# 行政院國家科學委員會專題研究計畫 期中進度報告

**於協同配置資料網格環境中具適應性複本管理的高效能醫療影像儲傳系統之實作(2/3)**

**研究成果報告(精簡版)**

|  |  |
|---|---|
| 計 畫 類 別 | ：個別型 |
| 計 畫 編 號 | ：NSC 98-2622-E-029-001-CC2 |
| 執 行 期 間 | ：98 年 08 月 01 日至 99 年 07 月 31 日 |
| 執 行 單 位 | ：東海大學資訊工程與科學系 |

計 畫 主 持 人：楊朝棟
共 同 主 持 人：楊晴雯

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 99 年 04 月 30 日

附件二

# 國科會專題研究計畫精簡報告

學門領域：資訊學門一 平行與分散處理

計畫名稱：於協同配置資料網格環境中具適應性複本管理的高效能醫療影像儲傳系統之實作(2/3)

計畫編號：NSC 98-2622-E-029-001-CC2

執行期間：自民國 98 年 08 月 01 日起至民國 99 年 07 月 31 日

執行單位：東海大學 資訊工程學系

主 持 人：楊朝棟

參與學生：羅裕翔　陳龍騰　王冠傑

## 合作企業簡介

合作企業名稱：亞盾科技股份有限公司
計畫聯絡人：潘介棟
資本額：新台幣 500 萬元
產品簡介：

亞盾科技股份有限公司成立於民國 89 年，成立之初公司的營業方向，是以販售個人電腦軟體、硬體及其周邊設備、耗材為主。隨著資訊科技的日新月異，產品研發技術不斷的創新下，亞盾科技公司逐漸轉型以研發「叢集計算系統」及「醫療自動化監控系統」為主，以販售個人電腦軟體、硬體及其周邊設備、耗材為輔，以期能服務更多的消費族群，滿足消費者多元化的需求。亞盾科技股份有限公司業務內容如下：

- Cluster 叢集系統之研發與應用
- 高效能護理站自動化系統(如附件)（自行研發）
- 區域網路及廣域網路規劃架設及維護
- 遠距教學架構及規劃
- 伺服器架設與設定

網址：www.ardness.com.tw 電話：(04) 23588880

## 研究摘要(500 字以內)：

影像診斷學中，醫療影像儲傳系統(Picture Archiving and Communication System, PACS)是一種專門用來儲存、取得、傳送與展示醫療影像的電腦或網路系統。PACS 的主要目的在於將醫療系統中所有影像，以數位化的方式儲存，並經由網路傳輸至系統中，供使用者遠端電腦螢幕閱讀影像並判讀。同時也可作為不同影像傳遞交換的工具，隨著軟體及運用程式的進步，將來更可進一步協助醫師進行診斷、教學及醫學研究。而成功的 PACS 不只須要強大硬體，更依賴完善軟體功能及作業程序。透過日漸成熟的網格計算(Grid Computing)技術，將散佈各地之虛擬組織的資源可以透過網格的概念來調派和集中。再者，利用資料網格(Data Grid)的容錯特性與高可用性，因此可以滿足各種在醫療資訊應用方面

的計算與檔案儲存需求。

為降低 PACS 系統之擴建成本與設置第二 PACS 系統為考量，本計畫為期 3 年，主要是研發 Smart Broker Centric 與具有自適性副本管理元件於於協同配置(Co-allocation)資料網格環境中。藉由導入 Open Source PACS 解決方案，建置於特別設計的網格功能元件之上，期能證實以網格技術來支援 PACS 系統的可行性，除保有 PACS 的優勢、實際提昇 PACS 影像副本交換效率與一個成本較低廉的 PACS 導入方案。

於上年度，計畫完成平台架構上所有系統元件的雛型設計開發，平台架構區分 Application、Smart Broker、Cyber Abstraction、Grid Middleware 以及 Fabric 等 5 個階層，其中較高層以使用者為中心，並以 Smart Broker 作為本架構核心；底層專注於資源整合，並以 Cyber Abstraction 描述高低二階層之間如何連結。另外設計一個網格知識管理(GKM)模型以支援各元件計算的參數資料輸入。其成果已在 APSCC08, IPCADS08 與 JNCA 發表。

本年度，計畫完成整合各階層元件，使用 Globus Tookit 與雲端計算(Cloud Computing)結合 Cross-CA 技術，建構跨網格系統的交互驗證機制，提昇計算與儲存資源利用率。實作 Co-Allocator 元件，該元件能經由 Resource Management System (RMS)提供對應資源，並透過 GKM 的知識基礎，設計出具有認識網格環境現況的能力，即為自適性。在資料網格環境中，資料集被重複製為複本且分送到多重的站台。由於資料集的檔案通常很大，如何有效率的存取及傳輸成為重大的課題。因此先前有學者發展出協同配置的架構，使得同時從多重站台平行下載資料變成可能，目前發展出數種協同配置法用來解決傳輸時本地端與伺服端網路傳輸率不斷變動的問題。本研究中，我們採用 TCP 頻寬估計模型與突發模式等新策略，藉此強化預測性遞迴調整的協同配置法，進一步提高大量資料集於資料網格中的傳輸效能。我們的方法能有效地找出一群快速伺服器並分配較多的工作量提高其資源利用率，動態計算出檔案切割量，有效減少各伺服器間的相互等待時間。藉由各項實驗證明其傳輸的高效能，並具有網路自適應性與高度容錯性，有效因應不同的網格環境。使用 Open Source PACS 系統為應用實例，進行實驗及測試雛型系統功能及相容性。其成果已在 Journal of Supercomputing 與 CCPE 發表。

接下來最後一年，我們根據應用實例的實驗結果，進一步改善各階層元件與雛型應用系統與 GKM 學習樣本調較。並就我們的 PACS 系統實際使用情況與真實醫院 PACS 系統作整體效能比較。

**關鍵詞：**網格計算、資料網格、醫療格網、醫療影像儲傳系統、協同配置

**English Abstract**

PACS (Picture Archiving and Communication System) is a system for archiving, retrieving, communicating and displaying medical images. The purpose of PACS is to acquire medical images from medical systems, store them in digital formats, and transmit them to remote users through networks for diagnostic usages. Furthermore, PACS can be sharing platforms for various images. As the development of software and computing technologies, PACS is promising to assist doctors in medical diagnoses, instruction and researches. The success of PACS depends on not only powerful hardware, but also advanced software utilities and operating procedures. By means of the developed grid computing technologies, resources of virtual organizations located in different places can be managed and dispatched. Moreover, the salient features of fault-tolerance and high availability of data grid can satisfy various kinds computing and storage requirements in medical applications. We propose a three-year project to design and implement Smart Broker Centric and adaptive replica management components in co-allocation data grid environments. By means of introducing Open Source PACS solutions based on specially designed grid modules, we plan to verify the feasibility of using grid technologies to support PACS. In addition, the contributions will include promoting the advantages of PACS, improving the sharing performance of PACS image replica and a cost-effective PACS solution.

In the first year, we design and implement the prototype, including all components of the platforms. The architecture is composed of five layers: Application, Smart Broker, Cyber Abstraction, Grid Middleware and Fabric. The higher layers are user-centric, and use Smart Broker as the core of the architecture. The bottom layers focus on resource integration, and use Cyber Abstraction to describe the interconnection of the top layers and the bottom layers. Also, a grid knowledge management (GKM) model is designed to facilitate the parameter input of all components. Some research results are published in APSCC08, ICPADS08 and JNCA.

In the second year, we plan to integrate components of all layers by using Globus Toolkit, Cloud Computing and Cross-CA technologies, in order to improve the computing performance and resource utilization. The implemented Co-Allocator component can provide corresponding resources by Resource Management System (RMS), and support grid context-awareness by GKM. Data grid consists of scattered computing and storage resources located in different regions yet accessible to users. Co-allocation architectures can be used to enable parallel transfers of data file from multiple replicas in data grids which are stored at different grid sites. Schemes based on co-allocation models have been proposed and used to exploit the different transfer rates among various client-server network links and to adapt to dynamic rate fluctuations by dividing data into fragments. In this report, the TCP Bandwidth Estimation Model (TCPBEM) is used to evaluate dynamic link states by detecting TCP throughputs and packet lost rates between grid nodes. We integrated the model into ARAM, calling the result the anticipative recursively adjusting mechanism plus (ARAM+); it can be more reliable and reasonable than its predecessor. We also designed a Burst Mode (BM) that increases ARAM+ transfer rates. This approach not only adapts to the worst network links, but also speeds up overall performance. Taking Open Source PACS for examples, we plan to conduct experiments on functionality and compatibility of the prototype.

In the last year, we will plan to refine the components and the application according to the experimental results. Also, the proposed system will be compared with a real-world PACS of hospitals in terms of overall performance.

**Keywords: Grid Computing, Data Grid, Medical Grid, PACS, Co-allocation**

## 人才培育成果說明：

研究初期工作為叢集計算系統的規劃與設計，參與人員能在叢集系統的架設及應用熟悉其技巧。對於叢集伺服器管理能更有經驗。參與研究的人員能深切瞭解 PACS 與格網的功能與意義，應用軟體工程的理論到實際的開發過程。對於未來投入軟體產業有非常大的幫助。

對於參與本子計畫研究之人員將能對，健康服務格網，醫療資料格網，及 HL7 資料管理這些重要的技術有更深刻的認識，包含其歷史背景、發展過程、時空環境、遭遇到的問題與未來發展前景都能有通盤的學習與了解。

對於參與本子計畫研究之人員將能對服務格網資源的擷取有更多的練習，包括服務格網資訊監控系統技術應用、RRDTool 技術應用、JRobin 圖表繪製，同時為了將資源提供給其他子計畫之模組來使用，如何把資料整合並有效利用也將會是學習的重點。包含其歷史背景、系統發展過程、系統整合測試與技術應用、遭遇到的問題與未來發展前景都能有通盤的學習與了解。

參與人員可以學習到結合不同理論建構問題以及獨立思考能力，在實驗的過程，參與者可學習到判斷研究成果的正確性以及回饋修正的能力。另外，也可以學習整套研究的方法，團隊工作的精神，以及撰寫科技論文的經驗。也可以建立觀摩網站供有志從事高效能醫療影像儲傳系統工作的朋友學習，規劃未來高效能醫療影像儲傳系統計畫的方向。

由於本計畫為整合資料網格環境與醫療影像儲傳系統，對於參與本計畫研究之人員，更能訓練橫向整合溝通的能力，與不同團隊之間的協調合作，是一個非常好的訓練。對於參與研究的學生，不論是作業系統、分散式計算、平行分散式資料庫、PACS、或格網計算都能有更深一層瞭解。另外，諸如網路與系統安全、網路與系統管理、邏輯分析能力，如數據的分析技巧及格網計算領域技術問題的處理能力都可以獲得一連貫的訓練。而對碩士班學生而言，除了核心理論的開發以外，更可以學習到系統整合與系統分析的經驗。

## 技術研發成果說明：

本計畫將依照研究內容與目標畫分三年進行。第一年計畫完成平台架構上所有系統元件的雛型設計開發，平台架構區分 Application、Smart Broker、Cyber Abstraction、Grid Middleware 以及 Fabric 等 5 個階層，其中較高層以使用者為中心，並以 Smart Broker 作為本架構核心；底層專注於資源整合，並以 Cyber Abstraction 描述高低二階層之間如何連結。另外設計一個網格知識管理(GKM)模型以支援各元件計算的參數資料輸入，例如嵌入隱藏式馬可夫模型(Hidden Markov Model, HMM)與決策樹(Diction Tree)的 Know-How、Grid 知識模型(GKM)及累積的 System Log(如副本運作及配置情形)，以設計學習型知識資料庫。

第二年，計畫整合各階層元件，使用 Globus Tookit 與 Cloud Computing 結合 Cross-CA 技術，建構跨網格系統的交互驗證機制，提昇計算與儲存資源利用率。實作 Co-Allocator 元件，該元件能經由 Resource Management System(RMS)提供對應資源，並透過 GKM 的知識基礎，設計出具有認識網格環境現況的能力，即為自適性。在資料網格環境中，資料集被重複製為複本且分送到多重的站台。由於資料集的檔案通常很大，如何有效率的存取及傳輸成為重大的課題。因此先前有學者發展出協同配置的架構，使得同時從多重站台平行下載資料變成可能，目前發展出數種協同配置法用來解決傳輸時本地端與伺服端網路傳輸率不斷變動的問題。本研究中，我們採用 TCP 頻寬估計模型與突發模式等新策略，藉此強化預測性遞迴調整的協同配置法，進一步提高大量資料集於資料網格中的傳輸效能。我們的方法能有效地找出一群快速伺服器並分配較多的工作量提高其資源利用率，動態計算出檔案切割量，有效減少各伺服器間的相互等待時間。藉由各項實驗證明其傳輸的高效能，並具有網路自適應性與高度容錯性，有效因應不同的網格環境。為了量測效率，另外建立客戶端應用程式以監控管理 Workflow，並且能將工作歷程與結果回饋給 GKM。而本計畫「具適應性」的各演算法，即可依據此學習型知識資料庫不斷累積 Grid 領域知識並

作適當的調整，即可達成最適性的目標。導入使用 Open Source PACS 系統為應用實例，進行實驗及測試雛型系統功能及相容性。

第三年，根據應用實例的實驗結果，進一步改善各階層元件與雛型應用系統。並就我們的 PACS 系統實際使用情況與真實醫院 PACS 系統作整體效能比較。設計規劃測試案例與系統目標，藉由與醫院實際數據作交差比對，以證實系統有效性與可用性。

此外，本計畫預計提供進階使用者的客製化應用程式，作為資訊整合、管理共享資源之介面，以滿足進階使用者對平台服務的特殊需求。例如放射科醫生希望 PACS 系統能提供更快速、安全的資料下載、醫療影像的快速搜尋等。系統平台透過整合的應用程式介面取得不同來源的影像和資料，滿足使用者需求。此客製化應用程式內建高速多點平行傳輸技術，提供使用者多點高速平行傳輸功能，例如查詢遠端大型影像檔案，可充份運用此技術加速下載，以減少使用者等待時間。在整個三年的工作安排規劃內容，較特別的即是將 PACS 整合於已運作許久的大型網格系統之上及引入雲端計算相關技術工具，其詳細將分述如后：

由於不同網格系統即代表有不同的驗証領域，其二者之間是無法直接跨越，必需透過 Cross-CA 技術結合 TigerGrid 及 UniGrid 資源，在得到近 3 百顆 CPU 計算能力及超過百部伺服主機的硬碟空間之後，以採用自由軟體的 PACS 系統架構於其上，測試其相容性與效能並藉以調整 Smart Broker 之 Grid KM 及 Workflow Engine 效能。

## 技術特點說明：

本計畫預計開發展一套可以運行在現行醫療院所環境中的以協同配置資料網格環境中具適應性複本管理的高效能醫療影像儲傳系統，並應用在科學、教育、商業等領域。在核心技術部份，我們應用各種 Grid Computing and Cloud Computing 技術開發具輕量化，高速傳輸，與容錯的格網系統。在軟體開發上，與開放原始碼社群整合，預期提供使用者一個便利、安全的高效能醫療影像儲傳系統平台，提供管理者一個集中式的管理介面，與提供開發者一個具高度擴充性及相容性的系統架構，使得未來管理的人力及時間成本大幅降低。執行本計畫所得到的研究理論、工具開發、與實務經驗亦可作為相關領域學術研究與教學的素材。

學術研究上的預期貢獻：本項研究以創新的方法與高等軟體開發技術解決複雜且難以設計的格網中介軟體。結合理論與應用，建構高效率、容易操作、管理、與維護的高效能醫療影像儲傳系統。本項研究所提出方法與技術，預期對研究或發展相似的高效能醫療影像儲傳系統，有一定的參考的價值。本計畫的研究成果，預期將在國際著名期刊與國際研討會上發表。另外，透過國內研討會的交流，可以結合國內從事此方面研究的教授研發能量。未來可以發展具有國際競爭力的系統。

所開發出來的系統教育上的預期貢獻，對於有興趣學習分散式計算、Data Grid 與 Grid Service 計算、叢集與格網計算的資訊相關科系的高年級學生與研究生，可以作為練習的平台；同時對於推廣高效能運算教育具有很大的助益。對教師而言，也可以有實際而且容易操作的高效能醫療影像儲傳系統作為課堂 Demo 的素材。

應用上的預期貢獻：本計劃研發新一代的醫療影像儲傳中介軟體系統，除了能夠透過 Data Grid 技術來達到資料的相容性與容錯度，高彈性的大量管理及部署，也能夠以更好的系統效能，輕量化的系統結構實現中介資料索引搜尋，虛擬空間等功能。研發成功後預期的影響，為提供產業、學術以及大眾網路一個可行的商業化高效能醫療影像儲傳系統平台，並實際導入學術及醫療院所應用。

本計畫若能承蒙貴會支持，透過計畫的功能規畫與研發，建立適合醫療院所之協同配置資料網格環境中具適應性複本管理的高效能醫療影像儲傳系統平台、相關服務元件及計算環境、PACS 裝置技術，有助於未來我國面臨高齡化人口所需的健康照護基礎環境建設。同時透過自由軟體技術及平台對國內相關產業能提供相關技術及支援，對於提昇我國資訊產業在健康照護方面的競爭力，將有相當助益。

**可利用之產業及可開發之產品：**

**推廣及運用的價值：如增加產值、增加附加價值或營利、增加投資/設廠、增加就業人數………等。**

2008 年國際情勢的變化，美國的二次房貸金融等危機造成產業蕭條，造成 1929 年以來全世界經濟大蕭條與通縮的危機。有鑑於此；迎合下一波經濟回升與產品競爭力的考量之下，勢必在研發設計的創新上累積具競爭力之產品，使企業在逆境中化為轉機的具體作法與籌碼。居家環境是每個人最熟悉的空間，也是停留時間最長的場所，建構於居家環境下的健康監測系統能提供長期、持續性的健康監測資料。「遠距居家照護(Tele-homecare)」研究便是希望結合資訊與通訊科技，使能在被照護者的家中便利、有效地提供個人健康管理與保健服務，並具有價格效能優勢及可延伸性。亞盾科技的目標是提供最先進的技術，期盼能強化國家整體發展，不論在學術研究、軍事國防、產業升級，並協助客戶快速提昇國際競爭力。

遏止全經濟蕭條與產品競爭力的頹勢能逆勢成長；本計畫提出「遠距居家健康照護監測系統之設計開發」的設計開發規劃，廠商在現有技術資源與能力上期能透過專家學者的診斷以解決問題：使用者在家中量測體溫、血壓等生理訊號，經過電話線、有線電視、或網際網路傳送到「居家照護服務提供者(home healthcare provider)」的資料庫儲存，並作進一步的資料管理與分析。如發現生理訊號有異常，居家照護服務提供者可轉介使用者至醫療單位作進一步診治，醫療單位也可讀取居家照護服務提供者的長期監測資料作為診斷參考。

悉知東海大學楊朝棟教授於高性能計算與網格計算之研究，在學界為其中之翹楚，為使得博盛數碼動力公司得以提供客戶在高效能計算與網格計算領域中巨量運算的解決方案，並滿足周邊相關使用與管理者的需求下，與楊教授合作，期盼能在教授的協助下，將技術轉移到產業界，並實際應用在各種不同的高效能叢集計算與網格計算領域中。

本計畫之合作企業之合作目的不僅是在培養人才，更重要的是 PACS 與 Grid 相關研究技術能量的累積。該合作企業已經與本系執行有過五年小產學之經驗，已培養出多位具 Cluster Computing Systems 與 Grid Computing Environments 相關經驗的人才。期望由此次開發型小產學之經驗，能進入醫療影像儲存系統技術領域，後續將朝醫療網格與居家服務網格相關產業技術發展。

每年配合款應達當年度計畫總經費 30％以上。合作廠商派二位研發人員參與本計畫。企業得與計畫執行機構協商繳交先期技轉金，額度不得低於計畫總經費之 15％(7 年授權)。產學計畫結束後 3 個月內，計畫執行機構應向本會及企業繳交精簡報告及完整結案報告電子檔。

## 計畫成果自評部份

**在學術期刊發表或申請專利**
**專利(申請中)**
**台灣申請案：**
「進階預測遞迴式調整協同配置法」，申請案號：098105346，申請日期為 2009/2/19。

**美國申請案：**
申請日：September 9, 2009 (主張台灣優先權申請日)
申請案號：12/556,413
專利名稱：ANTICIPATIVE RECURSIVELY-ADJUSTING CO-ALLOCATION MECHANISM

**Journal Papers**

[1] **Chao-Tung Yang***, I-Hsien Yang, and Chun-Hsiang Chen, "RACAM: Design and Implementation of a Recursively-Adjusting Co-Allocation Method with Efficient Replica Selection in Data Grids," *Concurrency and Computation: Practice and Experience*, 2010. (ISSN: 1532-0626, SCI JCR IF=**1.791**, EI)

[2] **Chao-Tung Yang***, Yao-Chun Chi, Ming-Feng Yang, and Ching-Hsieh Hsu, "An Anticipative Recursively-Adjusting Mechanism for Parallel File Transfer in Data Grids," *Concurrency and Computation: Practice and Experience*, 2010. (ISSN: 1532-0626, SCI JCR IF=**1.791**, EI)

[3] **Chao-Tung Yang***, Shih-Yu Wang, and William C. Chu, "A Dynamic Adjustment Strategy for Parallel File Transfer in Co-Allocation Data Grids," *Journal of Supercomputing*, Springer Netherlands, 2010. (ISSN: 1573-0484, SCI JCR IF=**0.615**, EI)

[4] **Chao-Tung Yang***, Chun-Pin Fu, and Ching-Hsien Hsu, "File Replication Maintenance and Consistency Management Services in Data Grids," *Journal of Supercomputing*, Springer Netherlands, 2010. (ISSN: 1573-0484, SCI JCR IF=**0.615**, EI)

[5] **Chao-Tung Yang***, Chih-Hao Lin, Ming-Feng Yang, and Wen-Chung Chiang, "A Heuristic QoS Measurement with Domain-based Network Information Model for Grid Computing Environments", *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, Volume 5, Number, 4, pp. 235-241, 2010. (ISSN Online: 1743-8233 - ISSN Print: 1743-8225, SCI JCR IF=0.66, EI)

※ 備註：精簡報告係可供國科會立即公開之資料，並以四至十頁為原則，如有圖片或照片請以附加檔案上傳，若涉及智財權、技術移轉案及專利申請而需保密之資料，請勿揭露。

# A heuristic QoS measurement with domain-based network information model for grid computing environments

Chao-Tung Yang*, Chih-Hao Lin
and Ming-Feng Yang

Department of Computer Science,
Tunghai University,
Taichung 40704, Taiwan, ROC
E-mail: ctyang@thu.edu.tw
E-mail: ljerome86@gmail.com E-mail: orsonyang@gmail.com
*Corresponding author

Wen-Chung Chiang

Department of Information and Networking Technology,
Hsiuping Institute of Technology,
Taichung 41280, Taiwan, ROC
E-mail: wcchiang@mail.hit.edu.tw

**Abstract:** Recently, Grid computing is more and more widespread. Therefore, there exists a common issue, i.e., how to manage and monitor numerous resources of grid computing environments. Mostly, we use Ganglia and Network Weather Service (NWS) to monitor machines' status and network-related information, respectively. But, information provided by Ganglia and NWS is not sufficient in some scenarios owing to varied user requirements. Therefore, we propose a heuristic Quality of Service (QoS) measurement constructed with domain-based information model that provides more effective information to meet user requirements. Furthermore, we expect that users could manage and monitor numerous resources of grid environments more effectively and efficiently.

