導入工作流程於網格計算之研究

張儀興　　彭金隆　　蘇建郡
南台科技大學資管所副教授　南台科技大學資管所研究生　南台科技大學資管所副教授
yhchang@mail.stut.edu.tw　M9490226@webmail.stut.edu.tw　ccsu@mail.stut.edi.tw

摘要

網格系統可以整合各機構之間所擁有的資源，透過高速網路共享夥伴所擁有的計算、儲存和顯示資源情況，來讓計算更加快速完成。其中任務調度以及資源分配是極為重要的議題。因此在本文中利用工作流程可以自動化執行之優點，來將網格運算的任務調度以及資源使用做結合，提出一網格計算工作流程架構，並實際用以解決 Maximum Independent Set 此類屬於所需要的龐大計算資源之分配調度的 NP-Complete 問題。

關鍵詞：網格計算，工作流程，Maximum Independent Set，NP-Complete

1. 前言

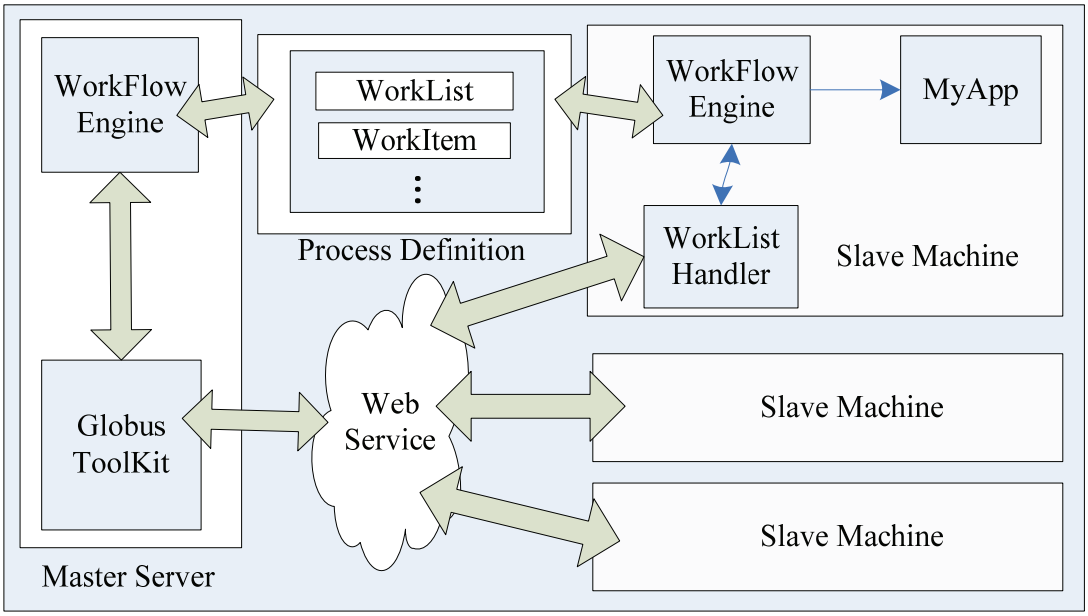
在電腦已經相當普及的這個年代，幾乎家家戶戶都有電腦，這代表著每部電腦都是一個可以運算的資源，再加上現在網路的發達，頻寬已經達到令人滿意的速度，這些分散的電腦資源在未使用時就如同浪費一般。網格的誕生[1]，是為了讓閒置的電腦運算資源能夠更加充分的利用，尤其是一些運算工作量相當龐大的計畫，例如有 LCG (LHC Computing Grid)[2，3]計畫係利用網格技術、歐洲粒子物理研究中心發展的全球網格。EGEE (Enabling Grid for E-sciencE)[4，5]主要是研究高能物理以及生命科學研究兩項為主，為了提供科學研究發展之需求，提出了許多整合的工具與環境，目前的成果已經初步成功，並且持續的將其他研究導入。Open Science Grid(OSG)[6]是由美國所主導的科學計算計畫，其將美國許多所大學的實驗室共同組成來營運，網格計算為其基礎運算建設，所包含的應用項目相當的多，有從高能物理,生物化學,天文物理,及天文學擴大成為高能物理,生物資訊,生物學,天文學,天文物理,重力波物理等。各計畫皆能展現網格運算有效利用網路的連結，將分散於各地的電腦運算資源以及儲存資源整合起來，成為一個龐大的虛擬超級電腦，而其中的運算任務調度以及網格資源利用是非常重要的一個技術，而資源是只在網格環境中對使用者存在利用價值的東西都可以稱作為資源，為了確保所有資源皆能讓使用者能夠調度使用，我們將會關心如何把資源有效、安全、方便的提供給使用者，像是電腦，高速網路設備、各式儀器、大容量的儲存硬體、各種科學資料、軟體、分散是檔案系統、資料庫等，都可算是網格資源。

