行政院國家科學委員會專題研究計劃成果報告

分散式物件導向程式設計、實作與效能分析－在 CORBA 平台上以 STPN 為實例的探討

Distributed object-orient programming design, implementation and performance analysis – A case study of STPN on CORBA

計劃編號：NSC 88–2213–E–032–004
執行期限：87 年 8 月 1 日至 88 年 7 月 31 日
主持人：陳伯榮　私立淡江大學資訊工程系

一、中文摘要

我們經由在 OMG 的分散式物件環境 CORBA 中實作隨機過程時間派翠網路 (Stochastic Timed Petri Nets (STPN))，學習如何整合物件導向的優點於 Foster 的分散式程式設計方法中。

我們選擇 OMG 的分散式計算系統 CORBA 為工作平台，主要是因為它幾乎能夠在每一種硬體和網路間運作並且也幾乎能夠整合、連結每一種程式語言，因此原有的程式碼都能夠重複使用。

另一方面，STPN 是最適合描述與分析分散式系統的模型之一。但是多數分散式系統過於複雜而難以驗證和分析，因此我們著重於研究新的設計技巧的理論和實作，進而完成在 STPN 中用來驗證、分析的三個重要演算法。對於選擇這三個演算法的理由是因為它們是作定性分析(Quality analysis)和定量分析(Quantity analysis)的主要工具。

關鍵詞：分散式計算系統，隨機過程時間派翠網路，分散式物件環境

Abstract

We learn how to integrate object-orient paradigm into Foster's distributed program design method, by building the well-known Stochastic Timed Petri Nets (STPN) on the OMG's distributed object environment CORBA.

We select the OMG's distributed computing system CORBA, mainly because it can run on almost every hardware platform and network and it can link almost every programming languages, thus can reuse what code you have already.

On the other hand, STPN is one of the best-known models for distributed system. But the complexity of many distributed system are often too hard to validate and analyze. Therefore, we focus on both the theoretical and experimental studies of the new design technique to build three important algorithms in STPN. One reason for choosing these three algorithms is that there are the main tools in STPN to analyze system both qualitatively and quantitatively.

Keywords: distributed computing system, Stochastic Timed Petri Nets, distributed object environment

二、緣由與目的

本計畫之主要目的有二，一是學習及研究如何整合物件導向的優點於 Foster 的分散式程式設計方法中、二.在 CORBA 上實作 STPN 中用來驗證、分析的三個重要演算法。

在[1]中 Bal 等廣泛的定義分散式計算系統(Distributed Computing Systems)為多個不共有記憶體的處理器，而其處理器間僅藉著互送訊息來溝通。近年來由於電腦硬體的發展趨勢及網路系統的普遍性由工作站及個人電腦組成的地域性、廣域性網路系統，是被用來做為分散式計算系統的主要硬體架構。而此架構也正是大部份分散式作業系統的主要開發硬體平台。
分散式計算系統中解決問題的方法和傳統的的方法，主要的不同在於平行化程度(parallelism)、內部程序的聯繫與同步(interprocess communication & synchronization)和局部性錯誤(partial failures)，針對此三個議題已發展了許多分散式計算系統上的分散式語言。

在[2]中，分散式物件基礎程式設計系統(Distributed Object-Based Programming System，簡稱 DOBPS) 為提供物件為基礎的分散式計算系統。它除了像分散式作業系統一樣需管要理資源外，也負責管理物件和物件之間的呼叫(invocation)。DOBPS 最大的優點即以物件來做為程式設計師和電腦硬體之間的橋樑，進而免除了傳統的分散式計算系統中建立或執行分散式程式的許多問題。同時 DOBPS 也藉著物件導向簡化了程式介面，使得程式設計師更方便描述/解決問題。

另一方面整合並行/分散式程式設計(concurrent/distributed programming) 和物件導向/基礎程式設計(object-oriented/based programming)不再僅是學界研究室的興趣。MIS、Telecommunication 和其他主要的電腦使用業界也希望在短期內能夠整合成功以利開發相關產品。當然要整合分別以簡易型程序(thread)和物件為主體的程式語言環境並不是件容易的事，譬如說：

- 如何利用類別和物件來構成/描述一分散式系統。
- 什麼是最基本/需要的並行方法。
- 如何將眾人皆知的物件導向設計開發方法進一步延伸到並行/分散式程式設計的領域。