**Keywords:** grid computing; heuristic; QoS; quality of service; network information model.

**Reference** to this paper should be made as follows: Yang, C-T., Lin, C-H., Yang, M-F. and Chiang, W-C. (2010) 'A heuristic QoS measurement with domain-based network information model for grid computing environments', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 5, No. 4, pp.235–243.

**Biographical notes:** Chao-Tung Yang received his BS in Computer Science from Tunghai University, Taichung, Taiwan, in 1990, and the MS in Computer Science from National Chiao Tung University, Hsinchu, Taiwan, in 1992. He received the PhD in Computer Science from National Chiao Tung University in July 1996. He is a Professor of Computer Science at Tunghai University in Taichung, Taiwan. He got the excellent research award by Tunghai University in 2007. In 2007 and 2008, he got the Golden Penguin Award by Industrial Development Bureau, Ministry of Economic Affairs, Taiwan. His present research interests are in grid and cluster computing, parallel and multi-core computing, and web-based applications.

Chih-Hao Lin received his BS in Computer Science from Feng Chia University in 1999, and his MS in Computer Science from the Tunghai University, Taiwan, in July 2009. His research interests include grid computing, cloud computing, and parallel computing.

Ming-Feng Yang received his BS in Hsiuping Institute of Technology in 2004. He received his MS in Department of Computer Science from Tunghai University in July 2009. He is a software engineer at Hsiuping Institute of Technology. His research interests include grid computing, cloud computing, and parallel computing.

Wen-Chung Chiang received his BS from the Department of Applied Mathematics in 1991, and his PhD from the Department of Applied Mathematics of the National Chung-Hsing University in 2002. He is an Assistant Professor of Department of Information and Networking Technology of Hsiuping Institute of Technology. His current research interests include grid computing, medical image processing and communication networks.

## 1 Introduction

As we known, Grid computing is increasingly used by organisations to achieve high-performance computing and heterogeneous resources sharing. All tasks executed in grid environments will be influenced by network status owing to complicated and numerous communications between computing resources (Krauter et al., 2002; Krefting et al., 2008). While we design algorithms for specific usages or assign tasks into grid environments, we will evaluate the influence of network-bandwidth-related information and then adjust algorithms to match up real-time state of network. The best-case scenario is that our grid environments have some mechanism to provide network state automatically. And then our applications or web service agents could achieve higher performance owing to dynamic parameters adjustment and algorithms optimisation.

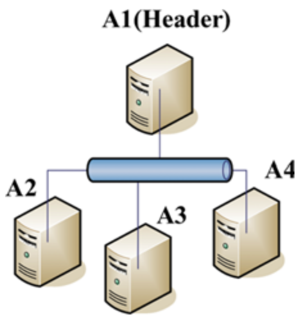
While grid computing becomes widespread gradually, it brings about a common issue, i.e., how to manage and monitor numerous resources of grid computing environments. Mostly, we use Ganglia and NWS to monitor machines' status and network-related information, respectively. But, information provided by Ganglia and NWS is not sufficient in some scenarios owing to varied user requirements.

According to the mechanism that we designed earlier, we could retrieve both real-time and historical network information in real-time manner; even advanced customisation for special purpose is available. With the customised shell scripts that we wrote for NWS service, we could easily and quickly deploy NWS service to grid nodes and fetch network-related information automatically and regularly. And with the database we built, we could obtain both historical information and some statistics from our grid computing environments. Statistics is indeed helpful in many fields, for example, job dispatching or replicas selection.

Maybe the mechanism that we designed earlier is helpful in most conditions, but it will not work efficiently if grid environment changed frequently. We found some inconveniences resulting from the technique that NWS adopted make this mechanism inefficient. The service provided by NWS will be affected if grid environment changed and we have to re-deploy NWS service manually and frequently. For network management, 'manual' is equivalent to 'inefficiency'.

A typical example is illustrated in Figure 1. If we have registered an NWS clique into for grid nodes A1, A2, A3 and A4 and the header is node A1, i.e., node A1 has stored network measurements between these nodes. While hardware failure occurs to node A1, or node A1 has just forced to reboot owing to software updating operations, the NWS clique fails, too. Network administrators have to restart clique manually again and again. Besides, we would not be notified if any nodes fail by default. Therefore, we lead in a Network Management System (NMS) using Simple Network Management Protocol (SNMP) technique to co-work with NWS service to resolve this issue.

**Figure 1** A typical NWS clique deployment in grid nodes (see online version for colours)



In this paper, we propose a heuristic QoS measurement constructed with domain-based information model that provides more effective information to meet user requirements. Furthermore, we hope that users could manage and monitor numerous resources of grid environments more effectively and efficiently.

## 2 Backgrounds

### 2.1 Machine information provider

The Ganglia (http://ganglia.info/) is an open-source project grew out of the University of California, Berkeley's Millennium initiative. This project was a scalable distributed system for monitoring status of nodes (processor collections) in wide-area systems based on Grid or clusters. It adopts a hierarchical, tree-like communication structure among its components to accommodate information from large arbitrary collections of multiple Grid or clusters. The information collected by the Ganglia monitor includes hardware and system information, such as processor type, CPU load, memory usage, disk usage, operating system information, and other static/dynamic scheduler-specific details. It also provides a web portal for users to observe all machines via web interface. Our grid environments are currently overseen by the Ganglia (as shown in Figures 2 and 3).

**Figure 2** Multi-grid resource broker with Ganglia web portal (see online version for colours)

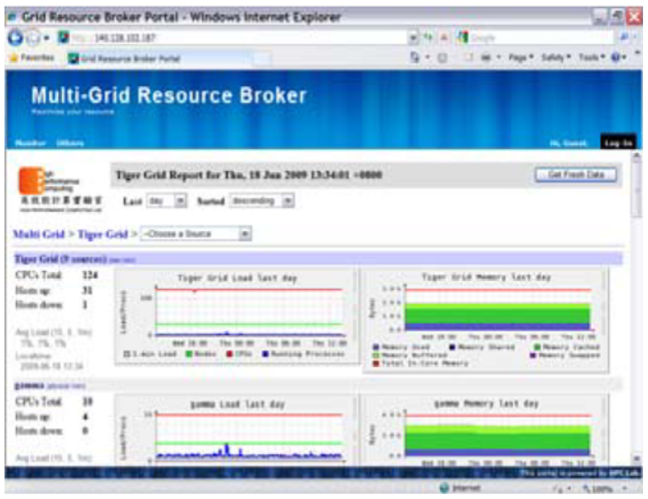
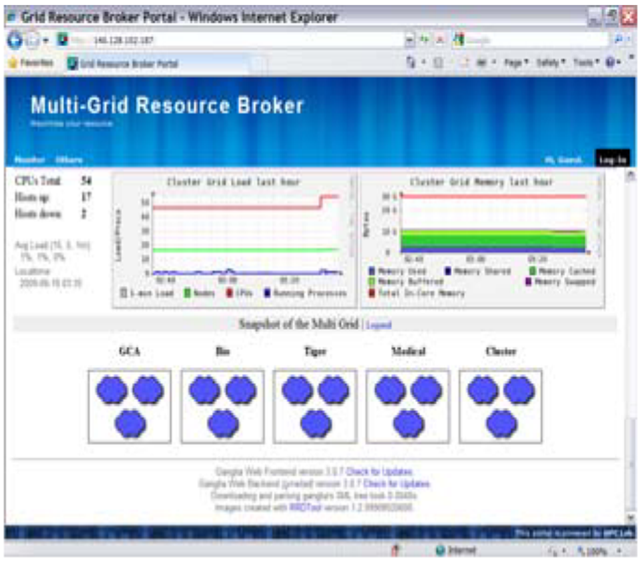

Grid Resource Broker Portal - Windows Internet Explorer
Multi-Grid Resource Broker
Tiger Grid Report for Thu, 18 Jun 2009 13:34:01 +0800
Get Fresh Data
Multi Grid > Tiger Grid >
Tiger Grid Load last day
Tiger Grid Memory last day
gamma Load last day
gamma Memory last day

**Figure 3** Multi-grid has integrated clusters and grids environments into a single Ganglia web portal (see online version for colours)


Grid Resource Broker Portal - Windows Internet Explorer
Multi-Grid Resource Broker
Cluster Grid Load last hour
Cluster Grid Memory last hour
Snapshot of the Multi Grid
GCA
Bio
Tiger
Medical
Cluster

As shown in Figure 3, we could oversee several clusters or grids environments via web portal provided by Ganglia.

## 2.2 *Network information provider*

The NWS (http://nws.cs.ucsb.edu/ewiki/) (Wolski et al., 1999) is a distributed system that detects computational resource and network status by periodic monitors and dynamic forecasts over a given time interval. The service operates a distributed set of performance sensors (network monitors, CPU monitors, etc.) from which it gathers system condition information. It then uses numerical models to generate forecasts of what the conditions will be for a given time period. The NWS system includes sensors for end-to-end TCP/IP performance (bandwidth and latency), available CPU percentage and available non-paged memory. The sensor interface, however, allows new internal sensors to be configured into the system. Some functions provided by NWS have overlapped with Ganglia; therefore, we primarily use NWS for end-to-end TCP/IP measurements.

As Wolski et al. (1999) mentioned, NWS was designed to maximise four possible conflicting functional characteristics. It must meet these goals despite the highly dynamic execution environment and evolving software infrastructure provided by shared meta-computing systems.

- predictive accuracy
- non-intrusiveness
- execution longevity
- ubiquity.

So, we choose NWS as primary tool for end-to-end TCP/IP measurements and we have excellent work in previous project. Except Ganglia, we also successfully deployed NWS service into our clusters and grids environments and then monitor network status via integrated web portal (as shown in Figures 4 and 5).

**Figure 4** NWS service integrated with Ganglia web portal (see online version for colours)

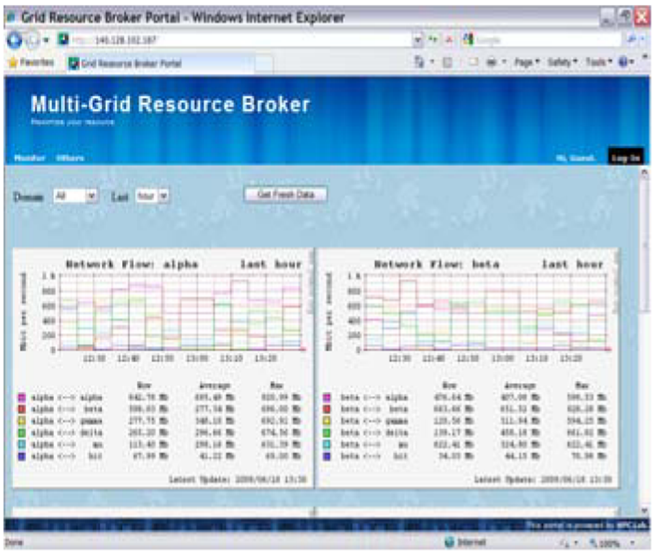
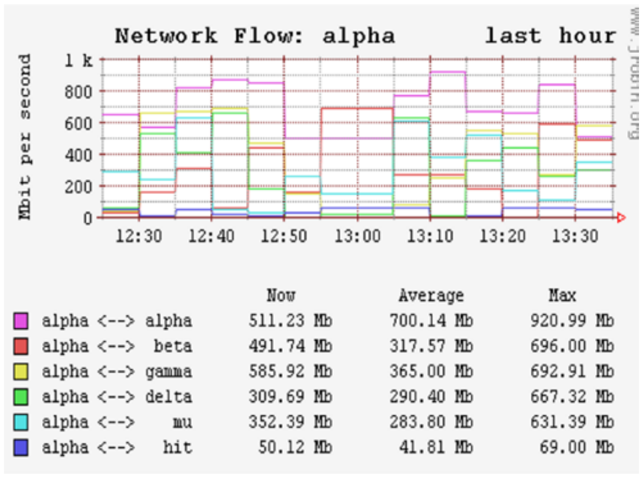

Grid Resource Broker Portal - Windows Internet Explorer
Multi-Grid Resource Broker
Domain
Get Fresh Data
Network Flow: alpha last hour
Network Flow: beta last hour

**Figure 5** Network statistics produced by NWS measurements demonstrated in web portal (see online version for colours)



| | Now | Average | Max |
|---|---|---|---|
| alpha <--> alpha | 511.23 Mb | 700.14 Mb | 920.99 Mb |
| alpha <--> beta | 491.74 Mb | 317.57 Mb | 696.00 Mb |
| alpha <--> gamma | 585.92 Mb | 365.00 Mb | 692.91 Mb |
| alpha <--> delta | 309.69 Mb | 290.40 Mb | 667.32 Mb |
| alpha <--> mu | 352.39 Mb | 283.80 Mb | 631.39 Mb |
| alpha <--> hit | 50.12 Mb | 41.81 Mb | 69.00 Mb |

## 2.3 *Quality of Service*

Quality of Service is the ability to provide different service priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow. It was widespread adopted in the field of computing networking, and we use it as a quality measurement of grid environments.

Quality of Service sometimes refers to the level of QoS, i.e., the guaranteed service quality. High QoS is an expectable crucial factor of highly reliable and high-performance grid environments.

Some characteristics, like 'Availability', 'Accessibility' or 'Maintainability', will also influence user experiences about the services provided by our system or services. To meet user requirements in diverse scenarios with sufficient quality, we are expected to have the ability to evaluate our performance in advance or in real-time manner. If not, how could we guarantee a certain level of QoS? Some researchers have proposed network performance evaluation model (Que et al., 2008) to aided network administrators to effectively analyse network performance and then adjust network devices properly. For us, we investigate how to propose a heuristic QoS measurement that could provide various, specific combinations of information from our grid environments.

## *2.4 Network management system*

In our previous project, we have constructed a web portal composed of Ganglia and NWS service for overseeing grid environments. As time goes on, we find that it is inefficient to manage these resources passively. We had better made use of NMS, which could help us to manage and monitor numerous resources of grid environments actively. NMS is a combination of hardware and software used to monitor and administer a network. The primary communication mechanism between NMS and network devices and grid nodes is based on SNMP. Then, we chose NINO (http://nino.sourceforge.net/nino/index.html) as shown in Figure 6 for ourgrid computing environments before long. NINO is not the most powerful NMS, but we choose it owing to some proper features to for our need.

For device and network discovery:

- *Network discovery*: NINO is able to discover network.
- IP address or default gateway and an optional alternate community string, and NINO will scan all devices and store it in the database. NINO uses the routing table of each router to scan the network.
- *Scan IP range*: NINO is able to scan network for SNMP capable devices. Just enter a range, start IP address, end IP address and an optional alternate community string, and NINO will scan all devices and store it in the database.
- *Network info*: NINO is also able to look inside a router, find routing tables and scan all attached networks. All network info per device will be stored into the database. In the device view, also the current routing table can be displayed, showing routing protocol, age, next hop, metric, etc.

For monitoring:

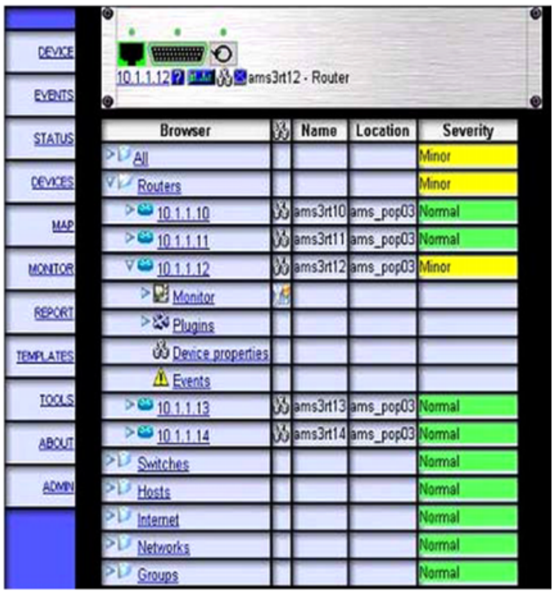
- *Monitoring*: NINO can monitor servers, routers, switches and applications.
- *Events*: NINO can send and receive SNMP traps. Traps are stored in the event log. Event actions can be defined to send e-mail alerts, escalation traps or command line scripts. NINO also has a trap-forwarding functionality to integrate NINO into larger networks. Traps can be forwarded per event or using source filters (i.e., forward all traps coming from: 10.1.1.*).
- *Monitoring presets and preset groups*: NINO uses monitoring presets to monitor devices. Presets can be stored in a preset group, i.e., all Windows presets (CPU, Disk, memory, network statistics) are grouped in a Windows preset group. Monitoring presets can use SNMP, WMI or Service Response to monitor a device. It is possible to create customised monitoring presets. Default presets are available for Cisco routers or switches, Windows, Linux and hosts using the standard Host Resources MIB.

For administration and diagnostics:

- *Database*: Browse SQL table content and edit rows. The import/export utility can be used to import/export SQL tables from/to a tab-delimited file.

After evaluation, we believe that we could integrate NINO with our previous work and help us to manage grid environments more actively. Meanwhile, we are working on integration of Ganglia, NWS and NINO.

**Figure 6** NINO's screenshot – the device browser, with severity status, monitoring drill down and plug-ins (see online version for colours)



10.1.1.12 ams3rt12 - Router

| Browser | | Name | Location | Severity |
|---|---|---|---|---|
| All | | | | Minor |
| Routers | | | | Minor |
| 10.1.1.10 | | ams3rt10 | ams_pop03 | Normal |
| 10.1.1.11 | | ams3rt11 | ams_pop03 | Normal |
| 10.1.1.12 | | ams3rt12 | ams_pop03 | Minor |
| Monitor | | | | |
| Plugins | | | | |
| Device properties | | | | |
| Events | | | | |
| 10.1.1.13 | | ams3rt13 | ams_pop03 | Normal |
| 10.1.1.14 | | ams3rt14 | ams_pop03 | Normal |
| Switches | | | | Normal |
| Hosts | | | | Normal |
| Internet | | | | Normal |
| Networks | | | | Normal |
| Groups | | | | Normal |

## 3 Heuristic QoS measurement

In our previous project, we have built an integrated grid environment including a web portal composed of Ganglia and NWS service. Afterwards, we start another project about Picture Archive and Communication System (PACS) (Yang et al., 2008a) and most experiments were done in the same platform. The primary mission in this project is to exchange medical images efficiently with specific application developed by our team. The application, named 'Cyber' (Yang et al., 2008b), has successfully integrated eight algorithms. For exchanging medical images efficiently with these algorithms integrated in Cyber, we have to configure a lot of parameters before tasks submitted. Unfortunately, we have no idea what is best combination of parameters we should take in advance. Therefore, we adopt "try and error method" unavoidably. But, it is definitely not practical for most conditions. For this reason, we expect to establish an automation of parameters self-optimisation. To guarantee a degree of QoS, we regard user requirements as constraints of tasks. With these constraints and heuristic QoS measurements we proposed in this paper, we could provide more QoS to meet user requirements.

## *3.1 Domain-based network information model*

In this paper, we adopt Domain-based Network Information Model (Yang et al., 2005, 2007a, 2007b) for NWS services deployment. The Domain-based Network Information Model is designed for solving a complete point-to-point bandwidth measurement problem. After investigating by experiments in physical environments, we can be sure that Domain-based Network Information Model is helpful for reducing network measurements. The measurement model and design of Domain-based Network Information Model are shown as Figures 7 and 8.

**Figure 7** The domain-based network measurement model

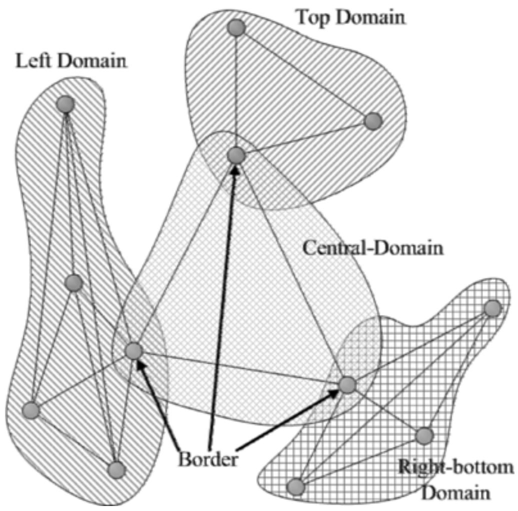
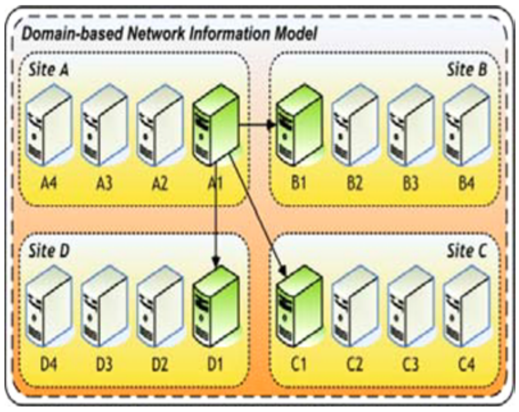


**Figure 8** The design of domain-based network information model (see online version for colours)



For example, assume a Grid with $n$ nodes. Each node measures the links between itself and all other nodes every $T$ seconds (e.g., $T = 1\sim3$ s) for a total of NMN ($n$) network measurements.

$$\text{NMN}(n) = n \times (n-1). \tag{1}$$

In large-scale Grid environments, the number of network measurements grows quickly. UniGrid and TigerGrid with, respectively, 96 and 46 nodes generate NMN(96) = 9120 and NMN(46) = 2070 measurements. Thus, network traffic will be very heavy, particularly when underlying Grid intra-traffic is originally busy.