工作流程管理系統是支援企業流程及資訊流程的科技，一個規劃完善的作業流程，透過工作流程系統的執行，可以達到自動化以及節省人力的目的。自動化特定的企業流程，根據程序規則，傳遞參與者之間的文件、資訊以及任務，始能順利完成特定任務[7]。而在網格環境中自動化工作流程執行系統，實現一個自動化工作流程引擎[8，9]，要在工作流程引擎於網格環境中該如何使用其資源以及資源的定義[10，11]便是一研究的重要課題，而目前[12]中是工作流程系統結合網格系統的研究。

在此我們應用工作流程系統的自動化特性，可以將運算任務交由工作流程系統來分配並且執行。由於我們將要嘗試解決的問題屬於 NP-Complete[13]問題，其屬於每次迴圈計算方式皆相同的特性，我們試著以暴力法來解決此類問題，但是若在單一主機上以暴力法來求解的話，將耗費相當長久的時間來運算，故我們嘗試將工作流程架構結合網格系統來解決此問題。在下段章節中，將會介紹了目前此類研究的相關工作；第三段中，將會有我們提出的網格工作流程架構；第四段會以說明前述之 NP-Complete 問題的案例分析，最後一個章節會有結論以及未來展望。

2. 文獻探討

從前在平行計算及分散式計算尚未普及之前，計算天文、物理、水文等等方面的計算工作時，就算擁有單一台效能強大的大型電腦主機恐怕也無法勝過由龐大數量組成的叢集系統。由於在網格環境下，我們所支配的計算資源，也就是可供作運算的電腦數量可以成長到非常龐大，而過去，我們作平行或分散式運



圖一 GWF(Grid Workflow)系統架構

算時最常使用到的環境就是叢集架構，為了能夠使用過去所以經建立好之叢集環境，將其納入網格環境之下，虛擬組織[14]的觀念尤然而生。叢集架構主要是針對『平行運算』而作最佳化，包含了資源分配以及運算效能負載平衡的設計，所以我們在這裡提出了使用網格工作流程系統，來達到虛擬組織化的目的，我們可以透過一台或數台 Grid Master 機器來管理虛擬組織。在虛擬組織之下，可以包含了數組叢集系統，當我們的工作流程引擎需要安排執行任務時，我們就可以利用虛擬組織的觀念，也就是每一組叢集系統，都假設成一部主機，而工作流程引擎，則只需要將計算流程交給此一虛擬主機，接下來就是等待叢集系統運算出結果之後，將結果回傳至上層的 Grid Master 主機，利用工作流程引擎其具有自動化執行任務的設計，只要將運算任務設計好，在透過良好的任務分配演算法[15]，即可讓運算更簡易，也更快速。

Globus Toolkit[16]為針對 Grid Computing 所設計之軟體，美國 Globus 項目提出的格網體系結構模型採用本地服務層、核心服務層、高層服務與工具層、應用層四層結構。在此基礎上，美國的 Argonne 國家實驗室、芝加哥大學、南加州大學和 IBM 公司共同提出了開放式格網服務體系結構(Open Grid Services Architecture，OGSA)[17]。OGSA 採用纖維層、聯絡層、資源層、協作層、應用層五層結構。它是目前國際上最具有影響力的網格計算軟體之一，幾乎可以說是網格中介軟體中等同於網格計算的軟體。它發起於 90 年代中期，最初的目的是希望把美國境內的各項高性能計算中心，通過高性能網路連結起來，以方便美國各大學和研究機構使用，提高高性能電腦的使用效率，並對資訊安全、資源管理、資訊服務、資料管理以及應用開發環境等網格計算的關鍵理論和技術進行了廣泛的研究，其目前已經推出至 Globus Toolkit 版本 4.0.3。使用上是完全免費，其遵循 Globus Toolkit Public License (GTPL)協議，任何人都可以從其網站上下載其原始碼回來進行研究並改進。

