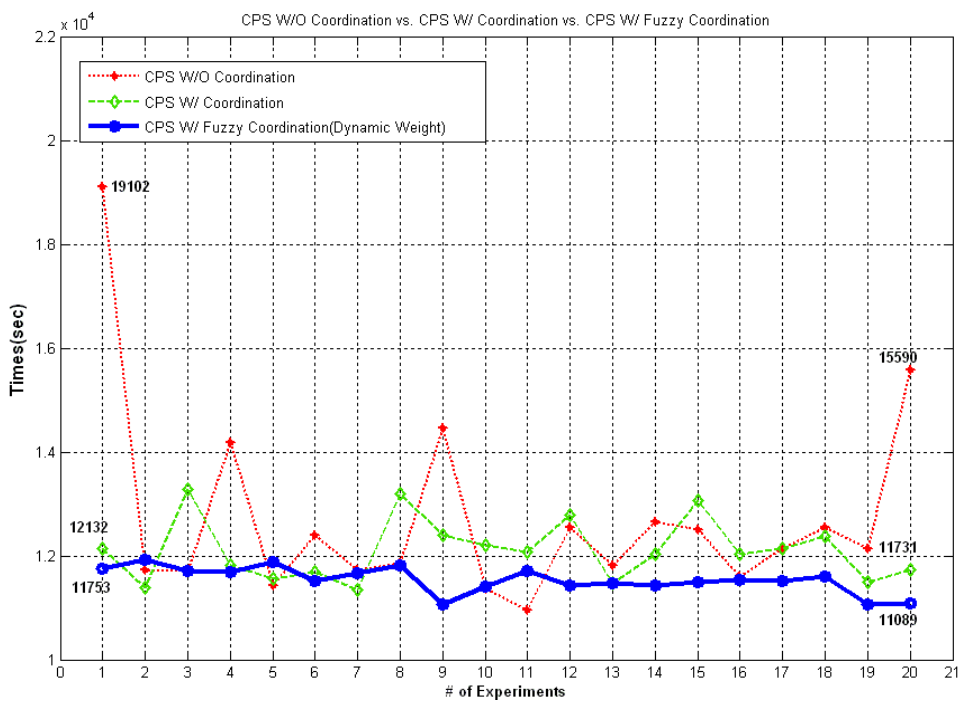

圖 8：CPS w/o Coordination、CPS w/ Coordination 和 CPS w/ Fuzzy Coordination (動態權重) 300 組 20 次實驗結果

伍、討論

一、協調機制對需求端的影響

由於 RPGDM 方法主要是專家針對方案間的優劣程度經過模糊轉換、彙整及探索三個階段的比較運算而決定，因此假設有 *m* 個專家針對 *n* 個方案做決策選擇，則轉換、彙整及探索階段所需的比較運算次數分別將是 *mn(n-1)*、*mn(n-1)* 及 *n(n-1)*次，以 time complexity 表示將是 $O(mn^2)$，由於本研究的方案定義為可使用的運算端，因此當運算端越多時，協調運算所需的時間就越久。如果假設 CPU 執行時脈是 2.0GHz 且比較運算所需的時間是兩個 clock time，則一個比較運算的時間約為 1ns，因此當運算端如果有 10^5 台時，則 RPGDM 方法總共所需時間約數十秒左右，此時相對於研究實驗所用的小波轉換運算，每個子工作在單一運算端執行平均約需 180 秒就顯巨大。然而若考慮 CPS 架構是運作在一個組織內部信賴網路時，運算端數目如果是百台左右，則所產生的時間負擔約數十毫秒是相對微小，因此協調機制所造成的額外時間負擔相對於所帶來的完成時間縮短及穩定性增加，應該是可以接受且值得採用。

二、協調機制對運算端的影響

本研究使用 XML 格式定時回報運算端的資源參數到需求端，這些資訊是運算協調機制運作時的依據，然而這些動作是否會降低運算端之效能？其對效能的影響可從系統面及網路面來探討。

就系統面來看，由於本研究所需的資源參數不管是 CPU 速度、CPU 使用率和記憶體使用率，皆直接取自系統的效能記錄，因此執行上只是一個記憶體存取的指令；而執行時間的參數計算，因需要記錄子工作開始與結束時間，雖然需要兩個記憶體存取的指令，但也是減法的直接運算，因此在系統面上效能影響不大。

至於在網路面的影響，因整個 XML 訊息(如圖 2)大小約為 256 位元組左右，且預設每 120 秒傳送一次，所以不會佔用太多網路頻寬。但如果分享環境所執行的子工作會傳回大量的運算結果時，這種密集且規律的訊息傳送將會互相影響運算端傳回結果，有時會造成需求端誤認運算端已停止執行的情況。然而本研究的協調機制是運算端傳回結果且該運算端的子工作執行佇列已空時，才會執行。這意謂如果不考慮運算端回報存活訊息的功能，則對每台運算端最新資源參數的需求唯有在傳回運算結果後才會產生，因此遇到網路因頻繁的 XML 訊息產生影響時，我們可調高訊息預設傳送的間隔，來舒緩網路的擁塞情況。