Our previous work (Yang et al., 2007c) used the domain-based network information model shown in Figure 9. Figure 10 shows four sites, each containing four nodes. The sites each have a head node, e.g., A1, B1, C1 and D1, are, respectively, the head nodes of sites A, B, C and D. Each head node in this model periodically measures the links between itself and the other three head nodes. Each head node also periodically measures the links between itself and all other nodes in its site. Hence, using the domain-based network information model, the measurement number will be dramatically reduced to

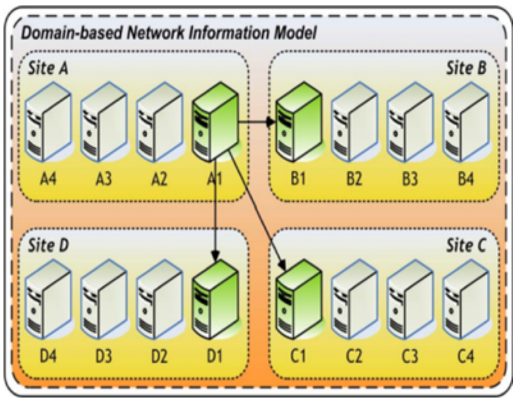
$$\text{NMS}(n,[n_i]) = \text{NMN}(n) + \sum \text{NMN}(n_i) \tag{2}$$

where $n_i$ is the total number of nodes in site $i$. In UniGrid and TigerGrid, the numbers of network measurements will be decreased to NMS(31, [4, 8, 8, 5, 8, 5, 7, 2, 1, 1, 3, 1, 4, 1, 3, 1, 1, 4, 8, 1, 1, 1, 1, 2, 2, 1, 4, 1, 2, 4, 1]) = 1316 and NMS(12, [4, 4, 4, 4, 8, 2, 3, 4, 4, 4, 4, 1]) = 292, respectively. The reduction rate $R$ is defined as:

$$R = \frac{\text{NMN}(n) - \text{NMS}(n,[n_i])}{\text{NMN}(n)}. \tag{3}$$

Compared with NMN(96) and NMN(46), the $Rs$ are 86.01% and 85.94%, respectively, which shows the obvious efficiency of the model.

**Figure 9** Previous design of domain-based network information model (see online version for colours)



Even though this model can eliminate huge amounts of measurement effort and bandwidth use, it lacks network information between pairs of nodes belonging to different sites (unless both are borders). For example, the link (target) between nodes A2 and B1 shown in Figure 7 is not measured.

In this model, it reduces a large number of connections, but it lacks network information of nodes except head nodes in two different sites. This model carries out an estimation model that provides network information of nodes in two different sites, but one of the two nodes should be a head node of site. For example, the link between Node A2 and B1 is not performed in this model, which is shown in Figure 3.

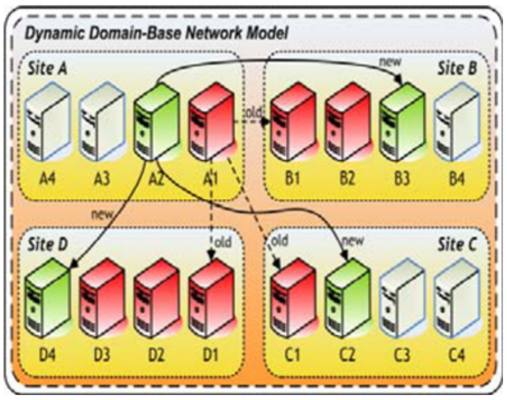
The Domain-based Network Information Model reduced the number of bandwidth measurement between all Grid Nodes, but it lacks network information between two Nodes other than the head Node located in two different sites other than the head Node. For example, the bandwidth measurement between Nodes A2 and B3 is not performed in this model.

We further enhanced the static model by improving the switching mechanism in the dynamic domain-based network information model. Figure 10 shows an example. The principal improvement is switching the site head node to the next free node. For example, when node A1 is busy, the next free node, node A2, becomes the head node of site A, and measures the bandwidth between itself and nodes B3, C2 and D4, if they are the respective free nodes in sites B, C and D. The purpose is to avoid having a busy node still act as a border, which would decrease system performance. There are three obvious advantages in using this model.

- first, the number of bandwidth measurements is the same as that for a static model; the measurement time complexity is not worsened
- second, bandwidth measurements between pairs of arbitrary nodes belonging to different sites are easily obtained
- finally, network bandwidth measurements obtain real values instead of estimated values, thus enabling the Resource Broker to effectively schedule jobs allocated to multiple sites.

If we could dynamically change each header of all Grid Nodes, we could obtain the advantages described earlier. There is an issue derived from this requirement consequently, i.e., how to choose headers dynamically instead of manual operation. Hence, we regard Heuristic QoS Measurement as a solution for this issue. After integration with NINO, we expect to attain this goal as soon as possible.

**Figure 10** The design of domain-based network information model (see online version for colours)



## 3.2 NWS deployment and flowchart

While deploying NWS services, we paid attention to try to get rid of intruding existed services on each grid nodes. In most cases, we deploy only one nameserver and multiple sensors on each computing resources. Besides, arbitrary 'Persistence State' may be set up in different locations. In this paper, we simply designate one nameserver, one memory server and one clique for a group of grid nodes.

We regard several grid nodes as a group, and each group has a header to deploy nameserver and memoryserver. A simple NWS services deployment procedure that we used is divided into three steps:

- clean all NWS process
- load NWS services
- register NWS clique.

And the standard procedure we wrote in shell scripts is shown as Figure 11. Owing to the non-intrusiveness characteristic of NWS, these shell scripts we wrote could be executed without root privilege.

Figure 12 has shown a simple flowchart we used. In this paper, we have edited crontab to schedule some routines for loading NWS information into database automatically and backing up raw data as plain text files locally.

**Figure 11** Procedure of NWS services deployment

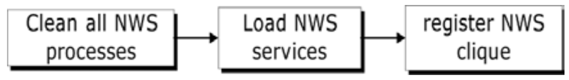
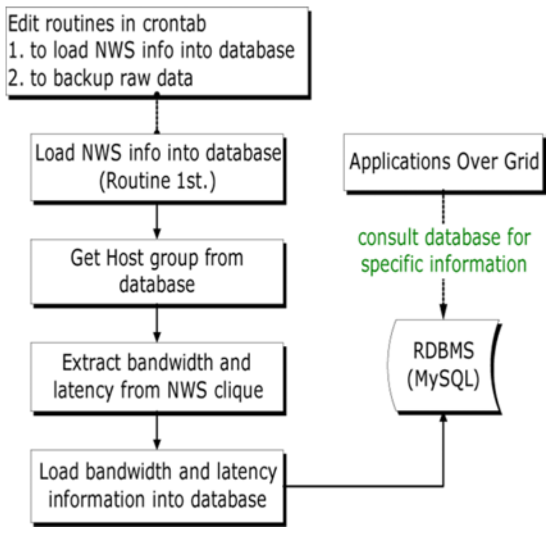


**Figure 12** The flowchart of gathering network information (see online version for colours)


Edit routines in crontab
1. to load NWS info into database
2. to backup raw data
Load NWS info into database (Routine 1st.)
Get Host group from database
Extract bandwidth and latency from NWS clique
Load bandwidth and latency information into database
Applications Over Grid
consult database for specific information
RDBMS (MySQL)

While routines that we scheduled in crontab are invoked, customised shell scripts that we wrote are executed. The first step of the shell script is to get host groups from database for NWS information gathering. Each host group is pre-defined in database and will be assigned a clique for measuring network status. After the clique is created, it will measure network information in an equal time interval, for example, 30 s. Then, the script will extract bandwidth and latency from NWS clique, respectively. If successes, it will load bandwidth and latency information into database.

The second routine that we defined to keep raw data as plain text files locally is designed for future use. Currently, it just provides a different storage than database to keep raw information of NWS services.

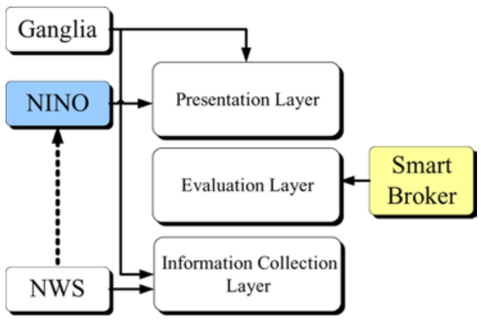
## *3.3 Heuristic approach*

Statistics is helpful in many fields, especially for prediction. In this paper, we gathered historical network information of grid environments and stored it into database. Applications could simply query database for network statistics with aggregative functions provided by RDBMS, like Max (), Min (), AVG () and so on. After analysing these statistics, applications can dynamically adjust their parameters about network for better performance without sending request to estimate network status between all grid nodes in real-time manner.

Besides, we have planned an innovative method to obtain real-time network state that worked with Dynamic Domain-based Network Information Model, i.e., dynamically deploying clique into dedicated node, measuring network state, and then reporting results to database, users, or applications. The enhanced version of shell scripts that support Dynamic Domain-based Network Information Model is currently under development.

We have designed a simple model for integration of Ganglia, NWS and NINO (as shown in Figure 13). Ganglia and NINO provide UI for users to manage and monitor grid environments. NWS and Ganglia collect related information from hosts and network regularly. And 'Smart Broker' provides parameters to applications like Cyber.

**Figure 13** Integration of Ganglia, NWS and NINO (see online version for colours)



Smart Broker is the key component for QoS measurement. Original version of Cyber provides users an interface for tuning up parameters, which is shown as Figure 14. Smart Broker will help us to achieve automation of parameters self-optimisation in diverse scenarios. Smart Broker works as evaluation layer between applications and information collection layer. We have pre-defined four task types that perform QoS measurement in various ways.

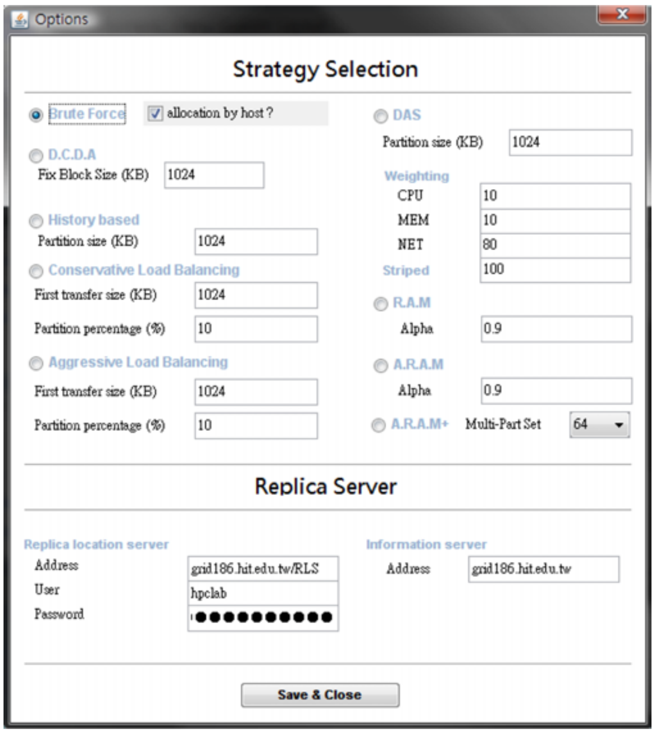
- download
- upload
- computational
- hybrid.

For example, Cyber is a typical application for 'Download' tasks. The QoS measurement we designed is to calculate the formula as follows:

$$E(\text{QoS}) = \alpha \times RM + (1-\alpha) \times HM. \qquad (4)$$

$E$(QoS) is expected value of QoS and $RM$ is real-time bandwidth measurement between nodes. $HM$ is historical statistics. $\alpha$ is the constant between real-time measurement and historical statistics. In the initial stage, we may set $\alpha$ to 0.5. With more and more tasks submitted, Smart Broker will adjust $\alpha$ dynamically. $\alpha$ is not always the same in different grid environments. We just try to use $\alpha$ to predict QoS in diverse grid environments more effectively.

**Figure 14** Strategy selection – UI provided by Cyber for parameters input (see online version for colours)



Strategy Selection

[✓ Brute Force] [✓ allocation by host ?]
D.C.D.A
Fix Block Size (KB) 1024
History based
Partition size (KB) 1024
Conservative Load Balancing
First transfer size (KB) 1024
Partition percentage (%) 10
Aggressive Load Balancing
First transfer size (KB) 1024
Partition percentage (%) 10
DAS
Partition size (KB) 1024
Weighting
CPU 10
MEM 10
NET 80
Striped 100
R.A.M
Alpha 0.9
A.R.A.M
Alpha 0.9
A.R.A.M+ Multi-Part Set 64

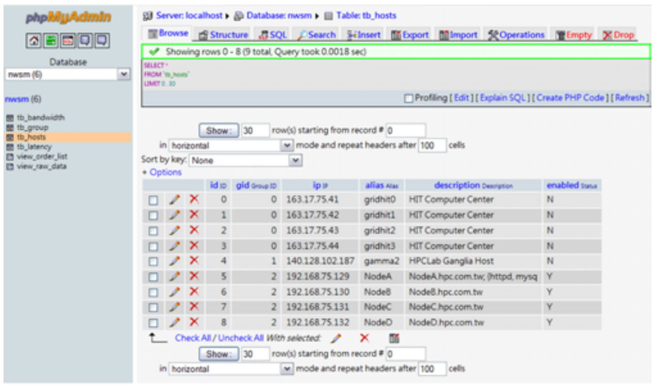
Replica Server

Replica location server
Address grid186.hit.edu.tw/RLS
User hpclab
Password ●●●●●●●●●●●
Information server
Address grid186.hit.edu.tw

## 4 Experimental environment and results

To verify the architect we proposed in the initial stage, we do not deploy NWS services and RDBMS onto physical environments. And we have built our test bed on virtualised environments instead. We have created four virtual machines and then installed Fedora 9 as default operation system. After deploying NWS services with customised scripts we wrote, we also built an open-source database for experiment. In this paper, we chose MySql as default RDBMS. Figure 15 shown here lists our experimental grid nodes in the initial stage.

**Figure 15** Experimental grid nodes' information (see online version for colours)



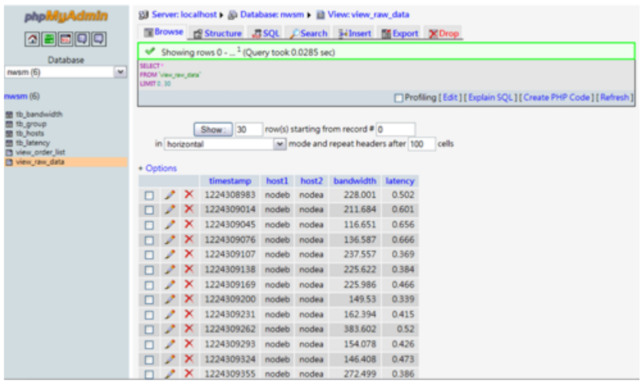
Showing rows 0 - 8 (9 total, Query took 0.0018 sec)

```
SELECT *
FROM `tb_hosts`
LIMIT 0, 30
```

| id | gid | ip | alias | description | enabled |
|---|---|---|---|---|---|
| 0 | 0 | 163.17.75.41 | gridhit0 | HIT Computer Center | N |
| 1 | 0 | 163.17.75.42 | gridhit1 | HIT Computer Center | N |
| 2 | 0 | 163.17.75.43 | gridhit2 | HIT Computer Center | N |
| 3 | 0 | 163.17.75.44 | gridhit3 | HIT Computer Center | N |
| 4 | 1 | 140.128.102.187 | gamma2 | HPCLab Ganglia Host | N |
| 5 | 2 | 192.168.75.129 | NodeA | NodeA.hpc.com.tw; (httpd, mysq | Y |
| 6 | 2 | 192.168.75.130 | NodeB | NodeB.hpc.com.tw | Y |
| 7 | 2 | 192.168.75.131 | NodeC | NodeC.hpc.com.tw | Y |
| 8 | 2 | 192.168.75.132 | NodeD | NodeD.hpc.com.tw | Y |

Figures 16 and 17 shown here demonstrate raw data and bandwidth information that we gathered with NWS services and loaded into database with our customised scripts. In this paper, we schedule a routine to extract and load network information into database every 5 min. Owing to Domain-based Network Information Model, overheads of measurement have been highly reduced and it would not take lots of time to load data into databases.

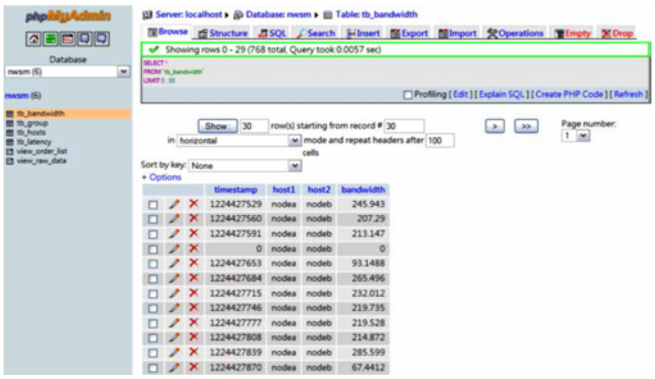
**Figure 16** Raw data gathered from NWS services (see online version for colours)



```
SELECT *
FROM `view_raw_data`
LIMIT 0, 30
```

| timestamp | host1 | host2 | bandwidth | latency |
|---|---|---|---|---|
| 1224308983 | nodeb | nodea | 228.001 | 0.502 |
| 1224309014 | nodeb | nodea | 211.684 | 0.601 |
| 1224309045 | nodeb | nodea | 116.651 | 0.656 |
| 1224309076 | nodeb | nodea | 136.587 | 0.666 |
| 1224309107 | nodeb | nodea | 237.557 | 0.369 |
| 1224309138 | nodeb | nodea | 225.622 | 0.384 |
| 1224309169 | nodeb | nodea | 225.986 | 0.466 |
| 1224309200 | nodeb | nodea | 149.53 | 0.339 |
| 1224309231 | nodeb | nodea | 162.394 | 0.415 |
| 1224309262 | nodeb | nodea | 383.602 | 0.52 |
| 1224309293 | nodeb | nodea | 154.078 | 0.426 |
| 1224309324 | nodeb | nodea | 146.408 | 0.473 |
| 1224309355 | nodeb | nodea | 272.499 | 0.386 |

**Figure 17** Bandwidth information between grid nodes (see online version for colours)



Showing rows 0 - 29 (768 total, Query took 0.0057 sec)

```
SELECT *
FROM `tb_bandwidth`
LIMIT 0, 30
```

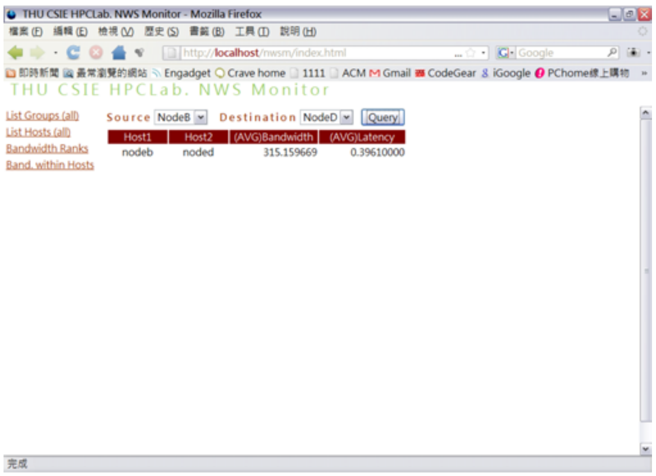
| timestamp | host1 | host2 | bandwidth |
|---|---|---|---|
| 1224427529 | nodea | nodeb | 245.943 |
| 1224427560 | nodea | nodeb | 207.29 |
| 1224427591 | nodea | nodeb | 213.147 |
| 0 | nodea | nodeb | 0 |
| 1224427653 | nodea | nodeb | 93.1488 |
| 1224427684 | nodea | nodeb | 265.496 |
| 1224427715 | nodea | nodeb | 232.012 |
| 1224427746 | nodea | nodeb | 219.735 |
| 1224427777 | nodea | nodeb | 219.528 |
| 1224427808 | nodea | nodeb | 214.872 |
| 1224427839 | nodea | nodeb | 285.599 |
| 1224427870 | nodea | nodeb | 67.4412 |

Figures 18 and 19 shown here demonstrate a simple web portal prototype for users to query bandwidth between host pairs and rank of grid nodes. Certainly, we could also provide more functions for variety of purposes. For example, a formula given by the MSRA Algorithm (Yang and Chen, 2008) to compute total performance power of the site or site pair is shown as follows:

$$TP = \beta \times \sum CP + (1 - \beta) \times NP \qquad (5)$$

where *CP* is the computing performance power of site of site pair, *NP* is the network performance power of a set of links that among sites in a site pair, and $\beta$ is the effect ratio used to regulate the percentage of *CP* and *NP*. If *NP* is estimated imprecisely, *TP* will be computed imprecisely, too. To provide more accurate network information, we pay close attention to both historical and dynamic network information.