3. 網格計算的架構

高效能網格計算於工作流程架構 GWF(Grid Workflow)如圖一所示，分為 Master Server、Slave Machine、Process Definition 和 Web Service 四部份。此架構運作是當有任務的需求時，先行產生 WorkList 及 WorkItem(此部份包含在流程定義中)，接著啟動在 Master Server 中的工作流程引擎，將任務分派至各 Slave Machine 中完成。架構中各部份詳細說明如下：

3.1 Master Server

在 Master Server 中包含了 Workflow Engine 及 Globus Toolkit 二部份，說明如下：

3.1.1 Workflow Engine：

主要是依據 Globus Toolkit 運行時所公開之環境參數，像是 Web Service 服務狀態，系統效能，任務運作情況等。當一個任務執行時，其演算法如下。

```
Procedure MasterWorkflow
Begin
Step 1 : Set List(n); (Refer process definition)
Step 2 : Set Item(m); (Refer process definition)
Step 3 : If(List(n) >>List(m)) then
Step 4 :    Distribution List(n) use more grid
rce;
Step 5 :    WorklistAndItem(list,item);
End
```

圖二 Algorithm MasterWorkflow

Step 1 :運算任務開始執行，工作流程引擎依據事先規劃好的流程定義，產生出 WorkList。
Step 2 :運算任務開始執行，工作流程引擎依據事先規劃好的流程定義，產生出 WorkItem。
Step 3 and 4:Globus 會依照資源的使用情況分配執行任務的 Slave Machine。
Step 5 : 流程任務將會交由 Slave Machine 繼續處理後續。

3.1.2 Globus Toolkit：

Globus Toolkit 提共了許多功能，像是(GRAM， Globus Resource Relocation Manager)、 (GSI，Globus Secutiy Infrastructure)、GridFTP,(MDS，Monitoring and Discovery Service)、(GRIS，Grid Resource Information Service)、(GIIS，Grid Index Information Service)等等，我們可以直接調用其所擁有之各項資源及 API 來使用。

3.2 Slave Machine

在 Slave Machine 上設計 WorkList Handler、WorkFlow Engine、MyApp 來處理 WorkList 以及 WorkItem，其中說明如下：

3.2.1 WorkList Handler：

其目的是負責傳送或接收來自 Globus 所傳送之資訊，包含經由 Master Server 所定義好之 Worklist，以及額外的運算參數，還有當作是 Slave Machine 要求尋找 Globus 所提共之 Web Service 時的代理人。

3.2.2 WorkFlow Engine：

當 WorkList Handler 傳送 Worklist 過來之後，WorkFlow Engine 將會參考流程定義，來執行此流程內容包含的任務。

3.2.3 MyApp：

WorkFlow Engine 在取得任務內容之後，即會呼叫此由運算計畫相關人員所自行撰寫或現有之運算程式，像是 MPICH-G2 等。

Slave Machine 的演算法如下：

```
Procedure WorklistAndItem(list,item)
Begin
Step 1 : If(list empty) then
Step 2 :    Report finish;
Step 3 :    Set slave free;
         Else
Step 4 :    List←WorkflowEngine;
Step 5 : If(item not empty) then
Step 6 :    Execute All item;
Step 7 :    WorklistAndItem(list,item);
End
```