三、權重對運算效能的影響

此外，本研究所使用的 RPGDM 方法原本依賴排序加權平均運算子(OWA)來彙整各專家對方案的偏好程度，其用意是避免專家對某方案的極端偏好，進而對整個共識結果產生巨大影響，因此 RPGDM 方法強調必須慎重選擇權重運算子，以免對最後方案的選擇產生誤差。然而在本研究因專家是定義為運算端的資源參數，所以可假設其行為是理性與公正。因此就可對專家(資源參數)的權重相對於歷史實驗結果做相關程度計算，實驗也發現這樣的權重調整，的確對結果產生良好的影響。

陸、結論

本研究提出以模糊理論為基礎的群體決策程序(RPGDM)來解決分享式環境中運算資源的協調問題，使 CPS 架構具有服務品質(Quality of Service)的即時運算協調能力，來提升整體架構的執行效能，適合在企業內的信賴網路中執行程式的運算，並使用 Web Services 和 BPEL 提供可圖形化發展環境和流程控制管理的能力。此機制並實作在分析數位浮水印強度的 filter bank 選擇，用以測試執行效能；從實驗結果發現，RPGDM 方法不只在靜態-只執行分享工作-環境中勝過先到先服務(First-Come-First-Serve)機制的運算效能，在運算端正常使用環境下，也能保持較好的運算效能及穩定性，因此很適合當作分享式環境的協調機制。

致謝

感謝諸位評審委員對本文所提供的協助與寶貴建議。本文承蒙國科會提供研究經費(NSC 94-2416-H009-018 和 NSC 95-2416-H009-027)，特此致謝。也感謝交通大學商務多媒體研究室曾立信、楊千毓、楊博宇、陳錡樂及羅子文等同學的協助，使本研究能夠持續進行。

參考文獻

1. Amnuaykanjanasin, P. and Nupairoj, N. “The BPEL Orchestrating Framework for Secured Grid Services”, *ITCC2005*, April 04-06, 2005, pp 348-353.
2. Arrastia, Carlos, Fernández, Alejandro.“Peer to Peer Web Services,” *Web Services Architecture Seminar*, 2004.
3. Chiclana, F., Herrera, F., and Herrera-Viedma, E. “Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations, ” *Fuzzy Sets and Systems* (97:1), Jul 1998, pp 33-48.

4. Chiclana, F., Herrera, F., and Herrera-Viedma, E. “Integrating multiplicative preference relations in a multipurpose decision-making model based on fuzzy preference relations,” *Fuzzy Sets and Systems* (122:2), Sep 2001, pp 277-291.
5. Cox, I.J., Kilian, J., Leighton, F.T. and Shamoon, T. “Secure Spread Spectrum Watermarking for Multimedia,” *IEEE Transactions on Image Processing* (6:12) 1997, pp 1673-1687.
6. Kacprzyk, J. “Group Decision-Making with a Fuzzy Linguistic Majority,” *Fuzzy Sets and Systems* (18:2), 1986, pp 105-118.
7. Loo, Alfred W. “The Future of Peer-to-peer Computing,” *Communications of The ACM* (46:9), September 2003, pp 57-61.
8. Saaty, T. L. *The Analytic Hierarchy Process*, McGraw-Hill, 1980.
9. Tanino, T. “Fuzzy Preference Orderings in Group Decision-Making,” *Fuzzy Sets and Systems* (12:2), 1984, pp 117-131.
10. Tsai, M.J. ”Filter Bank Selection for the Ownership Verification of Wavelet Based Digital Image Watermarking,” *ICIP 2004*, Oct 24-27, May, 2004 , pp 3415-3418.
11. Tsai, M.J., Wang, C.S., Yang, P.Y. and Yang, C.Y. “A Collaborated Computing System by Web Services based P2P Architecture,” *Lecture Notes in Computer Science* 3865, 2006, Computer Supported Cooperative Work in Design II pp194-204.
12. Wang, Y., Doherty, J.F. and Dyck, R.E. ”A Wavelet-Based Watermarking Algorithm for Ownership Verification of Digital Images,” *IEEE Transactions on Image Processing* (11:2), 2002, pp 77-88.
13. Yager, R.R. “On ordered weighted averaging aggregation operators in multicriteria decisionmaking,” *Systems, Man and Cybernetics, IEEE Transactions on* (18:1) 1988, pp 183-190.
14. Yager, R.R. "Families of Owa Operators," *Fuzzy Sets and Systems* (59:2), Oct 1993, pp 125-148.
15. Zadeh, L. “A computational approach to fuzzy quantifiers in natural languages,” *Computing and Mathematics with Applications* (9), 1983, pp 149-184.
16. Zhang, L.J., Li, H., Lam, H. ”Toward a Business Process Grid for Utility Computing,” *IT Pro*, Sept-Oct 2004, pp 61-63.