**Figure 18** Web portal prototype for users to query bandwidth between grid nodes (see online version for colours)

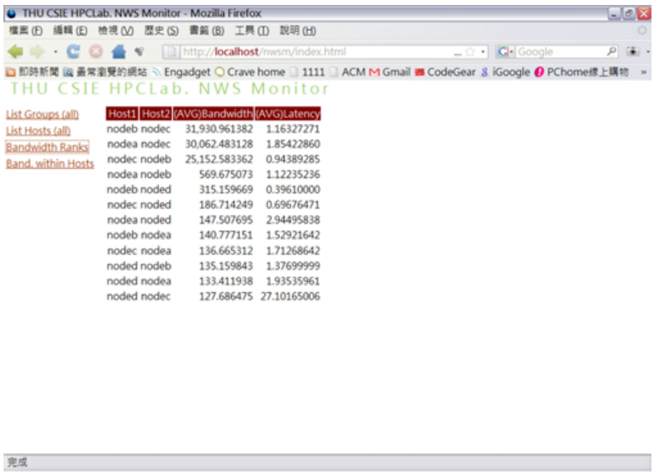


THU CSIE HPCLab. NWS Monitor

Source: NodeB  Destination: NodeD  Query

| Host1 | Host2 | (AVG)Bandwidth | (AVG)Latency |
|---|---|---|---|
| nodeb | noded | 315.159669 | 0.39610000 |

**Figure 19** Grid nodes bandwidth rank (see online version for colours)



THU CSIE HPCLab. NWS Monitor

| Host1 | Host2 | (AVG)Bandwidth | (AVG)Latency |
|---|---|---|---|
| nodeb | nodec | 31,930.961382 | 1.16327271 |
| nodea | nodec | 30,062.483128 | 1.85422860 |
| nodec | nodeb | 25,152.583362 | 0.94389285 |
| nodea | nodeb | 569.675073 | 1.12235236 |
| nodeb | noded | 315.159669 | 0.39610000 |
| nodec | noded | 186.714249 | 0.69676471 |
| nodea | noded | 147.507695 | 2.94495838 |
| nodeb | nodea | 140.777151 | 1.52921642 |
| nodec | nodea | 136.665312 | 1.71268642 |
| noded | nodeb | 135.159843 | 1.37699999 |
| noded | nodea | 133.411938 | 1.93535961 |
| noded | nodec | 127.686475 | 27.10165006 |

Historical network information is available with current work and dynamical network information measurement via web portal is under development. We are dedicated to rewrite a couple of shell scripts to measure network information dynamically without root privilege of grid nodes.

## 5 Conclusions and future work

In this paper, we use Domain-based Network Information Model for experiments, but it is not a proper model for dynamic grid environments. If any grid nodes that cause hardware failure or just have been reassigned to another IP, we have to manually reconstruct NWS cliques. This has already mentioned as drawback of NWS (Legrand and Quinson, 2004). In large-scale grid environments, it is a complicated task to manage these cliques and hosts' relations. Our future work will be adopting Dynamic Domain-based Network Information Model for next deployment so as to reduce overheads come from complicated management tasks.

Besides, we defined a standard operation procedure for managing grid nodes semi-automatically. And, we could simply manage grid nodes via web portal instead of writing shell scripts. To guarantee a certain degree of QoS, we also proposed a heuristic method to predict QoS from diverse grid environments for Download-oriented tasks. Furthermore, we expect that users could manage and monitor numerous resources of grid environments more effectively and efficiently.

## Acknowledgement

This work is supported in part by the National Science Council, Taiwan ROC, under grants no. NSC 96-2221-E-029-019-MY3, NSC 97-2622-E-029-003-CC2 and NSC 98-2622-E-029-001-CC2.

## References

Krauter, K., Buyya, R. and Maheswaran, M. (2002) 'A taxonomy and survey of grid resource management systems for distributed computing', *Softw. Pract. Exper.*, Vol. 32, No. 2, pp.135–164.

Krefting, D., Vossberg, M. and Tolxdorff, T. (2008) 'Simplified grid implementation of medical image processing algorithms using a Workflow management system', in Olabarriaga, S.D., Lingrand, D. and Montagnat, J. (Eds.): *Medical Imaging on Grids: Achievements and Perspectives*, MICCAI-Grid Workshop, 6 September, New York, NY, http://www.i3s.unice.fr/~johan/MICCAI-Grid08/pdf/kreftingMICCAIG.pdf

Legrand, A. and Quinson, M. (2004) 'Automatic deployment of the network weather service using the effective network view', Paper presented at the *Parallel and Distributed Processing Symposium*, *Proceedings*, *18th International*, Santa Fe, New Mexico.

Que, W-K., Zhang, G-Q. and Wei, Z-H. (2008) 'Model for IP network synthetical performance evaluation', *Computer Engineering*, Vol. 34, No. 8, pp.99–101, ISSN:1000-3428 (2008)08-0099-03.

Wolski, R., Spring, N.T. and Hayes, J. (1999) 'The network weather service: a distributed resource performance forecasting service for metacomputing', *Future Generation Computer Systems*, Vol. 15, Nos. 5, 6, pp.757–768.

Yang, C-T. and Chen, S-Y. (2008) 'A multi-site resource allocation strategy in computational grids', *Advances in Grid and Pervasive Computing*, 25–28 May, Kunming, China, pp.199–210.

Yang, C-T., Chen, C-H., Yang, M-F. and Chiang, W-C. (2008a) 'MIFAS: medical image file accessing system in co-allocation data grids', *IEEE Asia-Pacific Services Computing Conference*, December, Ilan, Taiwan, pp.769–774.

Yang, C-T., Yang, M-F. and Chiang, W-C. (2008b) 'Implementation of a cyber transformer for parallel download in co-allocation data grid environments', in Shenzhen, G.D. (Ed.): *Proceedings of the 7th International Conference on Grid and Cooperative Computing (GCC2008) and Second EchoGRID Conference*, 24–26 October, China, pp.242–253.

Yang, C-T., Shih, P-C., Lin, C-F. and Chen, S-Y. (2007a) 'A resource broker with an efficient network information model on grid environments', *The Journal of Supercomputing*, Vol. 40, No. 3, pp.249–267.

Yang, C-T., Chen, S-Y. and Chen, T-T. (2007b) 'A grid resource broker with network bandwidth-aware job scheduling for computational grids', *Advances in Grid and Pervasive Computing*, Paris, France, pp.1–12.

Yang, C-T., Chen, T-T. and Tung, H-Y. (2007c) 'A dynamic domain-based network information model for computational grids', Paper Presented at the *Future Generation Communication and Networking* (FGCN), Jeju-Island, Korea, Vol. 1, pp.575–578.

Yang, C-T., Shih, P-C., Chen, S-Y. and Shih, W-C. (2005) 'An efficient network information model using NWS for grid computing environments', *Grid and Cooperative Computing – GCC 2005*, Vol. 3795, pp.287–299.

## Websites

Ganglia: http://ganglia.info/

Network Weather Service (NWS): http://nws.cs.ucsb.edu/ewiki/

NINO: http://nino.sourceforge.net/nino/index.html

J Supercomput
DOI 10.1007/s11227-009-0302-9

# File replication, maintenance, and consistency management services in data grids

**Chao-Tung Yang · Chun-Pin Fu · Ching-Hsien Hsu**

**Abstract** Data replication and consistency refer to the same data being stored in distributed sites, and kept consistent when one or more copies are modified. A good file maintenance and consistency strategy can reduce file access times and access latencies, and increase download speeds, thus reducing overall computing times. In this paper, we propose dynamic services for replicating and maintaining data in grid environments, and directing replicas to appropriate locations for use. To address a problem with the Bandwidth Hierarchy-based Replication (BHR) algorithm, a strategy for maintaining replicas dynamically, we propose the Dynamic Maintenance Service (DMS). We also propose a One-way Replica Consistency Service (ORCS) for data grid environments, a positive approach to resolving consistency maintenance issues we hope will strike a balance between improving data access performance and replica consistency. Experimental results show that our services are more efficient than other strategies.

**Keywords** File replication · Dynamic maintenance · Consistency management · Data grids

## 1 Introduction

In recent years, many fields such as bioinformatics, climate transition, earthquake simulation, space shuttle flight simulation, weather prediction, and high-energy

C.-T. Yang (✉) · C.-P. Fu
Department of Computer Science, Tunghai University, Taichung, 40704 Taiwan, ROC
e-mail: ctyang@thu.edu.tw

C.-P. Fu
e-mail: socollkimo@pchome.tw

C.-H. Hsu
Department of Computer Science and Information Engineering, Chung Hua University, Hsinchu, 30013 Taiwan, ROC
e-mail: chh@chu.edu.tw

Published online: 04 July 2009

physics in Europe [3] have come to require more and more computing power to generate results. Those simulation results in turn produce terabytes, even petabytes of data. Only computer centers with many supercomputers and storage devices are sufficient to handle these data. However, data grid technologies, developed to solve these kinds of problems, offer an effective alternative means of utilizing large-scale computing power and storage capacities to compute and store data. Grids [11, 12, 20, 21, 27, 29–34] enable sharing of computing power and storage capacities geographically distributed around the world such that they work together as tremendous virtual computers [1, 2] on experiments and simulations. The Globus Toolkit [13, 27] is open-source software for building data grid environments. It provides middleware for creating information infrastructures including resource management, data management, communication, fault detection, security, and portability.

Data replication and consistency [5, 26] refer to the same data being stored in distributed sites, and kept consistent when one or more copies are modified. A good file maintenance and consistency strategy can reduce file access times and access latencies, and increase download speeds, thus reducing overall computing times. And if one storage site breaks, users can fetch desired data from another storage site, which improves overall fault tolerance and contributes to making the entire grid environment more stable and reliable. Another advantage of data grids is the ability grid users have to download data in parallel from the better sites they choose or an application chooses automatically after evaluating with a grid environment evaluation model. The bandwidth utilization of those parallel links is the most important factor affecting overall download speeds. Network environments vary, which means that replica sites also vary in their ability to download data efficiently. Replica files should be kept consistent and downloaded from storage sites nearest users to reduce download times and ensure high performance.

This paper presents two services, the Dynamic Maintenance Service (DMS) for maintaining files in data grid systems, and the One-way Replica Consistency Service (ORCS) for keeping all copies of files consistent when one is modified. The DMS automatically maintains data statuses such as access frequency, space available on storage elements to which data will be replicated or migrated, and the network statuses of file source sites to other sites. The ORCS provides asynchronous data replication and replica consistency mechanisms that can reduce replica maintenance costs and free up storage space for new data or temporary data produced by experiments or simulations to avoid creating too many identical replicas. Users can easily find the best replica sites for downloading desired data, thus increasing replica usage rates, and improving storage device usage efficiency ratios over other strategies.

The contribution of this paper is to help make data grid environments more efficient by using the DMS and ORCS algorithms. Using DMS adjusts data to locations appropriate to the sites that request the data more often, thus reducing the times required by those sites to get needed data and improving performance. Using ORCS improves accessing performance by keeping replica content consistent and improving data grid storage device usage ratios. The DMS and ORCS algorithms also consider storage element free space when storing new and temporary data produced while computing. This decreases the probability of applications crashing or having to resubmit jobs to other computing resources for processing. Our experimental results show that DMS and ORCS are more efficient than other strategies, increase computing performance, and make storage element usage ratios more efficient.

The rest of this paper is organized as follows. In Sect. 2 we give background information on data grids, grid computing, some grid middleware, and previously proposed related works on data replication strategies. In Sect. 3, we describe DMS and ORCS system components, component details and data grid framework design and algorithms. We also introduce the parameters and evaluation model that determine when replica adjustment data should be sent to appropriate locations to keep them consistent. In Sect. 4, we give experimental results and comparisons of the DMS and ORCS algorithms with other strategies. Section 5 concludes the paper and indicates areas for future work.

## 2 Related work

### 2.1 Replica management

Replica management involves creating and removing replicas at data grid sites [28, 29]. Most often, these replicas are exact copies of original files, created only to harness certain performance benefits. A replica manager typically maintains a replica catalog containing replica site addresses and file instances. The replica management service is responsible for managing replication of complete and partial copies of datasets, defined as collections of files.

The replica management service is just one data grid environment component providing support for high-performance, data-intensive applications. A replica or location is a subset of a collection stored on a particular physical storage system. There may be multiple, possibly overlapping, subsets of collections stored on multiple storage systems in a data grid. These grid storage systems may use a variety of underlying storage technologies and data movement protocols independent of replica management.

Many studies on data maintenance in data grids have been published. In [24, 29], Ranganathan and Foster introduced six dynamic replication strategies, and compared them using a simulator called PARSEC to measure average response times and total bandwidth consumed by each strategy. The authors concluded that if grid users are concerned about lower response times, the cascading strategy is the better choice. On the other hand, if grid users consider bandwidth consumption to be the most important issue, fast spread is the best choice among all six strategies. These strategies do not consider whether there is enough free space to store temporary data and job results. Data no longer popular will occupy space that could be used to store temporary data and job results, which affects overall performance.

In [9, 16, 22, 23], the authors all mentioned an issue called the p-median problem: “given a set of n client points, find a set of p server points for those client points that minimizes the distance between each client point and its nearest server point”. In grid environments, minimizing total distance minimizes total response time. In [9, 16], and many other works, the authors indicate that the complexity class of the p-median problem is NP-hard in two or more dimensions, which means that finding p nodes to serve all nodes in a grid environment is also an NP-hard problem. In

[22, 23], Rashedur M. Rahman et al., proposed a static replica placement algorithm for placing replicas in the best p candidate nodes to minimize the total response time of each node using Lagrangian relaxation, which is a heuristic approach [10] to measuring the response time of each client node to its nearest server node. The algorithm is most likely the p-median problem. They also use user requests and network latency as parameters in deciding when to maintain replicas dynamically. They use a simulator called OptorSim [19], developed by the EU Data Grid project, to compare their method, and called dynamic p-median, with static p-median and Best_client. Static p-median replicates no files to other nodes in the data grid environment when user requests or network latency change. Best_client replicates the desired data to client nodes when request ratios for certain files in a node are very high. Simulation results show average response times for the authors' method are the lowest over various network loadings and user requests. Although dynamic p-median is a good method for dynamic replica maintenance, it can't be used in real grid environments since p-median is an NP-hard problem that consumes too much computing power determining new replica locations.

In [20], Sang-Min Park et al. proposed a dynamic replica maintenance algorithm called Bandwidth Hierarchy based Replication (BHR) that divides sites into many regions putting sites close to one another in the same regions in the bandwidth hierarchy. The BHR optimizer terminates replication if a replica duplicate already exists in another site in the same region. In [4], Ruay-Shiung Chang and Jih-Sheng Chang indicated that the BHR algorithm performs better than other strategies only when the storage element capacity is small. We found the following problem in BHR: If a file must be replicated in a region, BHR replicates it to one other site in the region. If an attempt is made to replicate the same file to a third site in the same region, BHR will see that there is already a duplicate file in the region and terminate. Thus, files will have at most two copies in each region, which means only two links will be available for parallel data downloading [1, 2, 30–34] in any one region. This limitation leads to high time costs, thus reducing the effectiveness of an important grid computing feature and adversely affecting overall performance. Furthermore, the two-copy practice will cause load imbalances on the sites where the copies are stored.

## 2.2 Replica consistency

Grid environment files modified by grid users raise the critical problem of maintaining data consistency among the replicas distributed across various machines. Over the past decade, considerable effort has been devoted to developing several consistency models. These studies concentrated on trading off consistency for performance and availability. The various consistency models developed include Strong, Weak, Continuous, Data-centric, Strict, Sequential, Eventual, Causal, FIFO, and Release. For instance, the strong consistency approach keeps data consistent across all replicas simultaneously, which requires many more resources and expensive protocols than other consistency models. The converse of strong consistency is weak consistency, which can tolerate inconsistencies for certain periods of time.

Many studies on replica consistency in data grid environments have been published [2, 4, 7, 8, 14, 15]. Data grid environments need consistency services to synchronize them when replicas are modified by grid users. The European Data Grid

Project laid out a Replica Consistency Service in [7] that provides an interface for grid users to update or add new files. It uses a “single master approach” in which replicas modified by grid users are updated by the Replica Consistency Service. In [8] Dirk Düllmann et al. proposed the high-level Grid Consistency Service (GCS), that performs updating, file synchronization, consistency maintenance, and allows grid users to choose dynamically adjusting the degree of replica consistency from entirely synchronized to loosely synchronized.

In [15], Jiafu Hu et al. proposed an asynchronous model for avoiding replica inconsistency in grid environments despite system failures and network traffic congestion. They suggest that consistency concerns can be divided into data consistency and metadata replica consistency. Their model is based on the work of the HEP community, which established the Particle Physics Data Grid (PPDG) and the Grid Physics Network (GriPhyN) projects. These projects used the Primary-copy (master–slave) approach in which only one copy (the master) can be updated, and secondary copies are updated by changes propagated from the master. There is one site that always has all updates; consequently, the load on the primary copy can be large. A pilot project called the Grid Data Management Pilot (GDMP) in the EDG adopted the Subscription and Relatively Independent Sites method, which regards data consistency more flexibly and local sites as independent.

Several replication and data consistency solutions are discussed in [7], including Eager (Synchronous) replication and Lazy (Asynchronous) replication, Single-Master and Multi-Master Models, and pull-based and push-based. In [14], Changqin Huang et al. proposed differentiated replication in order to improve accessing performance and replica availability. It is effective on performance, availability, and consistency, but the maintenance consistency algorithm does not take storage capacity into account. Replicas accessed only infrequently will consume free space on storage devices.

In this paper, we propose the Dynamic Maintenance Service (DMS) which is intended to address the issues discussed above and to correct the problem we pointed out in [20]. We use request frequency and storage element free space as parameters in determining when files should be adjusted. We also propose the One-way Replica Consistency Service (ORCS), which puts emphasis on increasing storage device usage ratios. Our algorithm allows more than two replicas of the same data at one site, thus increasing parallel download speeds and more fully utilizing data grid environment storage resources. Files with low access frequencies are automatically deleted, freeing space on storage elements to store temporary data and job results, and increasing storage element usage ratios.

## 3 System design and implementation

### 3.1 Software stack diagram

Software stack diagrams for each node and all sites in our data grid system are shown in Figs. 1 and 2. The functions of the three layers, bottom, middle, and top, are described below.

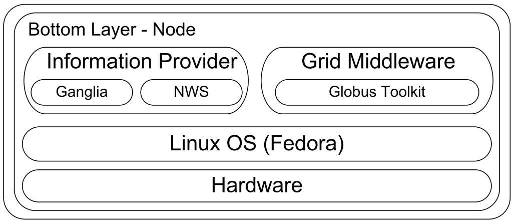


**Fig. 1** The software stack diagram of each node

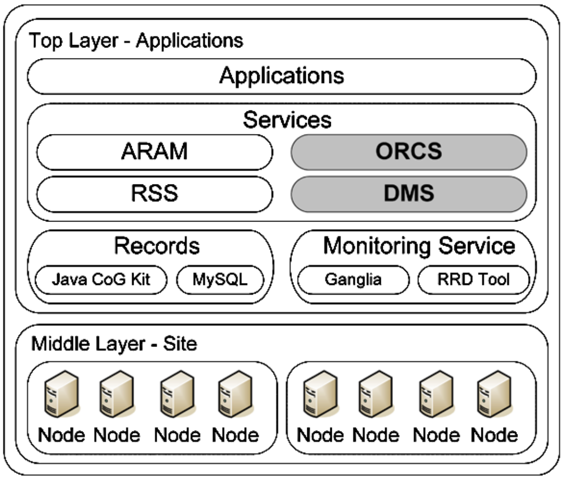


**Fig. 2** The software stack diagram of all sites, services, and portals

- Bottom Layer: shows the software installed on each node in the grid environment. The major components of the Bottom Layer are the Information Provider and Grid Middleware. The Information Provider consists of the Ganglia [35] and Network Weather Service (NWS) [18]. Ganglia gathers machine information such as numbers of processors and how many cores each has, the loading on each processor, total memory size and free space, and disk usage. The NWS gathers inter-node network bandwidths and each link's latency. The Grid Middleware consists of the Globus Toolkit [27], which is used to join nodes to the grid environment.
- Middle Layer: This layer is the Site, consisting of several nodes usually located in the same place or connected to the same switch or hub. Nodes in the Site are connected to one another via the Internet. Sites are usually built up as clusters, but each node has a real IP; the Site's first node is called the head node.
- Top Layer: This layer holds Applications, Services, the Monitoring Service, and Records [17]. Services consist of the Anticipative-Recursively-Adjusting Mechanism, Replica Selection Service, One-way Replica Consistency Service, and Dynamic Maintenance Service. Services operate on information gathered from the Monitoring Service and Records. Records can provide machine and file information prior to downloading files or adjusting file locations. The Monitoring Service provides a web front-end page for users to observe variations during job processing.

Relations among the components described above are shown in Fig. 3. The four services mentioned above are classified as User-side and System-side. The User-

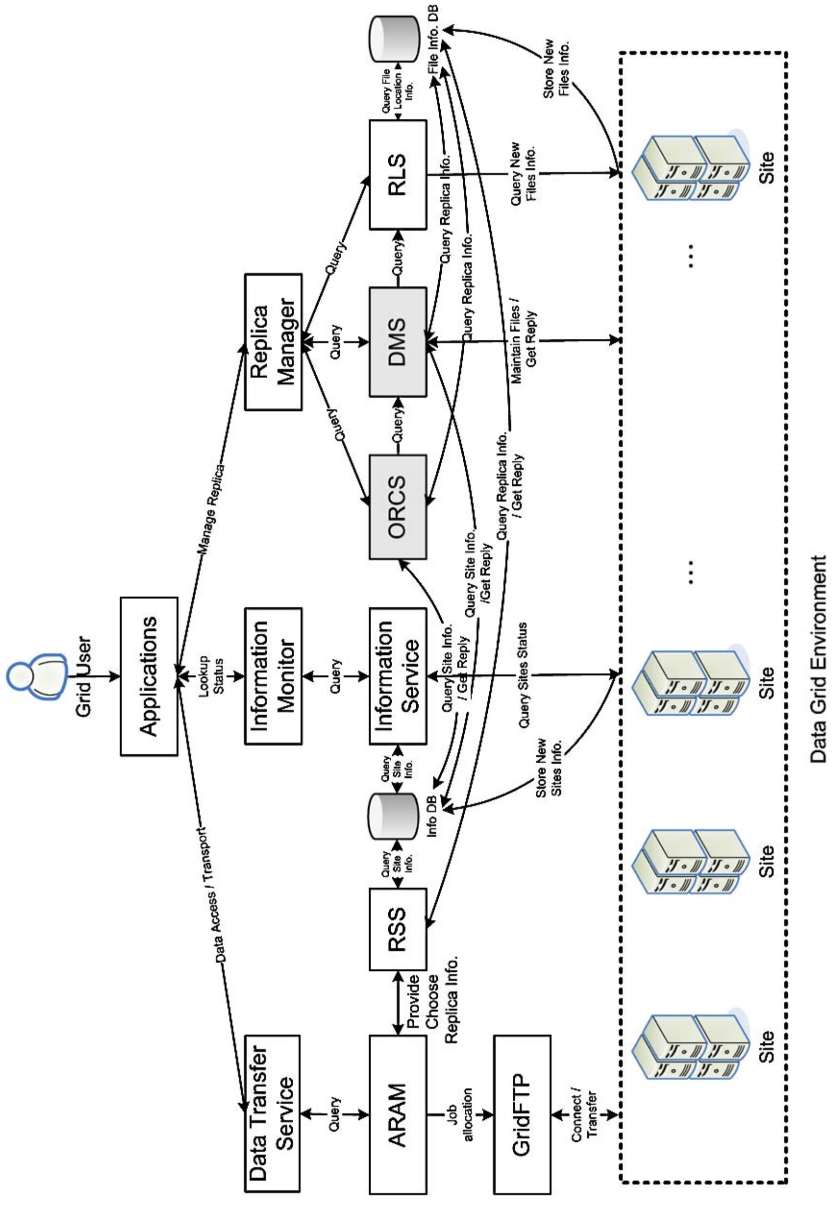


**Fig. 3** System architecture of data grid environment

side, which allows users to monitor application operations as the applications serve their needs, includes the Anticipative-Recursively-Adjusting Mechanism (ARAM) and Replica Selection Service (RSS). The System-side includes the Dynamic Maintenance Service (DMS) and One-way Replica Consistency Service (ORCS), which automatically direct files to appropriate locations and keep them consistent. Functional details of these services are described below.