圖三 Algorithm WorklistAndItem

3.3 Process Definition

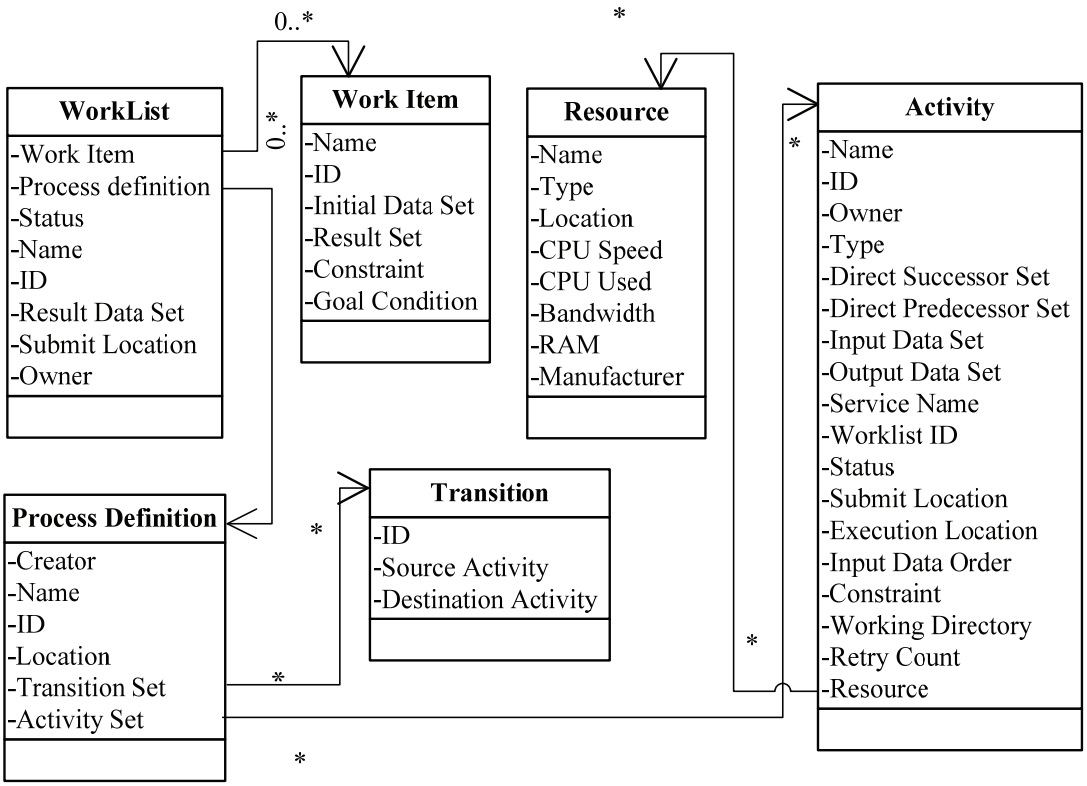
Process Definition 為一 XML 檔案，其最主要是將任務規劃為工作流程，我們參考了 Han Yu[18]定義了此工作流程引擎所需使用到之類別，定義如圖四之類別圖，類別圖包含了 Process Definition、WorkList、WorkItem、Transition、Resource、Activity 等六各類別，Process Definition 各欄位值及說明如表 1 所示。

表 1 Process Definition Description

	Description
Creator	建立者的大名。
Name	Process Definition 之名稱。
ID	Process Definition 之系統辨識 id。
Location	Process Definition 存放位置
Transition Set	所有 Transition 的集合
Activity Set	所有 Activity 的集合

3.3.1 WorkList：

在 Process Definition 中其內容為所有流程的父節點項目，內容包含有子流程規劃、流程狀態、任務所有人的資訊等，WorkList 類別各欄位值及說明如表 2 所示。



圖四 資源類別圖

表 2 WorkList Description

	Description
Status	定義此 WorkList 目前之執行情況，有已完成、重置、執行中、中斷等狀態。
Name	WorkList 之名稱。
ID	WorkList 之系統辨識 id，為所有流程中獨一無二不可重複。
Result Data Set	任務完成後之資料集合，定義該資料之類型及完成時間等資訊。
Submit Location	任務完成之資料集之收集處，以便作最後資料之整理。
Owner	此 WorkList 之製定者。

3.3.2 WorkItem：

是 WorkList 中之子項目，其目的為規劃子流程的流程細節，WorkItem 類別各欄位值及說明如表 3 所示。

表 3 WorkItem Description

	Description
Initial Data Set	初始化時需給定之資料之資料集合。
Name	WorkItem 之名稱。
ID	WorkItem 之系統辨識 id，為所有流程中獨一無二不可重複。
Result Set	當任務完成後之資料集，定義了哪些資料是此次任務所需要的。
Constraint	WorkItem 執行時可能會有限制條件才能執行，在此處定義。
Goal Condition	WorkItem 之完成條件。

表 4 Transition Description

	Description
ID	Transition 之系統辨識 id。
Source Activity	來源 Activity。
Destination Activity	Activity 之目的地。