在 [3] 有 8 件作品對並行式物件導向程式設計做出了拋磚引玉的呼聲。

在 1989 年，各主要電腦業界為了提升物件技術在軟體工程的應用和建立、發展分散式物件計算環境的模式和標準介面，因而共同成立了 OMG (Object Management Group)。

CORBA(Common Object Request Broker Architecture)為 OMG 在一九九一年十二月提出之物件導向分散式工作環境規格。此規格多次被修訂，目前最新的版本為一九九九年六月之 CORBA 2.3 版。

CORBA 它同時也是一個非常成熟的分散式作業平台。CORBA 能穿越網路、程式語言、元件界限以及作業系統。而 CORBA 會如此重要的原因也就是由於它在複雜的網路世界裡扮演著仲介的角色，並具有整合其它已存在的客戶/服務中介軟體 (client/server middleware) 的潛力。相對於傳統分散式架構如 DCE (Distributed Computing Environment) 及 ONC+ (Open Network Computing)上面的發展環境均無類別或繼承之觀念及功能, CORBA 規格最大的特色即為物件導向觀念的導入。

三、結果與討論

我們依 STPN 三個重要的子系統及演算法來分開討論。

(a) 分析解

我們採用以"轉移動作"（Transition Decomposition）為導向的資料分割法，將問題分解成數量相當並且大小相近的工作單元。共分成狀態擴展運算單元和狀態整合運算單元兩大類型，狀態擴展運算單元的數目是根據我們切割的準則及大小而決定，每一個擴展運算單元都擁有部分派翠網路（Sub Petri Net）的資料。當運算單元決定後我們分別賦予不同的運算功能：在狀態擴展運算單元中有"共用狀態訊息接收"、"狀態擴展"、"共用狀態訊息傳送" 和 "檢查重複狀態四種運算功能；在狀態整合運算單元中有 "整體狀態收集" 和 "馬可夫矩陣轉換"兩種運算功能。

每一個狀態擴展運算單元主要是以現有的資料擴展出所有的本地資料，也就是每個工作單元會將一個狀態所能擴展的所有狀態展出。依照我們的分割方式會發生同一個位置資料同時和兩個以上的轉移動作有所關聯，並且轉移動作可能分屬不同的工作單元。這樣會造成工作單元間共用位置資料，當狀態擴展更動到共用的資料時，我們就必須透過通訊來維持資料的一致。一個工作單元更改到共用資料時會將更動訊息傳遞給所有相關的工作單元，相對的也會收到其他工作單元更改到共用資料的訊息。另外，雖然每次擴展的順序不同，但是可能會得到相同的結果，這時藉由檢查重複出現的狀態來減少重複的計算。

當所有的狀態擴展運算單元工作結束，擴展出所有的地方資料時，經由狀態整合運算單元將完整的狀態關係圖（也就是可到達圖）整合出來，另外矩陣中的每個元素的位置會隨著由派翠網路找尋到達圖時，所找尋順序的不同而有所不同，這裡我們採用廣度優先擴展（Breadth First Expansion）的方式，利用圖形的重新走訪寫出聯立方程式組來求得分析解。(詳細報告請參閱[4])

(b) 可到達性分析

在展狀態的過程中，我們將採取把問題分割 (partition) 成三個部分來合作進行：第一部分 建構與維護可到達樹及最小累堆樹，第二部分 平行地狀態擴展，第三部分 平行地搜尋（檢查）狀態是否重複及平行地插入新狀態，如圖3.1所示。其中包含兩個重點：(詳細報告請參閱[5])