- Replica Selection Service: gathers relevant information from the RLS and Information Service to determine which sites are better for the ARAM to use for downloading files.
- Anticipative-Recursively-Adjusting Mechanism: enables users to download desired data in parallel, dynamically adjusting download speeds according to network bandwidths between server nodes and client nodes, and balancing file site loadings.
- One-way Replica Consistency Service: keeps files consistent with duplicates stored in distributed nodes. When one file in a node is updated, it will notify the other nodes that have the same file to update to the newest version.
- Dynamic Maintenance Service: dynamically replicates, migrates, and deletes grid environment files according to parameter variations. It reduces execution times, promotes system stability, and improves storage device usage ratio efficiency.

## 3.2 ORCS and DMS operation

The DMS maintains replicas; the ORCS keeps file copies consistent. Figure 4 shows general DMS and ORCS operation. Prior to file maintenance, the Information Service and Replica Location Service store relevant information in the database for DMS measurement using the cost model described below. The Information Service and Replica Location Service functions are described below:

- Information Service [6]: periodically gathers statuses such as CPU idle ratio, memory usage, storage device free space, and network bandwidth, and records them in real time in the Information Database (Info. DB) for the DMS to use.
- Replica Location Service (RLS): stores file information such as logical file name, file size, file physical location, time of file creation or updating, and file access frequency in the File Information Database (File Info. DB). Users can use the Replica Location Service to search for desired files and the closest sites in the grid environment where the files are stored.

Before the Replica Manager triggers the ORCS and DMS, it first queries the Information Service and Replica Location Service, which then separately query the Information Database and File Information Database to get all file and system status information. If a Replica Manager determines some files need to be adjusted or kept consistent, it directs the ORCS and DMS to make the necessary adjustments. After all adjustments have been made, the ORCS and DMS query the Replica Location Service to check the new statuses of all files in the grid environment. After checking, the Replica Location Service records the new information in the File Information Database.

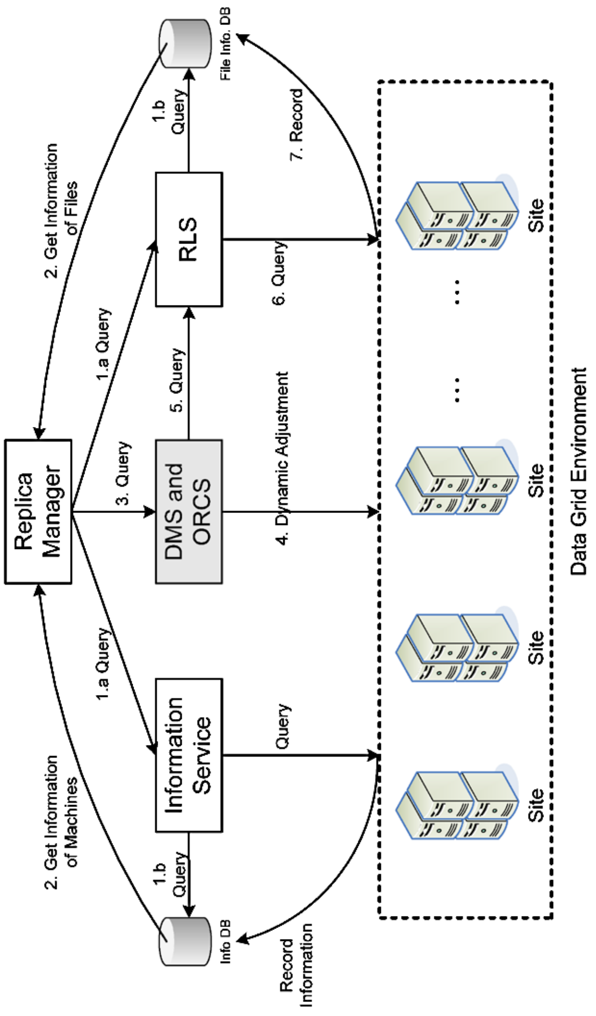


**Fig. 4** DMS and ORCS operations

## 3.3 Parameters and evaluation model

In this subsection we introduce our affect parameters, define measurable parameters, and present the evaluation models we use to measure the performance of the two services described above.

### *3.3.1 Affect parameters*

Because grid environments have many factors that affect performance, we calculated how the following static and dynamic factors affect overall performance.

- Static Parameters: These factors do not change when the grid environment changes. As Xuanhua Shi et al. indicated in [25], they include system site attributes such as CPU type and frequency, each storage element's hard disk capacity, memory capacity, and network card transfer rate. In general, faster frequency CPUs, larger memory and hard disk capacities, and network cards with faster transfer rates are better choices for executing jobs. Since these cannot be major factors in measuring grid environment performance due to the changeable nature of grid environments, we focus on the dynamic factors.
- Dynamic Parameters: These factors change when the grid environment changes. Job execution consumes computing power and uses memory space downloading or uploading data, and storing computational results. Thus, CPU usage rate, memory space, bandwidth, and node free space may all change. Among these, network bandwidth has the most important influence on performance. The NWS [18] monitors and periodically forecasts the performance of various network and computational elements. Real-time requirements must be met to achieve high performance. We use the NWS to measure network bandwidth, and the Linux commands "sar" and "df" to measure CPU, memory, and hard disk free space.

### *3.3.2 Cost model*

Before files are replicated, migrated, or deleted, their affect factors must be measured to determine what operations are necessary. Below, we define our strategic parameters.

- $BW_{LAN}(i-j)$: LAN connection bandwidth between node $i$ and node $j$ in Mbit
- $BW_{WAN}(i-j)$: WAN connection bandwidth between site $i$ and $j$ in Mbit
- $F\_size$: File size for transfer in MB
- $T\_trans(i-j)$: Time to transfer data from node $i$ to node $j$
- $T\_auth(i-j)$: Time for authenticating transfer of data file from node $i$ to node $j$
- $T\_replica\_local(i-j)$: Time for local file replication from node $i$ to node $j$
- $T\_replica\_remote(i-j)$: Time for remote file replication from node $i$ to node $j$
- $F\_space(i)$: Node $i$ storage device free space
- $FA\_Min$: Minimum file access rate
- $FA\_Max$: Maximum file access rate
- $\alpha$: Adjustable parameter for checking whether the storage element free space is sufficient for replication

- $P\_choice(i)$: Permission to transfer data to site $i$
- $D\_Replica(j)$: Check to determine whether the replica at node $j$ needs deleting
- $N\_AF$: Replica access frequency
- $NA$: Replica access time threshold
- $T$: Access frequency threshold.

Files in local and remote sites with access frequencies less than $FA\_Min$ will be deleted. The evaluation models are as follows: we assume that if file access frequency is more than $FA\_Max$ or between $FA\_Max$ and $FA\_Min$, the file should be replicated or migrated. It is very important to check for sufficient free space at the destination site before replicating or migrating. The measurement model for determining whether there is enough free space to store generated temporary data and job execution results is:

$$P\_choice(i) = \begin{cases} 0, & F\_space(i) \times \alpha < F\_size, \\ 1, & F\_space(i) \times \alpha \geq F\_size, \end{cases} \tag{1}$$

where $\alpha$ is an adjustable parameter. Files are then replicated or migrated to appropriate locations. We measure the time required for local and remote site transfers as follows.

- Local: For files replicated or migrated from node $i$ to node $j$ within the same site, the time required for the server node to transfer files to the client node is

$$T\_replica\_local(i-j) = T\_auth(i-j) + (F\_size \times 8)/BW_{LAN}(i-j). \tag{2}$$

- Remote: For files replicated or migrated from site $i$ to site $j$, the time cost is

$$T\_replica\_remote(i-j) = T\_auth(i-j) + (F\_size \times 8)/BW_{WAN}(i-j). \tag{3}$$

File size and network bandwidth are the most important factors in replicating or migrating files locally and remotely, and if $i = j$, we can assume that $BW_{LAN}(i-j)$ between site $i$ and site $j$ is $\infty$ and that $T\_auth(i-j)$ is zero.

In measuring storage capacity, $D\_Replica(i)$ determines whether the replica of node $i$ should be deleted or not. When $F\_space(j) \leq F\_size$ or $N\_AF < T$ is satisfied and the result of the $D\_Replica(j)$ is $True$, then the replica file will be deleted. A $D\_Replica(j)$ result of $False$ is opposite. The formula is as follows:

$$D\_Replica(j) = \begin{cases} True = \begin{cases} F\_space(j) \leq F\_size(i) \\ N\_AF < T \end{cases} \\ False = \begin{cases} F\_space(j) > F\_size(i) \\ N\_AF \geq T \end{cases} \end{cases}. \tag{4}$$

When a destination node's storage resource is not adequate for replication, then the next nearest node with sufficient free space and appropriate performance is chosen.

## 3.4 The DMS and ORCS algorithms

### *3.4.1 DMS algorithm*

The DMS algorithm shown in Fig. 5 consists of three parts, replication, migration, and deletion.

**Fig. 5** DMS replication algorithm

```
Check all file access rates
If (File $i'$s access rate is greater than FA_Max in site $j$) Then
{
        Check site $j$ storage device free space
        If (Not enough free space) Then
            Find an alternative site closest to site $j$
          If (The same file exists intra-region) Then
        Replicate file $i$ to site $j$ from intra-region file site
      Else
        Replicate file $i$ to site $j$ from best inter-region file site
    Else
          If (The same file exists intra-region) Then
            Replicate file $i$ to site $j$ from intra-region file site
      Else
        Replicate file $i$ to site $j$ from best inter-region file site
}

If (File $i$'s access rate is greater than FA_Min in site $j$) and
    (File $i$'s access rate is less than FA_Max in site $j$) Then
{
    Find the nearest file site $j$ that no longer needs file $i$
      Check site $j$ storage device free space
        If (Not enough free space) Then
          Find an alternative site closest to site $j$
      Migrate file $i$ to alternative site from file site
    Else
      Migrate file $i$ to site $j$ from file site
}

If (File $i$'s access rate is less than FA_Min in site $j$) Then
{
        Check the File Info. DB for another site with the same file
        If (A site with the file is found) Then
          Delete file $i$ in site $j$
        Else
          Keep file $i$
}
```

- *Replication*: If the access frequency for file $i$ at site $j$ exceeds the maximum access rate *FA_Max*, the DMS first checks to see if the storage device at site $j$ has enough free space to store the replicated file. If it does, the DMS duplicates the data to site $j$ using the intra-region copy of file $i$ if such a copy exists, or it creates a duplicate of file $i$ at site $j$ in the intra-region. If site $j$ does not have enough free space, the DMS first checks to see if it can duplicate file $i$ in the inter-region. If not, it stores the duplicated data in the site closest to site $j$.
- *Migration*: When an original file site no longer needs a file, or has insufficient free space to store duplicated data, temporary data, or computing results, but other sites still need the file, migration is used to move the file to an appropriate location. This avoids generation of excessive file copies in the data grid system and saves free space for storing temporary data and job execution results. If the request frequency of file $i$ in site $j$ is between *FA_Min* and *FA_Max*, the DMS first checks to see if

other sites need the file. If it is needed, the DMS finds a suitable site to transmit the file data to the destination site. If there is insufficient free space in the destination site's storage device, it migrates the file to the site nearest the destination site.

- *Deletion*: If file $i$'s access rate is less than *FA_Min* and another site has a copy of the file, the DMS deletes it; otherwise the DMS keeps it to ensure there is at least one copy in the grid environment.

The DMS algorithm increases storage device and file usage ratios. It dynamically maintains data in the grid environment and fixes the *BHR* algorithm problem described in [20]. More than two replicas in one region are allowed, thus users have more choices of sites for parallel file downloads. It also improves grid system performance by considering free space when storing computed results and temporary job data.

### *3.4.2 ORCS algorithm*

The ORCS algorithm maintains replica consistency with synchronous and asynchronous approaches. We assume only source files can be modified by grid users, and these files are then replicated to other nodes in the grid environment. All replicas in distributed nodes are read-only. Where a replica is stored in the data grid system and when it is replicated depend on grid user parameter settings and storage capacities.

Our replica distribution topology has three type nodes: super node (SN), master node (MN), and child node (CN). The data source saved in SN can only be added or modified by grid users, called original data. The original data were replicated from SN to MN automatically when added or modified by grid users, called master replica. The child replica replicated to CN depends on two factors: the access frequency of files and the storage capacity. The master replica and child replica are read only files. These files are one way replicated from SN to MN, from MN to CN.

The first function of the ORCS algorithm is shown in Fig. 6. When original data is updated, the super node (SN) immediately replicates the file to all master nodes (MNs). Original data may be modified by file owners and others with updating rights. The MN then checks the parameter $N\_AF$ for each grid site to determine whether to replicate the files from the MN to the CN.

The second section of the ORCS maintenance algorithm is shown in Fig. 7. When a user submits a request from a CN, the algorithm checks the file *N_AF* parameter. If the replica exists in the grid site node, its last update time is compared with that of the MN. If the replica needs updating, the algorithm checks storage capacity. If there is not enough free space in the storage device, the old replicated file in the CN is deleted and a new replica is copied from the MN to the CN nearest the previous CN with the best resource status. Finally, replica-update records are added to the database for later tracing.

Figure 8 shows original data automatically replicated from an SN to an MN after an addition or modification by a grid user. When the replication is complete, each grid site MN will check whether the replica's access frequency is greater than its threshold value and the site's storage capacity is sufficient, as shown in Fig. 9. The red line in Fig. 10 indicates the CN does not have adequate storage capacity, which means the

```
// Once Original Data has been updated
If original data is updated from a super node then
    Copy the original data to all master nodes
    Add update records to the replication database for tracing
End
// For each Grid Site
If a replica's access frequency by CN to MN is greater than its threshold then
    If the CN has sufficient storage capacity then
        Copy a replica from MN to CN
        Add a replica update record to the database
    Else
        Find all replicas with access frequencies smaller than the CN threshold
            Sort these CN replica access frequencies in ascending order
        Delete replicas one by one from small to large until CN has sufficient storage capacity
        If the storage capacity of CN is sufficient then
            Copy a replica from MN to CN
            Add a replica update record to the database
        Else
            Copy a replica from MN to a CN with the best resource status
            Add a replica update record to the database
        End
    End
End
```

**Fig. 6** First section of the ORCS algorithm

algorithm must find the nearest CN with the best resource status that has sufficient storage capacity, as shown in Fig. 11.

In Fig. 12, Node $j$ replica's access frequency is lower than its threshold value, thus Node $j$ accesses the MN replica. In contrast, in Fig. 13, grid users access Node $j$ directly if the last update time of the CN replica is the same as that of the MN replica. If the latest update times are not equal, the replica will be copied automatically from the MN to the CN, as shown in Fig. 14. Figures 15 and 16 show the algorithm finding the nearest CN with the best resource status and sufficient storage capacity because node $j$ has insufficient storage capacity.

## 4 Experimental environment and results

We compared and evaluated the performance of the DMS and ORCS algorithms against other strategies. The Least Frequently Used (LFU), Least Recently Used (LRU) strategies, and the Bandwidth Hierarchy-based Replication algorithm (BHR) were tested against the DMS algorithm. The LFU and LRU always replicate when requests occur, but choose files for deletion differently when storage element free space is insufficient for replication. LRU chooses the oldest files for deletion, while LFU chooses the least frequently requested files. The synchronous and asynchronous consistency strategies were tested against the ORCS. We used a simulator called OptorSim, developed by the EU Data Grid [3], to compare the strategies mentioned above. Our experimental grid environment is shown in Fig. 17. It consisted of four

```
// For each access to the MN by a Grid Site CN
If a replica's access frequency by a CN to the MN is smaller than its threshold then
    Change the access directory to MN.Replica
Else
    If the CN.Replica.LastUpdateTime = the MN.Replica.LastUpdateTime then
        Change the access directory to CN.Replica
    Else
        If the CN has sufficient storage capacity then
            Copy a replica from the MN to the CN
            Add a replica update record to the database
        Else
            Find all replicas in the CN with access frequencies smaller than the threshold
                Sort these replica access frequencies in ascending order
            Delete replicas one by one from small to large until the CN storage capacity is sufficient
            If the CN storage capacity is sufficient then
                Copy a replica from the MN to the CN
                Add a replica update record to the database
            Else
                Find the node with the best resource status and sufficient storage capacity nearest the CN
                If this node is the MN then
                    Direct-access the replica to the MN
                Else
                    Copy a replica from the MN to the CN
                    Add a replica update record to the database
                End
            End
        End
    End
End
```

**Fig. 7** Second section of the ORCS algorithm

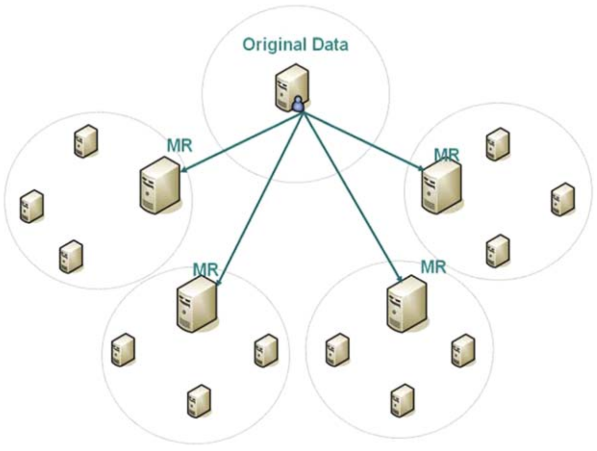


**Fig. 8** Operation 1 of the first section

regions, each containing 8 sites. Initially, all files were randomly stored, and sites in the four regions then requested files from appropriate sources.

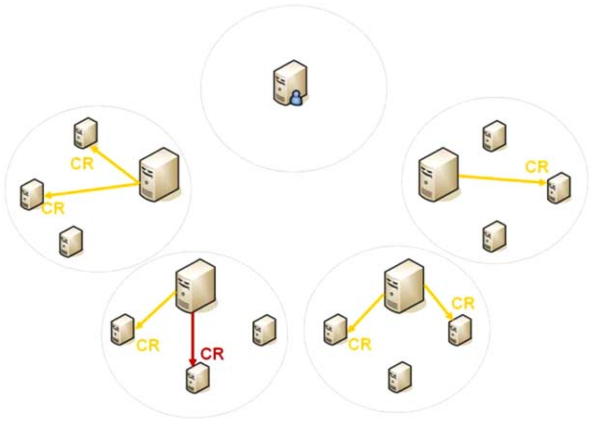


**Fig. 9** Operation 2 of the first section

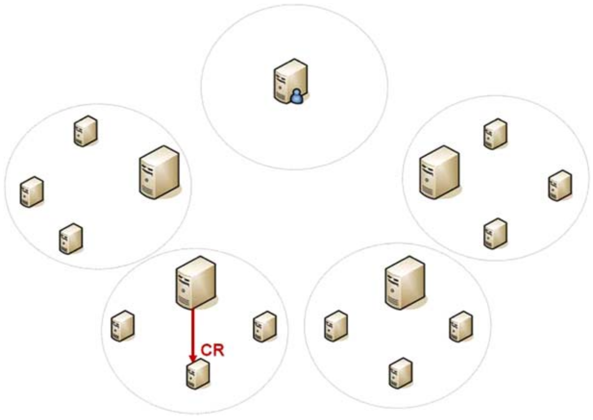


**Fig. 10** Operation 3 of the first section

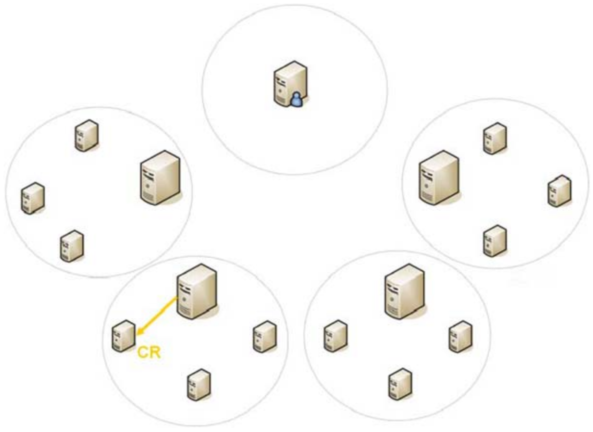


**Fig. 11** Operation 4 of the first section

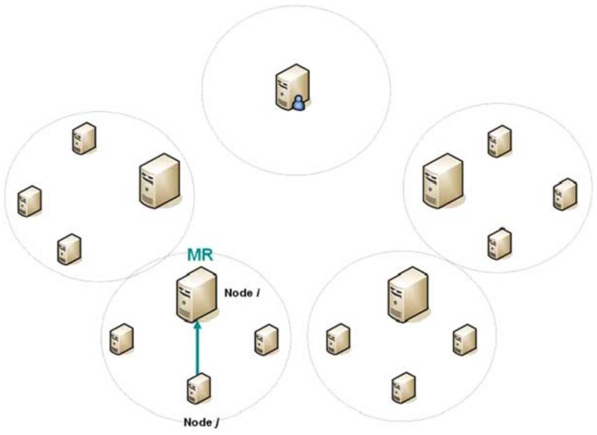


**Fig. 12** Operation 1 of the second section

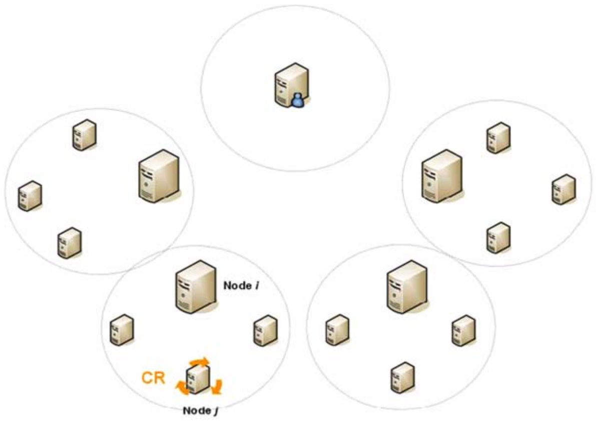


**Fig. 13** Operation 2 of the second section

## 4.1 Parameter setting

### *4.1.1 DMS parameter setting*

As Table 1 shows, we assumed 500 total jobs, and each site in the four regions sent requests to file sites at random. There were 30 job types, each job requiring accessing 15 files when executed. Files were 250, 500, 750, and 1000 MB in size; see Table 2 for the quantities of each file size. Each site's hard disk had 50 GB of free space. Intra-region bandwidth was 500 Mbps, and inter-region bandwidth was 250 Mbps. The *FA_Max* in our simulation was set to 10, and the *FA_Min* was set to 5. The job delay time was 2500 milliseconds. The DMS could perform migration and deletion operations up to the total job size and the total hard disk free space in each region. Also, we assumed temporary data and results would be produced during job execution. Before comparing the DMS with other replication strategies, values had to be assigned to the important factors. We evaluated $\alpha$ for the grid environment shown in Fig. 17 with the parameters in Table 1.