3.3.3 Transition：

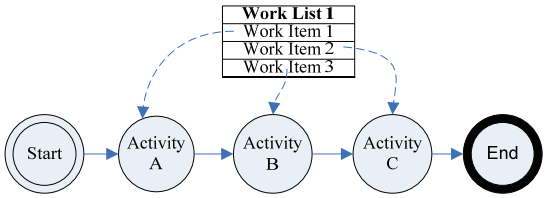
此為每個 Activity 的動向，可以為”Or”或”And”等動作，Transition 類別各欄位值及說明如表 4 所示。

3.3.4 Resource：

再此我們定義了所需要使用到的運算資源，包含有影響運算最顯著的幾個項目，例如：cpu 處理速度、記憶體大小、網路頻寬、系統架構、等，Resource 類別各欄位值及說明如表 5 所示。

表 5　Resource Description

	Description
Name	此資源之名稱。
Type	資源類型，可能是 32 位元系統或 64 位元系統等。
Location	資源之所在位置。
CPU Speed	此資源使用之 CPU 處理速度。
CPU Used	CPU 目前使用率。
Bandwidth	網路頻寬大小。
RAM	記憶體大小。
Manufacturer	使用之系統平台架構。



圖五 Worklist 與 Activity 關係圖

3.3.5 Activity：

工作流程引擎於自動化執行時即為執行此 Activity，每一個 Activity 可以視為一個節點，而此節點將包含運算任務中所需執行之程式，如圖五所示，WorkItem 1、WorkItem 2、WorkItem 3，在初始化之後，即可依據流程定義之每一個節點，Activity 類別各欄位值及說明如表 6 所示。

我們由 worklist handler 取得由 Master Server 所傳遞過來的工作流程，也就是 Worklist 和 WorkItem。由圖五可以看到，Worklist 包含了數個 WorkItem，而每一 WorkItem 就是一個 Activity，在 Slave Machine 的 WorkFlow Engine 執行時就是依照流程定義執行每一個 Activity。

為何我們要將所有流程拆開成 Worklist 和 WorkItem，是因為我們在執行網格運算時，網格資源隨時可能會有更動，例如其中一部 Slave Machine 離線或是有新的 Slave Machine 加入，我們可以在不更動主要流程的情況之下，將子流程分配給新加入的 Slave Machine，而已經離線的 Slave Machine 所執行的之未完成的任務，也將重新分配給其餘 Slave Machine 執行，以確保我們任務運算的完成率，以及運算資源能有效利用。

表 6 Activity Description

	Description
Name	Activity 之名稱。
ID	之系統辨識 id。
Owner	所有者。
Type	Activity 之類型為何種。
Direct Successor Set	直接繼承此 Activity 之集合。
Direct Predecessor Set	此 Activity 之前任繼承者集合。
Input Data Set	輸入資料集。
Output Data Set	輸出資料集。
Service Name	所需使用服務名稱，像是要呼叫之 Web Service 或系統內部的服務等。
Worklist ID	對應之 Worklist ID。
Status	目前狀態，可能有等待或中斷或執行中等情況。
Submit Location	Activity 執行完畢之結果輸出目的地。
Execution Location	執行此 Activity 之位址。
Input Data Order	輸入資料順序。
Constraint	執行此 Activity 所可能會有之條件限制。
Working Directory	Activity 之工作目錄。
Retry Count	若發生中斷或錯誤等意外狀況時之重試次數。
Resource	需使用之資源。