(1) 在已經產生但尚未擴展的狀態的狀態集合中，要如何選出最適當的狀態來擴展。

(2) 設計適當的雜湊函數（hash function）來協助系統平行地檢查狀態是否重複，以減少檢查時所需花費的大量時間。

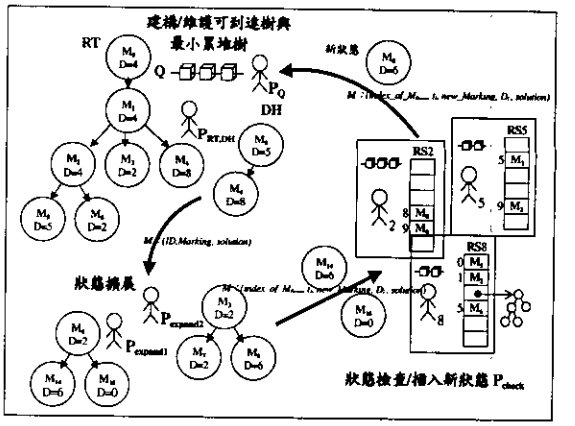

圖3.1 可到達性問題的平行程式設計 - 問題分割

(c) 模擬解

本系統的切割方式是以轉移動作為主，因為 STPN 之所以能動作，完全拜轉移動作被啟動(fire)所賜，藉著啟動使位置(Place)中的小圓點(Tokens)個數變化，亦發生了系統不同的狀態，因此，於切割方法上，除了以轉移動作 為主外，將與該轉移動作相關的位置 歸於同一個群集(Group)，切割後，每個群集均不失仍為一個 STPN，因此，每個群集基本運作原理均相同，經由

上述的方法切割後，資料不免需要於網路上傳遞，傳送的方式是藉由訊息傳遞，訊息內部的資料結構為時間、訊息種類及轉移動作編號等，分為傳送與接收二種時機討論。(詳細報告請參閱[6])

四、計畫成果自評

(a) 學習及研究如何整合物件導向的優點於 Foster 的分散式程式設計方法中。

Briot 在[7]將物件導向加入分散的方法分成三類：函式庫法(Library approach)、整合法(Integrative approach)、及反射法(Reflective approach)，而其中反射的方法使得程式語言、作業系統及資料庫之間的距離變模糊，使我們容易去發展及最佳化一個最小化的可動態擴充的運算系統(computing system)，但這並不能免除我們對於要有一個好的基礎設計及找到一個好的抽象化基礎集合的需求[8][9]。

(b) 在 CORBA 上實作在 STPN 中用來驗證、分析的三個重要演算法。

(1) 在分析解方面：在有效率的分散擴展狀態後，整合及收集狀態的複雜度，使得我們的結果並無實質的突破，此部分將進一步藉著 E-DAG 與 H-DAG 的關係來尋求可能的演算法。

(2)在可到達性分析方面：在成功的分工合作之後，如何測知此分散式演算法是否停止的方法也很快的找到，將進一步的整理投稿到學術會議或期刊。

(3)在模擬解分析方面：在整合分散的的模擬結果時，因有共同的模擬時間，使得求得結果的過程並無預期的複雜。

五、參考文獻

[1] Henri E. Bal, Jennifer G. Steiner, Andreew S. Tanenbaum, "Programming Languages for Distributed Computing Systems," *ACM Computing Surveys*, Vol. 21, No. 4, September 1989.

[2] Roger S. Chin, Samul T. Chanson, "Distributed Object-Based Programming systems," *ACM Computing Surveys*, Vol. 23, No. 1, March 1991.

[3] Diane Crawford (Eds.), "Concurrent Object-Oriented Programming," *Comm. ACM*, Vol. 36, No. 9, September 1993.

[4] 呂彥良,"在 CORBA 平台上設計與實作隨機過程時間派翠網路的分析解",私立淡江大學資訊工程研究所碩士論文,1999,6.

[5] 周秀玟,"在 CORBA 平台上設計與實作隨機過程派翠網路的可到達性問題",私立淡江大學資訊工程研究所碩士論文,1999,6.

[6] 閻海英,"在 CORBA 平台上設計與實作隨機過程派翠網路的模擬器",私立淡江大學資訊工程研究所碩士論文,1999,6.

[7] Jean-Pierre Briot, Rachid Guerraoui, Klaus-peter Lohr, "Concurrency and Distribution in Object-Oriented Programming," ACM Computing Survey, Vol. 30, No. 3, September 1998.

[8] Rachid Guerraoui, Mohamed E. Fayad," OO Distributed Programming Is Not Distributed OO Programming," Communications of the ACM, Vol. 42, No . 4, April 1999.

[9] Rachid Guerraoui, Mohamed E. Fayad," Object-Oriented Abstractions for Distributed Programming -- going beyond wrapping existing built-in toolkits with OO-like dressing," Communications of the ACM, Vol. 42, No . 8, August 1999.