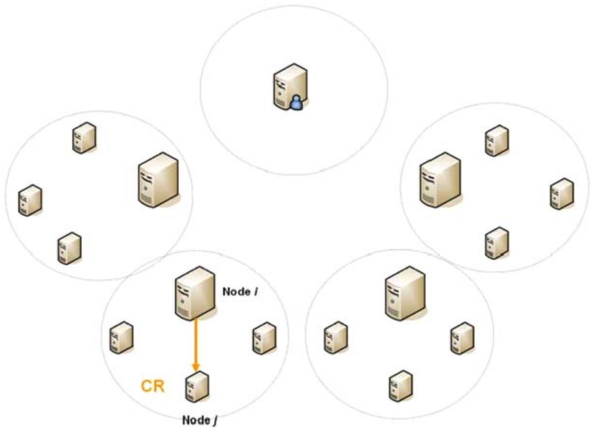


**Fig. 14** Operation 3 of the second section

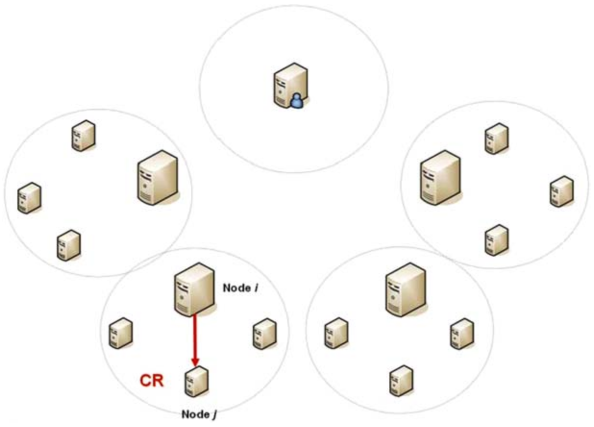


**Fig. 15** Operation 4 of the second section

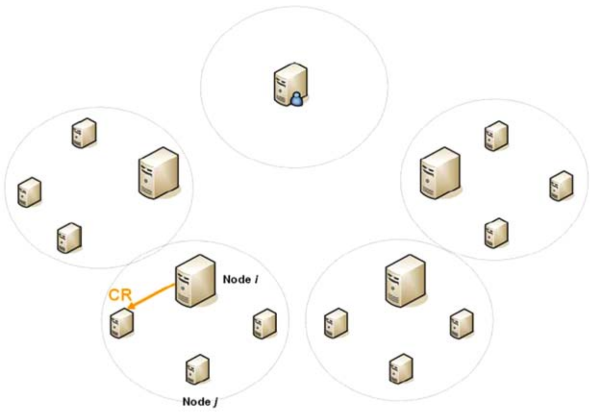


**Fig. 16** Operation 5 of the second section

The evaluation results are shown in Figs. 18 and 19. Figure 18 shows the execution time for 500 jobs was best when $\alpha$ was set to 0.9, and Fig. 19 shows the storage

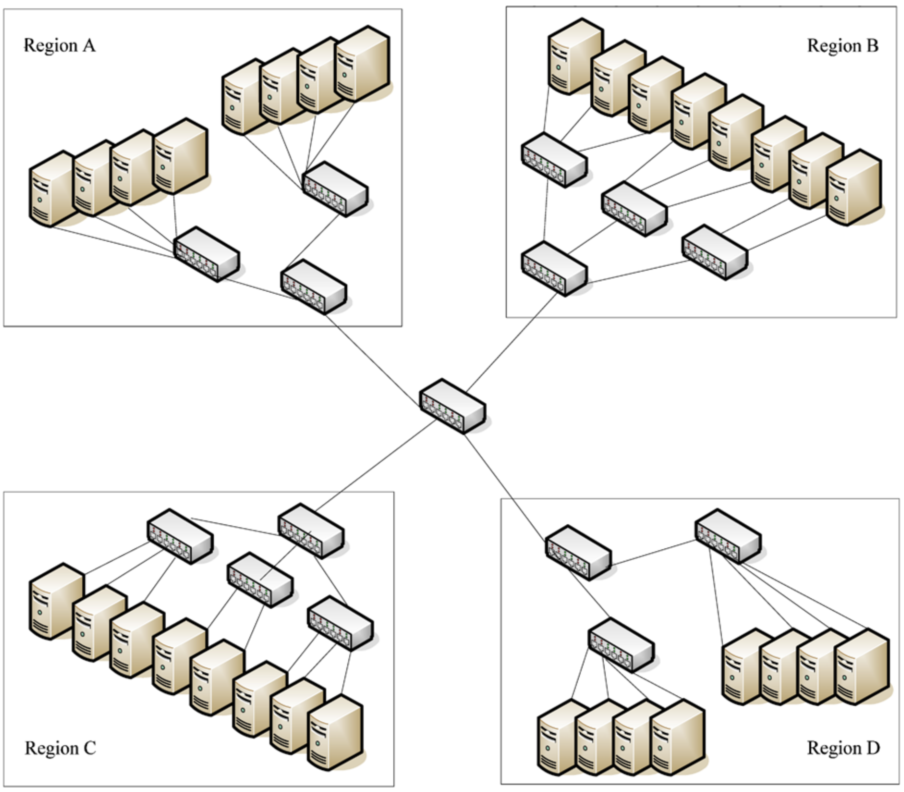
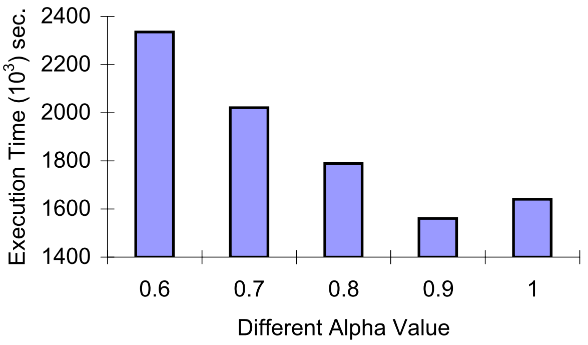


**Fig. 17** The experimental grid environment

**Fig. 18** Execution times for various alpha values



element usage rate was better when $\alpha$ was set to 1. But setting $\alpha$ to 1 led to greater execution times than setting $\alpha$ to 0.9 because when there was not enough free space to store temporary data and results produced during job execution, time was consumed finding other computing elements to continue execution. Even though the storage element usage ratio was better when $\alpha$ was set to 1 than when $\alpha$ was set to 0, overall performance was better when $\alpha$ was set to 0.9 than when $\alpha$ was set to 1. Thus, we set $\alpha$ to 0.9 for our experiments.

**Table 1** Parameters used in the simulation

| Parameters | Values |
|---|---|
| Number of jobs | 500 |
| Number of job types | 30 |
| Number of files accessed per job | 15 |
| File sizes | 250/500/750/1000 MB |
| Intra-region bandwidth | 500 Mbps |
| Inter-region bandwidth | 250 Mbps |
| Hard disk space at each site | 50 GB |
| FA_Max | 10 |
| FA_Min | 5 |
| Job Delay | 2500 ms |

**Table 2** Numbers of files in each size

| File size | Number |
|---|---|
| 250 MB | 175 |
| 500 MB | 90 |
| 750 MB | 100 |
| 1000 MB | 85 |

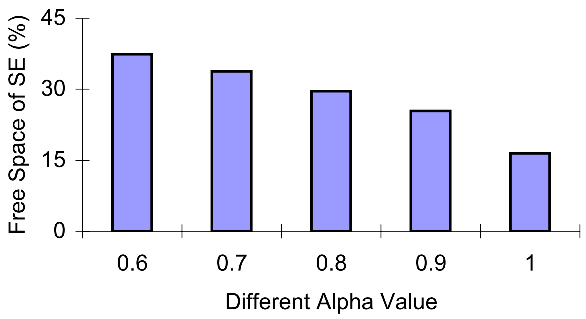


**Fig. 19** Free space for various alpha values

### *4.1.2 ORCS parameter setting*

In Table 3, we define several parameters used to derive the experimental results shown below. We submitted 100 writing jobs from the SN and 1000 read jobs from each CN. Files were 100 MB in size and the access threshold was 10. We used synchronous replication, asynchronous replication, and the ORCS algorithms in our experimental environment.

## 4.2 Results

### *4.2.1 File management results*

Figures 20 and 21 show, respectively, DMS execution times and storage element free space with and without the migration mechanism. Experimental results show execution times were better with the migration mechanism than without the migration

**Table 3** Experimental parameters

| Parameter | Value |
|---|---|
| Number of write jobs | 100 |
| Number of read jobs | 1000 |
| Various file sizes (MB) | 5, 50, 100, 1000, 2000 |
| Access threshold | 10, 20, 30, 40, 50 |
| Replication frequency | 200 |

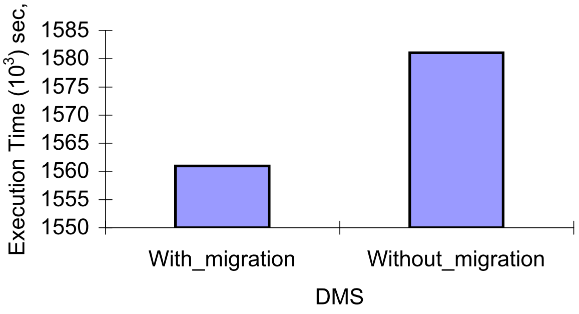


**Fig. 20** DMS performance comparison with/without migration

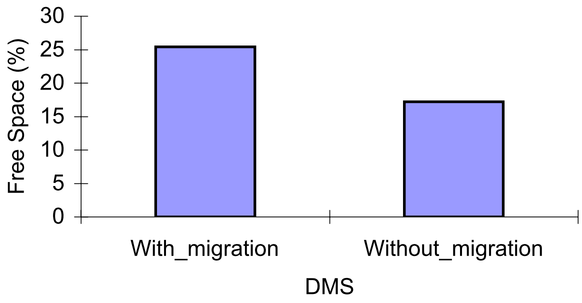


**Fig. 21** DMS free space comparison with/without migration

mechanism. But the storage element usage ratio results show the opposite because when the migration mechanism is used, files are adjusted to appropriate locations for job use, and numbers of copies will reflect job needs. Storage elements will need more free space to store temporary data and job results. Not using the migration mechanism results in more replicas being generated than when the migration mechanism is used, causing unnecessary waste of storage element free space. Thus, the probability is higher that storage elements will not have enough space to store temporary files and job results than when the migration mechanism is used.

We used the assumptions explained above to compare and evaluate the performance of the DMS algorithm against three other replication strategies. Figures 22 and 23 show, respectively, the results of comparing the DMS strategy execution time and free space performance with migration mechanism to the LFU, LRU, and BHR replication strategies. Figure 22 shows the DMS had better execution times than the other strategies. LFU and LRU always replicate when file accesses occur, which consumes a lot of free space. Although the LRU and LFU usage ratios for storing tem-

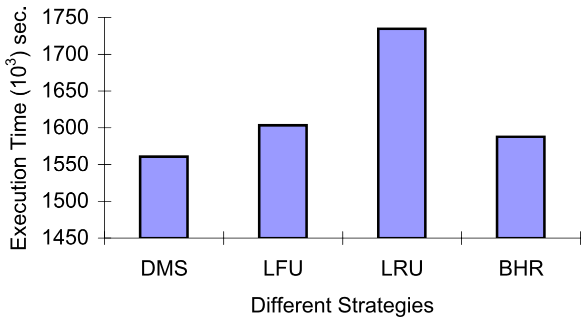


**Fig. 22** Performance comparison of four strategies

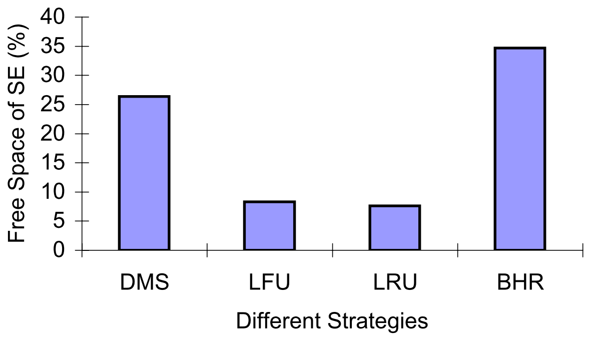


**Fig. 23** Free space comparison of four strategies

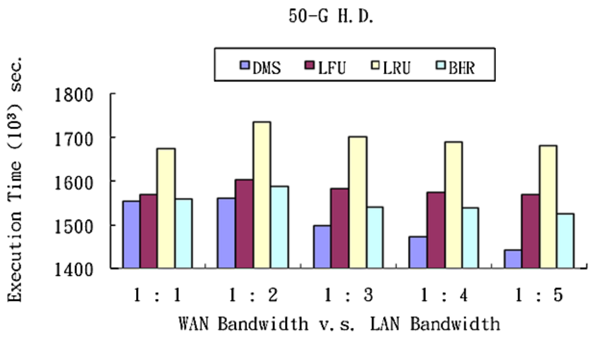


**Fig. 24** Execution time comparison for various bandwidth ratios with 50 G H.D.

porary data and job results were better than those of the DMS, choosing files for deletion and downloading files for job execution took considerable time. The BHR strategy caused jobs to spend excessive time getting needed files. Although the BHR strategy saved a lot of free space, its execution times were greater than those of the DMS strategy. Thus, the DMS performed more efficiently than other three replication strategies.

Figures 24 to 28 show the use of various network bandwidths and storage element capacities to demonstrate variations in the four strategies' performance. For all variations in hard disk size, LFU and LRU performed better when WAN bandwidth equaled LAN bandwidth. When there was not enough space to store replicas, temporary files, and job results, computing elements spent less time getting relevant files

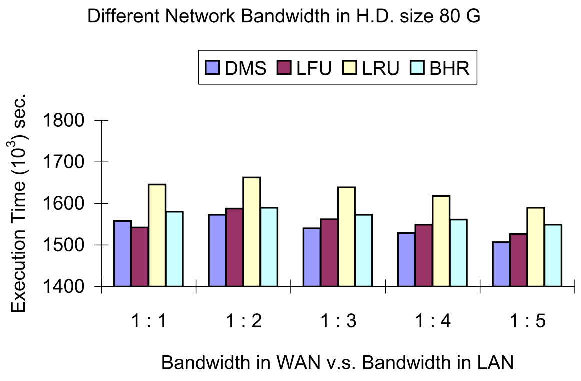


**Fig. 25** Execution time comparison for various bandwidth ratios with 80 G H.D.

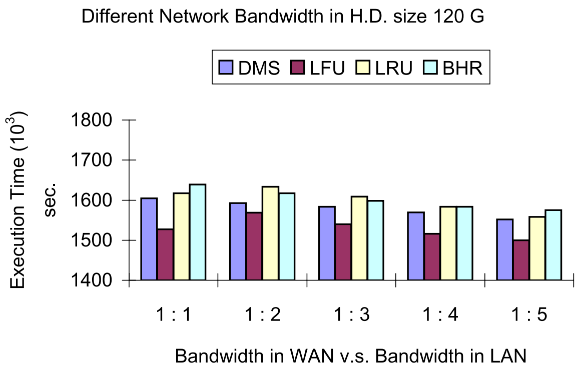


**Fig. 26** Execution time comparison for various bandwidth ratios with 120 G H.D.

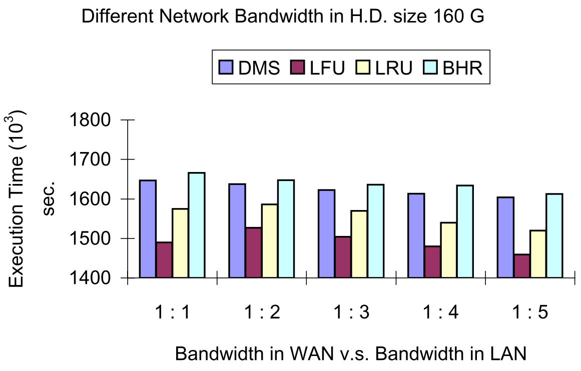
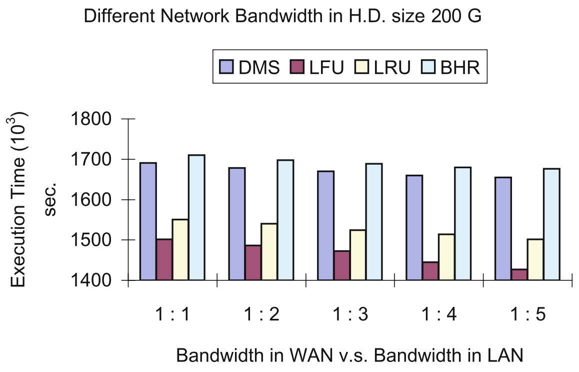


**Fig. 27** Execution time comparison for various bandwidth ratios with 160 G H.D.

from other storage elements or choosing other computing elements to continue job execution, shortening total execution times. When WAN bandwidth was smaller than LAN bandwidth, computing elements spent less time downloading and transferring files to appropriate LAN computing elements, but if the files computing elements needed were stored in inter-region storage elements, much time was wasted replicating and transferring files via the WAN, increasing total execution times. Total exe-

**Fig. 28** Execution time comparison for various bandwidth ratios with 200 G H.D.



cution times decreased gradually as the ratio of WAN bandwidth to LAN bandwidth was increased.

The DMS replication strategy performed better than other replication strategies when storage element capacity was small because it provided more storage element free space for storing replicas, temporary files, and job results. But increasing the storage element capacity sufficiently provided enough free space for the LFU and LRU replication strategies to store replicas, temporary files, and job results, reducing the time computing elements needed to get required files from other storage elements in the LAN and even in the WAN, thus shortening total execution times. And increasing storage element capacity also increased DMS replication strategy execution times. This means that as capacity was increased, DMS replication strategy performance worsened due to inefficient storage element usage ratios, and that the DMS was more effective when the storage element capacity was small.

### *4.2.2 File consistency results*

There were three replication algorithms in our experiment, Synchronous, Asynchronous, and our ORCS algorithm. We assumed that each file was 100 MB and the access frequency threshold was 10. When a replica in the CN was the same as the one in the MN, no transfer was necessary. The experimental file replication times are shown in Fig. 29. The ORCS replicated files according to grid users' needs. It consumed less network bandwidth than the Synchronous algorithm and accessed files more efficiently than the Asynchronous algorithm.

In the next experiment, we compared the storage capacity usage of our ORCS with that of the Massive Data Oriented Replication Algorithms (MDORA) proposed by Changqin Huang et al. in [14]. We assumed CN storage capacities of 80 G, a replication frequency of 200, and file sizes of 5 MB, 50 MB, 100 MB, 1000 MB, and 2000 MB. The result is shown in Fig. 30. If the CN storage capacity is inadequate, MDORA cannot store replicas. ORCS deletes files in ascending order of access frequency until there is adequate storage space for replicas. Thus, replicas can be written, even though the storage space was initially inadequate.

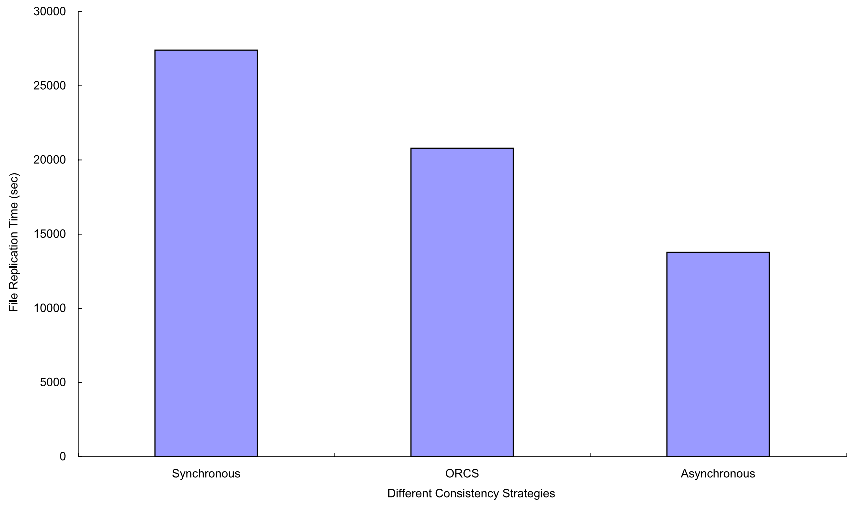


**Fig. 29** Comparison of three replication algorithms

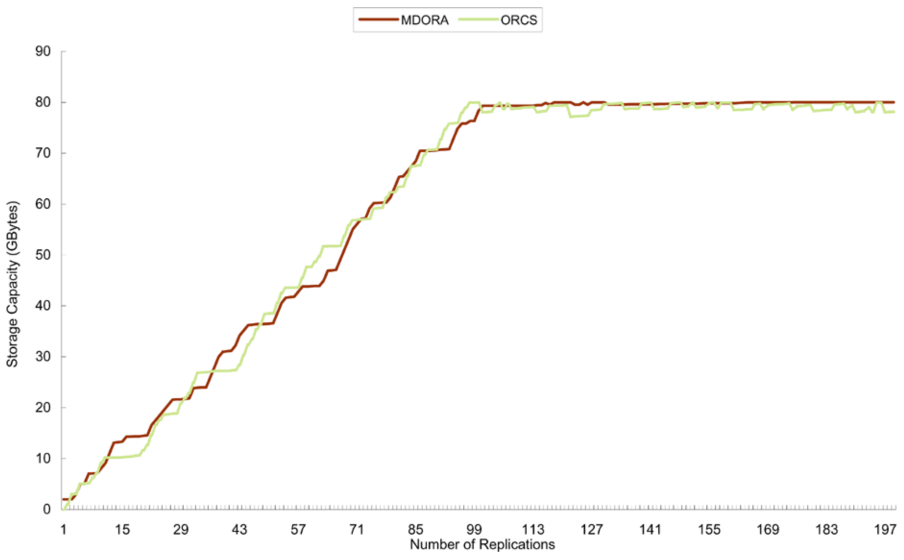


**Fig. 30** Storage capacity usage

There were 1400 read jobs in our experiment, each file was 100 MB in size, and threshold values of 10, 15, 20, 25, and 30 were used. In Figs. 31 and 32, we note that the replication time costs increased as the threshold value was increased, resulting in fewer and fewer replications. Therefore, the smaller the threshold value, the higher the data access availability.