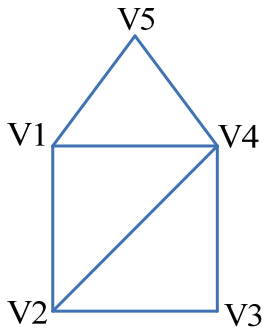
3.4 Web Service

所有訊息將透過 Web Service 來作資訊交換，包含了已經生成之 WorkList 或 WorkItem 等資訊。

我們定義了一 XML 範例列出上述各資源類別實例，如圖六所示。

```
<!--  WorkList  -->
<WorkList name="WorkList1">
 <id>10000001</id>
 <WorkItem name="Item1">
  <id>20000001</id>
  <InitialDadaSet><![CDATA[ {0,1,0,1,1},{1,0,1,1,1},{[0,1,0,1,0},{1,1,1,0,1},{1,0,0,1,0} ]]>
  </InitialDadaSet>
  <Constraint>0</Constraint>
  <ResultSet><![CDATA[ {0,1},{1} ]]></ResultSet>
  <GoalCondition>0</GoalCondition>
 </WorkItem >
 <status>RESET</status>
 <ResultDataSet><![CDATA[ {0,1},{1} ]]></ResultDataSet>
 <SubmitLocation>163.26.222.1</SubmitLocation>
</WorkList >
<!--  WorkItem  -->
<WorkItem name="Item1">
 <id>20000001</id>
 <InitialDadaSet><![CDATA[ {0,1,0,1,1},{1,0,1,1,1},{[0,1,0,1,0},{1,1,1,0,1},{1,0,0,1,0} ]]>
 </InitialDadaSet>
 <Constraint>0</Constraint>
 <ResultSet><![CDATA[ {0,1},{1} ]]></ResultSet>
 <GoalCondition>0</GoalCondition>
</WorkItem >
<!--  transition  -->
<transitionSet name="transition1">
 <id>50000001</id>
 <transition SourceActivity="BEGIN" DestinationActivity="DATA INPUT"/>
 <transition SourceActivity="DATA INPUT" DestinationActivity="FORK"/>
 <transition SourceActivity="FORK" DestinationActivity="A1"/>
 <transition SourceActivity="FORK" DestinationActivity="A2"/>
 <transition SourceActivity="FORK" DestinationActivity="A3"/>
 <transition SourceActivity="A1" DestinationActivity="And"/>
 <transition SourceActivity="A2" DestinationActivity="And"/>
 <transition SourceActivity="A3" DestinationActivity="And"/>
 <transition SourceActivity="And" DestinationActivity="End"/>
</transitionSet>
<!--  Activity  -->
<activitySet name="BEGIN">
 <id>70000001</id>
 <type>start</type>
 <owner>PGL</owner>
 <WorklistID>10000001</WorklistID>
 <ServiceName>Initial</ServiceName>
 <InputDataOrder>1</InputDataOrder>
 <Status>start</Status>
 <SubmitLocation>163.26.222.1</SubmitLocation>
 <ExecutionLocation>163.26.222.135</ExecutionLocation>
 <Constraint>0</Constraint>
 <WorkingDirectory><![CDATA[/home/grid/test1 ]]></WorkingDirectory>
 <RetryCount>0</RetryCount>
 <Resource name="Cluster1"> ... </Resource>
 <InputDataOrder></InputDataOrder>
</activitySet>
<!--  Resource  -->
<Resource name="Cluster1">
 <type>32bit</type>
 <Location>163.26.222.135</Location>
 <CPUSpeed>500Mhz</CPUSpeed>
 <CPUUsed>10%</CPUUsed>
 <Bandwidth>100Mbps</Bandwidth>
 <RAM>256MB</RAM>
 <Manufacturer>Debian Linux 3.1</Manufacturer>
</Resource>
```

圖六 各資源類別之 XML



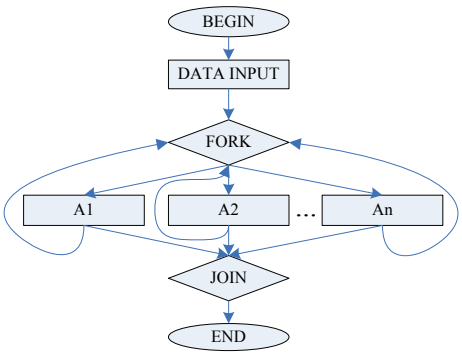
圖七 圖形 G

4. 案例研究

本研究中我們將求解最大獨立集合（Maximum Independent Set）此 NP-Complete 問題，並將我們所提出之（GWF）導入其中來作最佳的求解過程。在圖論中，獨立集是指圖的頂點集的一個子集，該子集的導出子圖不含邊。如果一個獨立集不是任何一個獨立集的子集，那麼稱這個獨立集是一個極大獨立集。若為如圖七此圖形，考慮圖七中的圖形 G≡(V,E)，V={V1,V2,V3,V4,V5}，E={(V1,V5),(V4,V5),(V1,V2),(V2,V4),(V2,V3),(V3,V4),(V1,V4)}，其中如，{V1}，{V1,V3}，{V4}皆為(Independent Set)；{V4}，{V1,V3}，{V2,V5} 則為極大獨立集；{V1,V3}，{V2,V5}，{V3,V5}則為最大獨立集(Maximum Independent Set)。與之相關的問題還有尋找圖的最大團問題，很容易可以證明這個問題與尋找最大獨立集的問題是等值的。還有一些如最佳拍元方案、最大匹配問題、所以解決圖的最大獨立集問題是具有鄉當的價值的。而在本研究中我們將使用暴力演算法，來利用網格系統強大的運算能力，並且配合工作流程系統，來求解某圖的最大獨立集。

實驗環境的配置，包含了一部 CPU 為 P4-2.8Ghz、記憶體為 768Mb、硬碟 40GB 將作為 Master Server 使用;兩部 PIII-500Mhz，記憶體 256Mb，硬碟 10GB 的主機作為 Slave Machine。Workflow Engine 將使用 Java 來撰寫，搭配 Globus Toolkit 4.0.3 來完成此實驗，所有主機之作業系統皆為 Linux。

圖八為一網格工作流程執行範例流程圖。在我們將求解最大獨立集所需之各項初始化參數接預備完成之後，於 Master Server 端之工作流程引擎將會依照我們自行定義之 Slave Machine 運算能力或是其他各項參數等，並參考預先定義好之流程定義來將任務分配至各 Slave Machine，而最終結果將在每一 Slave Machine 執行完流程之後，會回報至 Master Server 中作彙整整理，最後即可得到我們所需之最大獨立集。



圖八 網格工作流程執行範例流程圖

5. 結論

網格計算目前正處於熱烈發展的階段，而資源調度以及任務排程於網格中正是格外受到重視的研究議題，而網格工作流程更是還處於初步之研究階段，因此本論文中我們提出了結合工作流程引擎以及網格系統，來針對 NP-Complete 問題的運算提出了解決方案，利用工作流程具備預先流程定義以及自動化的特性，相信能夠有效的解決 NP-Complete 問題的運算需求，而在未來我們將完成此系統的架構，並套入 NP-Complete 的範例來做進一步實現。

參考文獻

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, 1998.
[2] A. Pfeiffer, "*Overview of the LCG applications area software projects*," 2004.
[3] "LCG (LHC Computing Grid),http://lcg.web.cern.ch/LCG/."
[4] "EGEE (Enabling Grid for E-sciencE),http://public.eu-egee.org/."
[5] F. Gagliardi and M. E. Begin, "*EGEE - providing a production quality grid for e-science*," 2005.
[6] "Open Science Grid(OSG),http://www.opensciencegrid.org/."
[7] C. M. Dan, *Internet-Based Workflow Management: Towards a Semantic Web*: John Wiley \& Sons, Inc., 2002.
[8] T. Heinis, C. Pautasso, and G. Alonso, "*Design and Evaluation of an Autonomic Workflow Engine*," 2005.
[9] J. Nichols, H. Demirkan, and M. Goul, "*Autonomic workflow execution in the grid,*"

Systems, Man and Cybernetics, Part C, IEEE Transactions on, vol. 36, pp. 353-364, 2006.
[10] T. Gomes Ramos and A. C. Magalhaes Alves de Melo, "*An Extensible Resource Discovery Mechanism for Grid Computing Environments*," 2006.
[11] M. Marzolla, M. Mordacchini, and S. Orlando, "*Resource discovery in a dynamic grid environment*," 2005.
[12] C. Junwei, S. A. Jarvis, S. Saini, and G. R. Nudd, "*GridFlow: workflow management for grid computing*," 2003.
[13] R. G. Michael and S. J. David, *A Guide to the Theory of NP-Completeness*: W. H. Freeman \& Co., 1990.
[14] I. Foster, C. Kesselman, and S. Tuecke, "*The Anatomy of the Grid - Enabling Scalable Virtual Organizations*," 2001.
[15] Z. Shaohua, G. Ning, and L. Saihan, "*Grid workflow based on dynamic modeling and scheduling*," 2004.
[16] I. Foster and C. Kesselman, "*The Globus project: a status report*," 1998.
[17] "OGSA(Open Grid Architecture),http://forge.gridforum.org/sf/projects/ogsa-wg."