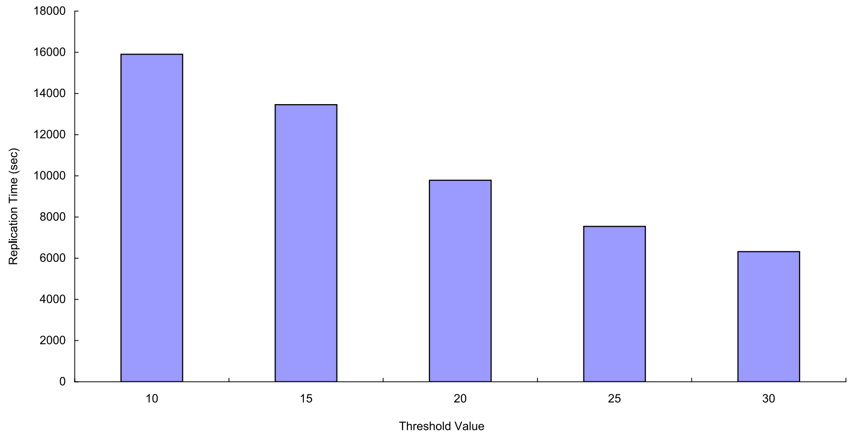


**Fig. 31** Replication times for various threshold values

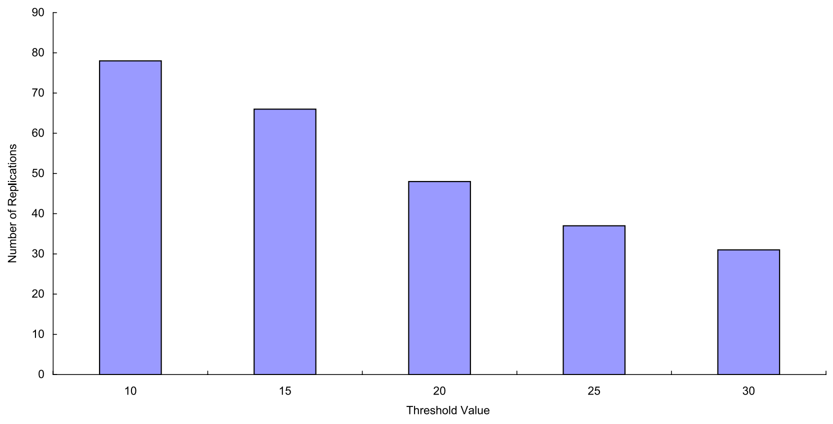


**Fig. 32** Numbers of replications for various threshold values

## 5 Conclusions and future work

This paper presents the Dynamic Maintenance Service (DMS) and the One-way Replica Consistency Service (ORCS) for improving grid environment performance. DMS is also aimed at the "one-replica" problem the BHR incurs. It improves grid system performance and increases storage element usage ratio efficiency by handling temporary data and results that jobs produce during execution. Via ORCS, we addressed the principal problems with maintaining consistency among existing replicas. Experimental results show that DMS and ORCS both perform more efficiently than other strategies and make storage element usage ratios more efficient as well. We conducted the design and implementation of a data grid system using the components and services proposed in Sect. 3 to enable general users to use data grid systems and monitor details of grid resource and file statuses.

Our future work will entail enhancing the accuracy of the DMS and ORCS evaluation models for various applications and situations. The DMS and ORCS algorithms need improved fault tolerance and adaptability to render them better able to handle various challenges than other strategies. We are also considering development of a simulator based on real-world grid topology to determine which applications these two strategies are best suited to, after which we can combine the data grid system with a good job scheduling strategy and develop applications that require grid technology.

**Acknowledgements** This work is supported in part by the National Science Council, Taiwan, ROC, under Grant nos. NSC 96-2221-E-029-019-MY3, NSC 97-2622-E-029-003-CC2, and NSC 97-3114-E-007-001-.

## References

1. Allcock B, Bester J, Bresnahan J, Chervenak A, Foster I, Kesselman C, Meder S, Nefedova V, Quesnel D, Tuecke S (2002) Data management and transfer in high-performance computational grid environments. Parallel Comput 28(5):749–771
2. Allcock B, Bester J, Bresnahan J, Chervenak A, Foster I, Kesselman C, Meder S, Nefedova V, Quesnel D, Tuecke S (2001) Secure, efficient data transport and replica management for high-performance data-intensive computing. In: Proceedings of the eighteenth IEEE symposium on mass storage systems and technologies, pp 13–28
3. CERN. http://public.web.cern.ch/Public/Welcome.html
4. Chang RS, Chang JS (2006) Adaptable replica consistency service for data grids. In: Proceeding of the third international conference of information technology (ITNG'06), pp 646–651
5. Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S (2001) The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. J Netw Comput Appl 23:187–200
6. Czajkowski K, Fitzgerald S, Foster I, Kesselman C (2001) Grid information services for distributed resource sharing, In: Proceedings of the tenth IEEE international symposium on high-performance distributed computing (HPDC-10'01), August 2001, pp 181–194
7. Domenici A, Donno F, Pucciani G, Stockinger H, Stockinger K (2004) Replica consistency in a data grid. Nucl Instr Methods Phys Res A 534(1–2):24–28
8. Düllmann D, Hoschek W, Martinez JJ, Segal B (2001) Models for replica synchronisation and consistency in a data grid. In: Proceedings of the 10th IEEE international symposium on high performance distributed computing (HPDC-10'01), October 2001, pp 67
9. Fathali J (2006, to appear) A genetic algorithm for the p-median problem with pos/neg weights. Appl Math Comput 8 (August)
10. Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Manag Sci 27:1–18
11. Foster I (2002) The grid: a new infrastructure for 21st century science. Phys Today 55(2):42–47
12. Foster I, Kesselman C (1999) The grid 2: blueprint for a new computing infrastructure, 2nd edn. Morgan Kaufmann, San Mateo (Elsevier series in grid computing)
13. Foster I, Kesselman C (1997) Globus: a metacomputing infrastructure toolkit. Int J Supercomput Appl High Perform Comput 11(2):115–128
14. Huang CQ, Xu FY, Hu XY (2006) Massive data oriented replication algorithms for consistency maintenance in data grids. ICCS 2006, Part I, LNCS 3991, pp 838–841
15. Hu JF, Xiao N, Zhao YJ, Fu W (2005) An asynchronous replica consistency model in data grid. In: Parallel and distributed processing and applications (ISPA 2005 workshops), pp 475–484
16. Jackson LE, Rouskas GN, Stallmann MFM (2007) The directional p-median problem: definition, complexity, and algorithms. Eur J Oper Res 179:1097–1108. http://people.engr.ncsu.edu/mfms/Publications/2007-EJOR-Jackson.pdf
17. Java CoG. http://www-unix.globus.org/cog/
18. NWS. http://nws.cs.ucsb.edu/

19. OptorSim. http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html
20. Park SM, Kim JH, Ko YB, Yoon W-S (2003) Dynamic data grid replication strategy based on Internet hierarchy. In: The second international workshop on grid and cooperative computing (GCC2003), pp 838–846
21. Park SM, Kim JH (2003) Chameleon: a resource scheduler in a data grid environment. In: Proceedings of third international symposium on cluster computing and the grid, p. 258. http://portal.acm.org/citation.cfm?id=792481
22. Rahman RM, Barker K, Alhajj R (2006) Replica placement design with static optimality and dynamic maintainability. In: Proceedings of the sixth IEEE international symposium on cluster computing and the grid (CCGRID'06), pp 434–437
23. Rahman RM, Barker K, Alhajj R (2006) Effective dynamic replica maintenance algorithm for the grid environment. In: Proceedings of advances in grid and pervasive computing, vol 3947: Grid and pervasive computing 2006 (GPC2006), pp 336–345
24. Ranganathan K, Foster I Design and evaluation of dynamic replication strategies for a high performance data grid. In: Proceedings of international conference on computing in high energy and nuclear physics
25. Shi XH, Jin H, Qiang WZ, Zou DQ (2003) An adaptive meta-scheduler for data-intensive applications. In: Proceedings of grid and cooperative computing (GCC'03), pp 830–837
26. Stockinger H, Samar A, Allcock B, Foster I, Holtman K, Tierney B (2002) File and object replication in data grids. J Cluster Comput 5(3):305–314
27. The Globus Alliance. http://www.globus.org/
28. Vazhkudai S, Tuecke S, Foster I (2001) Replica selection in the globus data grid. In: Proceedings of the 1st international symposium on cluster computing and the grid (CCGRID 2001), pp 106–113
29. Venugopal S, Buyya R, Ramamohanarao K (2006) A taxonomy of data grids for distributed data sharing, management, and processing. ACM computing surveys, vol 38, Article 3, March 2006
30. Yang CT, Yang IH, Li KC, Wang SY (2007) Improvements on dynamic adjustment mechanism in co-allocation data grid environments. J Supercomput 40(3):269–280
31. Yang CT, Wang SY, Fu CP (2007) A dynamic adjustment mechanism for data transfer in data grids. In: Network and parallel computing: IFIP international conference, NPC 2007. Lecture notes in computer science, vol 4672. Springer, Berlin, pp 61–70. ISSN 1611-3349
32. Yang CT, Yang MF, Chiang WC (2008) Implementation of a cyber transformer for parallel download in co-allocation data grid environments. In: Proceedings of the 7th international conference on grid and cooperative computing (GCC2008) and second EchoGRID conference, October 24–26, 2008 in Shenzhen, Guangdong, China, pp 242–253
33. Yang CT, Yang IH, Chen CH, Wang SY (2006) Implementation of a dynamic adjustment mechanism with efficient replica selection in co-allocation data grid environments. In: Proceedings of the 21st annual ACM symposium on applied computing (SAC 2006) – distributed systems and grid computing (DSGC) track, vol 1, pp 797–804, Dijon, France, April 23–27, 2006
34. Yang CT, Yang IH, Wang SY, Li KC, Hsu CH (2009) A recursively-adjusting co-allocation scheme with cyber-transformer in data grids. Future Gener Comput Syst 25(7):695–703
35. Ganglia. http://ganglia.info/

**Chao-Tung Yang** is Professor of Computer Science at Tunghai University in Taiwan. He was born on November 9, 1968 in Ilan, Taiwan, R.O.C. and received his B.Sc. degree in Computer Science from Tunghai University, Taichung, Taiwan, in 1990, and the M.Sc. degree in Computer Science from National Chiao Tung University, Hsinchu, Taiwan, in 1992. He received the Ph.D. degree in Computer Science from National Chiao Tung University in July 1996. He won the 1996 Acer Dragon Award for an outstanding Ph.D. dissertation. He has worked as Associate Researcher for ground operations in the ROCSAT Ground System Section (RGS) of the National Space Program Office (NSPO) in Hsinchu Science-based Industrial Park since 1996. In August 2001, he joined the faculty of the Department of Computer Science at Tunghai University. He got the excellent research award by Tunghai University in 2007. In 2007 and 2008, he got the Golden Penguin Award by Industrial Development Bureau, Ministry of Economic Affairs, Taiwan. His researches have been sponsored by Taiwan agencies National Science Council (NSC), National Center for High Performance Computing (NCHC), and Ministry of Education. His present research interests are in grid and cluster computing, parallel and multicore computing, and web-based applications. He is a member of both the IEEE Computer Society and ACM.

**Chun-Pin Fu** was born on August 3, 1982, in Tainan City, Taiwan, R.O.C. He received the B.Sc. degree in Department of Computer Science and Information Engineering from Leader University in 2005, and he also received M.Sc. degree in Department of Computer Science from Tunghai University in 2007, respectively. He is working at Household Registration Office, North District in Tainan City now. His present research interests are in grid and cluster computing and web-based applications.

**Ching-Hsien Hsu** received his B.Sc. degree in Computer Science from Tunghai University in 1995, and the Ph.D. degree in Information Engineering and Computer Science from Feng Chia University in 1999, respectively. From 2001 to 2002, Dr. Hsu had been Assistant Professor in the Department of Electrical Engineering at Nan Kai College. He joined the department of Computer Science and Information Engineering, Chung Hua University, in 2002, and has become Associate Professor since August 2005. He was awarded as annual outstanding researcher by Chung Hua University in 2005, 2006 and 2007 and got the excellent research award in 2008. Doctor Hsu has published more than 100 academic papers in journals, books and conference proceedings. Doctor Hsu is serving in a number of journal editorial boards, including International Journal of Communication Systems, International Journal of Computer Science, International Journal of Grid and High Performance Computing, International Journal of Smart Home, and International Journal of Multimedia and Ubiquitous Engineering. He has edited more than 10 international journal special issues as a guest editor, such as IEEE Transactions on Services Computing, Future Generation Computer Systems, Journal of Supercomputing, etc. Doctor Hsu's research interest is primarily in parallel and distributed computing, grid computing, P2P computing, RFID and services computing. Doctor Hsu is currently an IEEE senior member and serves as an executive committee of IEEE Technical Committee on Scalable Computing (TCSC).

J Supercomput
DOI 10.1007/s11227-009-0307-4

# Implementation of a dynamic adjustment strategy for parallel file transfer in co-allocation data grids

**Chao-Tung Yang · Shih-Yu Wang · William Cheng-Chung Chu**

**Abstract** Co-allocation architecture was developed to enable parallel transferring of files from multiple replicas stored in the different servers. Several co-allocation strategies have been coupled and used to exploit the different transfer rates among various client-server links and to address dynamic rate fluctuations by dividing files into multiple blocks of equal sizes. The paper presents a dynamic file transfer scheme, called dynamic adjustment strategy (DAS), for co-allocation architecture in concurrently transferring a file from multiple replicas stored in multiple servers within a data grid. The scheme overcomes the obstacle of transfer performance due to idle waiting time of faster servers in co-allocation based file transfers and, therefore, provides reduced file transfer time. A tool with user friendly interface that can be used to manage replicas and downloading in a data grid environment is also described. Experimental results show that our DAS can obtain high-performance file transfer speed and reduce the time cost of reassembling data blocks.

**Keywords** Data grid · File replica · Parallel file transfer · Co-allocation · Dynamic adjustment

C.-T. Yang (✉) · S.-Y. Wang · W.C.-C. Chu
Department of Computer Science, Tunghai University, Taichung, 40704, Taiwan
e-mail: ctyang@thu.edu.tw

W.C.-C. Chu
e-mail: cchu@thu.edu.tw

S.-Y. Wang
Communication System Division Service-Oriented Network System Department, Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, 31040, Taiwan
e-mail: Laurence@itri.org.tw

Published online: 17 June 2009

## 1 Introduction

Data grid traditionally represents the network of distributed storage resources from archival systems to caches and databases, which are linked using a logical name space to create global, persistent identifiers and provide uniform access mechanisms. Data grid aggregates distributed resources to resolve large-size dataset management problems [1–6, 8, 10, 11, 14, 17, 21, 22]. Increasingly, large collections of measured and computed data are emerging as important resources in many data-intensive applications. Certain data-intensive scientific applications, such as high-energy physics, bioinformatics applications, and virtual astrophysical observations, entail huge amounts of data that require data file management systems to replicate files and manage data transfers and distributed data access. In physics experiments, for example, data file sizes can range from 2 to 10 gigabytes. These high-performance and data-intensive computing applications require efficient management and transfer of terabytes or petabytes of information in wide-area, distributed-resource environments. Data grid infrastructure integrates data storage devices and data management services in grid environments consisting of scattered computing and storage resources, perhaps located in different countries/regions yet accessible to users [11, 14, 17–19].

Replicating popular content in distributed servers is widely used in practice [14, 17, 21, 22]. Recently, large-scale, data-sharing scientific communities such as those described in [3, 4] used this technology to replicate their large datasets over several sites. Downloading large datasets from several replica locations may result in varied performance rates, because the replica sites may have different architectures, system loadings, and network connectivity. Bandwidth quality is the most important factor affecting transfers between clients and servers since download speeds are limited by the bandwidth traffic in the links connecting the servers to the clients.

One way to improve download speed is to use replica selection techniques to determine the best replica locations [21]. The replica selection algorithms may aim at maximizing network throughput, reducing load on “expensive” links or reducing the response time perceived by the user. Most replica selection algorithms aim at selection of “nearby” replicas to either reduce response time or the load on network links. This method selects the servers most likely to provide optimum transfer rates because bandwidth quality can vary unpredictably due to the sharing nature of the Internet. Another way is to use co-allocation technology [17, 21, 22, 27, 28] to download data. Co-allocation of data transfers enables the clients to download data from multiple locations by establishing multiple connections in parallel. This can improve the performance over single-server downloads and alleviate the Internet congestion problem [17].

Several co-allocation strategies were presented in our previous work [25–27]. An idle-time drawback remains since faster servers must wait for the slowest server to deliver its final block. Thus, reducing the differences in finish times among replica servers is important. In this paper, we propose a dynamic file-transfer scheme with co-allocation architecture, called the *dynamic adjustment strategy* (DAS), which reduces file-transfer times and also improves data transfer performance in data grid environments. Our approach can reduce file server idle times and decrease file-transfer completion times. We also present a new toolkit, called cyber-transformer, with a friendly client-side GUI interface integrated with the information service, replica location service, and data transfer service [25] that makes it easy for inexperienced users to manage replicas and download files in data grid environments. And we provide an effective scheme for reducing the cost of reassembling data blocks. Experimental results show that our approach is superior to previous methods and achieves the best overall performance. We also discuss combination cost and provide an effective improvement.

The remainder of this paper is organized as follows. Related background review and studies are presented in Sect. 2 and the co-allocation architecture and our research approaches are outlined in Sect. 3. In Sect. 4, a powerful toolkit, cyber-transformer, is proposed by us, and experimental results and a performance evaluation of our scheme are presented in Sect. 5. Section 6 concludes this research paper.

## 2 Background review

### 2.1 Data grid and replications

In data grid environments, access to distributed data is typically as important as access to distributed computational resources [1–5, 7, 22]. Distributed scientific and engineering applications require transfers of large amounts of data between storage systems, and access to large amounts of data generated by many geographically distributed applications and users for analysis and visualization, among others.

We used the grid middleware Globus Toolkit [9, 10, 12, 13, 16] as our data grid infrastructure. The Globus Toolkit provides solutions for such considerations as security, resource management, data management, and information services. One of its primary components, MDS [7, 13, 16], is designed to provide a standard mechanism for discovering and publishing resource status and configuration information. It provides a uniform and flexible interface for data collected by lower-level information providers in two modes: static (e.g., OS, CPU types, and system architectures) and dynamic data (e.g., disk availability, memory availability, and loading). And it uses GridFTP [3, 13, 16] to provide efficient management and transfer data in a wide-area, distributed-resource environment. We use GridFTP to enable parallel data transfers. Among its many features, its partial file transfer ability allows files to be retrieved from data servers by specifying the start and end offsets of file partitions. This protocol, which extends the standard FTP protocol, provides a superset of the features offered by the various grid storage systems currently in use.

The data grid community tries to develop secure, efficient data transport mechanisms and replica management services. Another key technology from the Globus project, called the Replica Catalog [16], is used to register and manage complete and partial copies of data sets. The Replica Catalog contains mapping information from a logical file or collection to one or more physical files.

Replica management involves creating or removing replicas at a data grid site [21]. A replica manager typically maintains a replica catalog containing replica site addresses and file instances. The replica management service is responsible for managing the replication of complete and partial copies of datasets, defined as collections of files. The replica management service in a data grid environment provides support for high-performance, data-intensive applications. A replica or location is a subset of a collection that is stored in a particular physical storage system, which means that multiple possibly overlapping subsets of collections may be stored in multiple storage systems in a data grid. These grid storage systems may use a variety of underlying storage technologies and data movement protocols, which are independent of replica management.

A data grid may contain multiple replica catalogs. It is possible to create hierarchies of replica catalogs to impose a directory-like structure on related logical collections. The purpose of the replica catalog is to provide mappings between logical names for files or collections and one or more copies of objects in physical storage systems. The catalog registers three types of entries: logical collections, locations, and logical files. A logical collection is a user-defined group of files. We expect that users will find it convenient and intuitive to register and manipulate groups of files as collections, rather than require that every file be registered and manipulated individually.

Despite the benefits of registering and manipulating collections of files using logical collection and location objects, there may be a need for users and applications to characterize individual files. For this purpose, the Replica Catalog includes optional entries that describe individual logical files. Logical files are entities with globally unique names and one or more physical instances. The catalog may optionally contain one logical file entry in the Replica Catalog for each logical file in a collection.

Replica selection [16] is used to select replicas from among the sites in a data grid. The selection criteria depend on application characteristics. This mechanism enables users to efficiently manage replicas of data sets at their sites. Much previous effort has been devoted to solving replica selection problems. The replica selection process commonly consists of three steps: data preparation, preprocessing, and prediction. Applications then select replicas according to their specific attributes.

Replica selection is important to data-intensive applications, and it can provide location transparency. When a user requests access to a data set, the system will determine an appropriate way to deliver the replica. Another issue concerning replica selection is the predicting transfer time, a complex task involving the inspection of many characteristics.

### 2.2 The co-allocation architecture and related work

Candidate replica locations are passed to the replica selection service, which was presented in a previous work [5, 6, 17, 18, 21, 28]. This replica selection service provides estimates of candidate transfer performance based on a cost model and chooses appropriate amounts to request from the better locations. The architecture proposed in [17] consists of three main components: an information service, a broker/co-allocator, and local storage systems. Figure 1 shows data transfer in a co-allocation architecture, which is an extension of the basic template for resource management [6] provided by the Globus Toolkit. Applications specify the characteristics of desired data and pass the attribute description to a broker. The broker queries available resources and gets replica locations from an information service [7, 15, 28] and a replica management

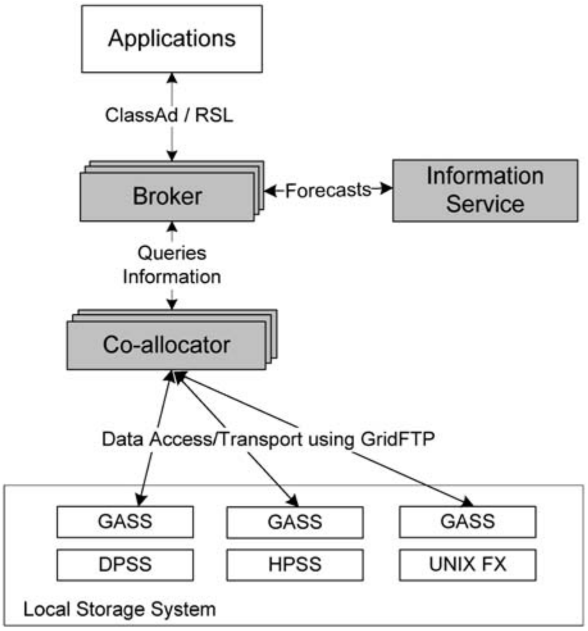


**Fig. 1** The co-allocation architecture in data grids

service [21] creates a list of the desired files physical locations. The co-allocation agent then downloads the data in parallel from the selected servers.

Data grids consist of scattered computing and storage resources located in different countries/regions yet accessible to users [8]. As datasets are replicated within grid environments for reliability and performance, clients require the abilities to discover existing data replicas, and create and register new replicas. A replica location service (RLS) [5] provides a mechanism for discovering and registering existing replicas. Several prediction metrics have been developed to help replica selection. For instance, Vazhkudai and Schopf [19–21] used past data transfer histories to estimate current data transfer throughputs.

In [17], the author proposes a co-allocation architecture for co-allocating grid data transfers across multiple connections by exploiting the partial-copy feature of GridFTP. It also provides brute-force, history-based, and dynamic load balancing for allocating data blocks.

- Brute-force co-allocation: brute-force co-allocation (see Fig. 2) works by dividing files equally among "$n$" available flows (locations). Thus, if the data to be fetched is size "$S$" and there are "$n$" locations to fetch it from, then this technique assigns to each flow a data block of size, "$S/n$". For example, if there are three sources, the target file will be divided into three blocks equally. And each source provides one block for the client. With this technique, although all the available servers are utilized, bandwidth differences among the various client-server links are not exploited.
- History-based co-allocation: The history-based co-allocation (see Fig. 3) scheme keeps block sizes per flow proportional to transfer rates predicted by the previous results of file transfer results. In the history-based allocation scheme, the block

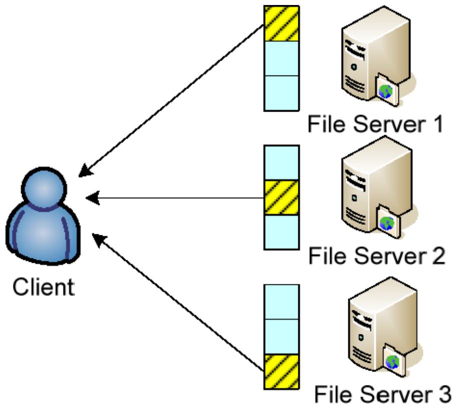


**Fig. 2** The brute-force co-allocation process

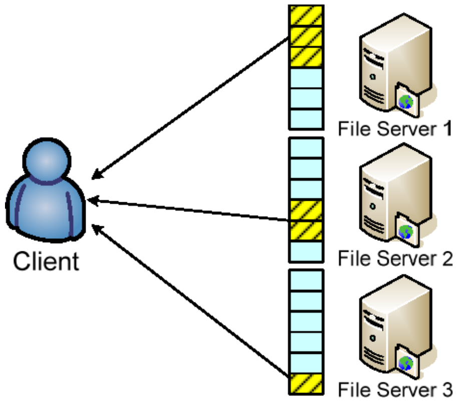


**Fig. 3** The history-based co-allocation process

size per flow is commensurate to its predicted transfer rate, decided based on a previous history of GridFTP transfers. Thus, the file-range distribution is based on the predicted merit of the flow. If these predictions are not accurate enough, re-negotiations of flow sizes might be necessary as slower links can get assigned larger portions of data, which could be weight heavily on the eventual bandwidth achieved. With the history-based approach, client divides the file into "$n$" disjoint blocks, corresponding to "$n$" servers. Each server "$i$", $1 \leq i \leq n$, has a predicted transfer rate of "$B_i$" to the client. In theory then, the aggregate bandwidth "$A$" achievable by the client for the entire download is $A = \sum_{i=1}^{i=n} B_i$. For each server "$i$", $1 \leq i \leq n$, and for the data to be fetched is the size of "$S$", the block size per flow is $S_i = \frac{B_i}{A} \times S$.

- Conservative load balancing: One of the proposed dynamic co-allocation algorithms [17] is conservative load balancing (Fig. 4). The conservative load balancing dynamic co-allocation strategy divides requested datasets into "$k$" disjoint blocks of equal size. Available servers are assigned single blocks to deliver in parallel. When a server finishes delivering a block, another is requested, and so on, until the entire file is downloaded. The loadings on the co-allocated flows are automatically adjusted because the faster servers will deliver more quickly providing larger portions of the file.

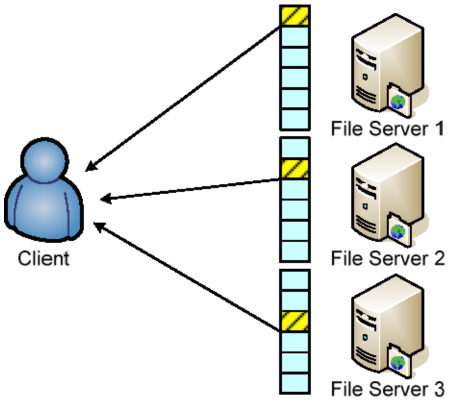


**Fig. 4** The conservative load balancing process

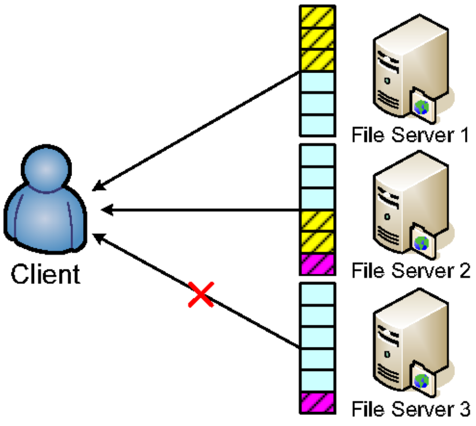


**Fig. 5** The aggressive load balancing process

- Aggressive load balancing: This method is shown in Fig. 5 and adds functions that change block size in deliveries by: (1) gradually increasing the amounts of data requested from faster servers, and (2) reducing the amounts of data requested from slower servers or stopping requesting data from them altogether.

In our previous work [24, 26], we proposed a replica selection cost model and a replica selection service to perform replica selection. These co-allocation strategies do not address the shortcoming of faster servers having to wait for the slowest server to deliver its final block. In most cases, this wastes much time and decreases overall performance. Thus, we propose an efficient approach, called the *dynamic adjustment strategy*, and based on the co-allocation architecture. It improves dynamic co-allocation and reduces waiting time, thus improving overall transfer performance.

## 3 The dynamic adjustment strategy

Dynamic co-allocation, described above, is the most efficient approach to reducing the influence of network variations between clients and servers. However, the idle time of faster servers waiting for the slowest server to deliver its last block is still a major factor affecting overall efficiency, which conservative load balancing and aggressive load balancing [17] cannot effectively avoid. The approach proposed in the present paper, a dynamic allocation mechanism, called *dynamic adjustment strategy*, can overcome this, and thus, improve data transfer performance.

Co-allocation technology [17] enables the clients to download data from multiple locations by establishing multiple connections in parallel. We proposed a replica selection cost model and a replica selection service to perform replica selection. We now propose a new data transfer strategy based on this model. It consists of three phases: (1) initial phase, (2) steady phase, and (3) completion phase.

- Initial phase: We assign equal block sizes to all GridFTP servers. In this phase, our system determines the next block size for each replica server.
- Steady phase: As job transfers are completed, servers are assigned their next jobs. Jobs sizes are determined by multiplying the client bandwidth by the weighting.
- Completion phase: To avoid the generating excessively small job sizes, we set an end condition such that if the remaining target file size is smaller than the initial block size, it is transferred immediately.

The parameters used for our algorithm are listed in the following:

- *Job size*: the next job size for a server sending to the client.
- *initialPT*: the initial job size of the transferred file.
- *remnantFileSize*: the remnant file size of the transferred file.
- *ClientBandwidth*: the bandwidth of current client.
- *ClientMaxBandwidth*: the current client max bandwidth.
- *Number of Replica Source*: the number of replica sources for parallel transferring.
- $Score_i$: the replica selection cost model for server $i$ such that $1 \leq i \leq n$.
- $P_i^{\mathrm{CPU}}$: percentage of server $i$ CPU idle states [15].
- $R^{\mathrm{CPU}}$: CPU load ratio defined by the user.
- $P_i^{\mathrm{Mem}}$: percentage of server $i$ memory free space [15].
- $R^{\mathrm{Mem}}$: memory free space ratio defined by the user.
- $P_i^{\mathrm{BW}}$: percentage of bandwidth available from server $i$ to client (user node); current bandwidth divided by highest theoretical bandwidth [23, 24, 26].
- $R^{\mathrm{BW}}$: network bandwidth ratio defined by users.
- $weighting_i$: the weight model for server $i$ such that $1 \leq i \leq n$.
- $newPT_i$: the next job size for server $i$.

To determine the initial block size, we set an upper bound that is dependent on the relation between the client's maximum bandwidth and the number of replica sources. Though multiple replicas can be downloaded in parallel, the gathered portions of files from different links must be transferred to the client in a single link. It is clear that the client's bandwidth could be a bottleneck in co-allocation architecture. The formula for upper bound is:

$$initialPT \leq ClientMaxBandwidth/Number\ of\ \ Replica\ Source \tag{1}$$

We proposed a replica selection cost model in which we defined a formula for calculating the weighting. First, we get a score based on the states of the various server devices:

$$Score_i = P_i^{\mathrm{CPU}} \times R^{\mathrm{CPU}} + P_i^{\mathrm{Mem}} \times R^{\mathrm{Mem}} + P_i^{\mathrm{BW}} \times R^{\mathrm{BW}},$$
$$\text{and} \quad R^{\mathrm{CPU}} + R^{\mathrm{Mem}} + R^{\mathrm{BW}} = 1 \tag{2}$$

After getting the scores for all server nodes, the system calculates the $weighting_i$:

$$weighting_i = Score_i \bigg/ \sum_{k=1}^{n} Score_k \tag{3}$$

The weighting is then used to determine the size of the next job:

$$newPT_i = ClientBandwidth \times weighting_i \tag{4}$$

where $newPT_i$ denotes the next job size for server $i$, and $ClientBandwidth$ denotes the current client bandwidth.

When server $i$ finishes transferring of a block, it gets a new job whose size is calculated according to the real-time status of server $i$. Each time, our strategy dynamically adjusts a job size according to source device loading and bandwidth. The lighter the loading a source device has, the larger job size is assigned. Figure 6 shows a flowchart illustrating this new strategy.

Next, the average transfer rate of all replicas can be calculated by total transferred file size divided the cost time ratio of combination of CPU, memory, and network bandwidth. We used the dynamic adjustment strategy with various sets of replica servers and measured overall performances, where overall performance is:

$$Total\ Performance = File\ Size / Total\ Completion\ \ Time \tag{5}$$

## 4 An efficient toolkit: cyber-transformer

We gave experimental results for cyber-transformer, a powerful new toolkit for replica management and data transfer in data grid environments. It not only can accelerate data transfer rate, but can also manage replicas over all various sites. The friendly interface enables users to easily monitor replica sources, and add files as replicas for automatic cataloging by our replica location service. Moreover, we provide a function for administrators to delete and modify replicas. Cyber-transformer can be invoked with either the logical file name of a data file or a list of replica sources host names. When users search for a file by its logical file name, cyber-transformer queries the replica location services to find all the corresponding replicas, and contacts each replica source to start parallel transfers. The file is then gathered from replica sources and finally combined into a single file.

### 4.1 System components

Cyber-transformer is implemented in the Java Cog Kits [13] library. Figure 7 shows the system stack of cyber-transformer, consisting of three integrated mechanisms:

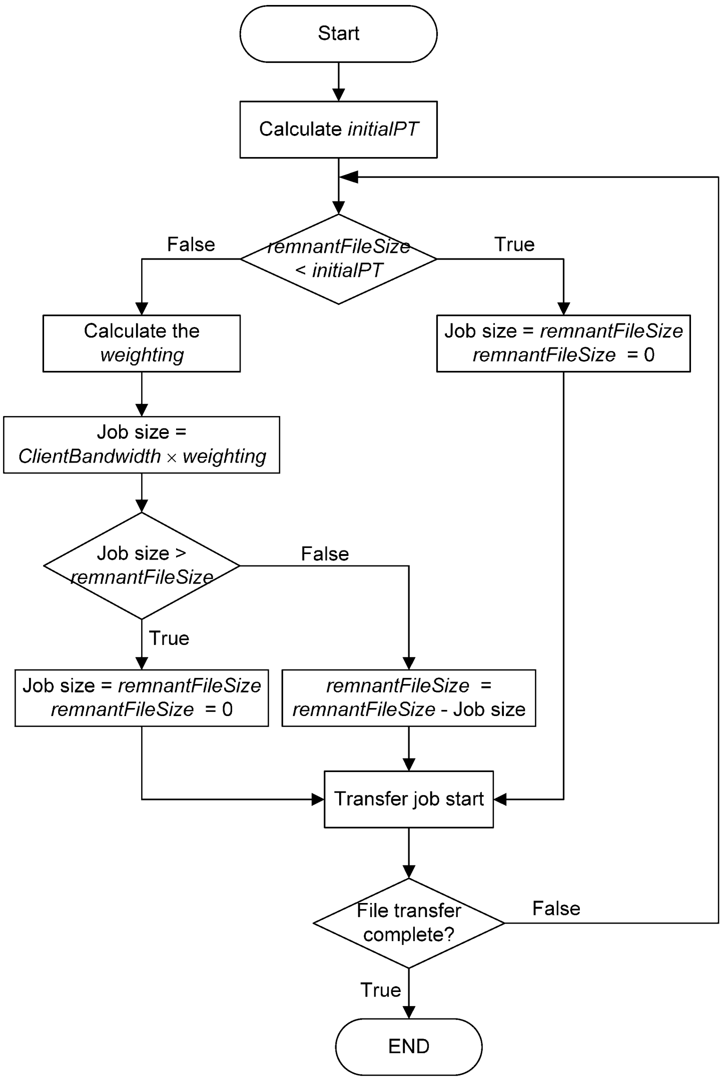

Start
Calculate *initialPT*
*remnantFileSize* < *initialPT*
False
True
Calculate the *weighting*
Job size = *ClientBandwidth* × *weighting*
Job size = *remnantFileSize*
*remnantFileSize* = 0
Job size > *remnantFileSize*
False
True
Job size = *remnantFileSize*
*remnantFileSize* = 0
*remnantFileSize* = *remnantFileSize* - Job size
Transfer job start
File transfer complete?
False
True
END

**Fig. 6** The flowchart of dynamic adjustment strategy

(1) information service, (2) replica management service, and (3) data transfer service. It also includes a friendly GUI for inexperienced users who may not be familiar with data grids.

The interface consists of three parts: (1) information monitor, (2) replica manager, and (3) GridFTP browser, to simplify replica management and data transfers. With the intuitive interface, users can easily invoke the services to transfer data without delay. Figure 8 shows the Cyber-Transformer system components and the three main services they provided.

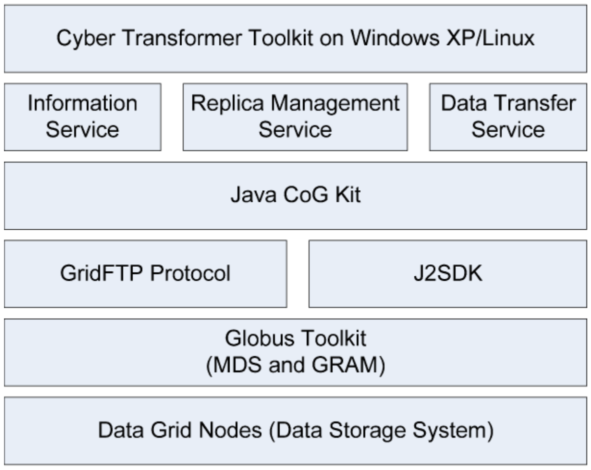

Cyber Transformer Toolkit on Windows XP/Linux
Information Service
Replica Management Service
Data Transfer Service
Java CoG Kit
GridFTP Protocol
J2SDK
Globus Toolkit (MDS and GRAM)
Data Grid Nodes (Data Storage System)

**Fig. 7** The system stack of cyber-transformer

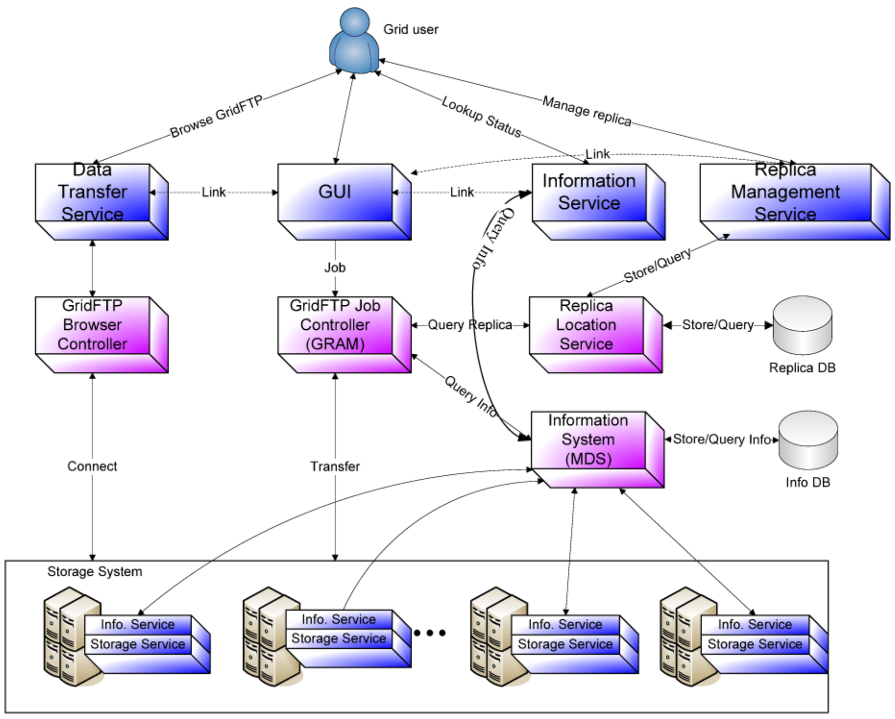


**Fig. 8** The components of cyber-transformer

- Information service: This service is invoked by the information monitor and provides replica sources statuses allowing users to monitor all replica source sites in the data grid. Sites status, such as CPU loading, free memory, hard disk free space, and bandwidth, are gathered by the information service and reported to the information monitor.

- Replica management service: This serves as middleware between users and replica databases. It enables convenient user replica searches by listing logical file names and replica source host names. Users can also easily upload files as replicas, and mark the importance of these files.
- Data transfer service: This is the most important cyber-transformer service, and is easily summoned through the GridFTP Browser. Our dynamic adjustment strategy is integrated into it, and an "option" function enables users to compensate for various data grid environment conditions by adjusting transfer factors such as machine loading, bandwidth, partition size, and stripe numbers, thus accelerating data transfer rates.

## 4.2 System transaction flow

Figure 9 shows the cyber-transformer transaction flow. Users must first pass the grid proxy certification provided by Simple CA to get access to the grid. They may then connect to any data grid site via the GridFTP browser. The system automatically authenticates site certifications as connections are made. The security mechanism of our grid environment is depicted below.

Steps 4 and 5 show how users query the replica location service for replica information, and the replica location service reports on requests. The system ranks all replica servers according to our replica selection model and users can then choose the better servers for parallel downloading.

The data transfer service is invoked in Step 6. Information about the replicas chosen by the user is picked up by the GridFTP job controller. The controller then dynamically adjusts replica transfer job sizes according to the conditions presented in the information. Job sizes are continually adjusted until all transfers have been completed. The portions from the various replica sources are then gathered into complete file.

To enable users lacking deep knowledge of data grids to easily download and manage files in data grid environments, we developed a user-friendly GUI for cyber-transformer. It is implemented in the Java CoG library (see below), and it can be executed on any operating system with JVM. Figure 10 and Fig. 11 show part of the file download operating process.

## 4.3 Improvements in Java CoG for parallel downloading

In [17], the author proposes a co-allocation cost model. He defined that clients downloading datasets using GridFTP co-allocation technology incur three time costs: the time required for client authentication to the GridFTP server, actual data transmission time, and data block reassembly time. Our approach can reduce the data block reassembly time to zero. A function in cyber-transformer allows delivered file portions to be written to one destination file in parallel without extra overhead.

The Java Commodity Grid Kit (Java CoG, http://www-unix.globus.org/cog/) combines Java technology with grid computing to develop advanced grid services and accessibility to basic Globus resources. It allows easier and more rapid application development by encouraging collaborative code reuse and avoiding duplication of

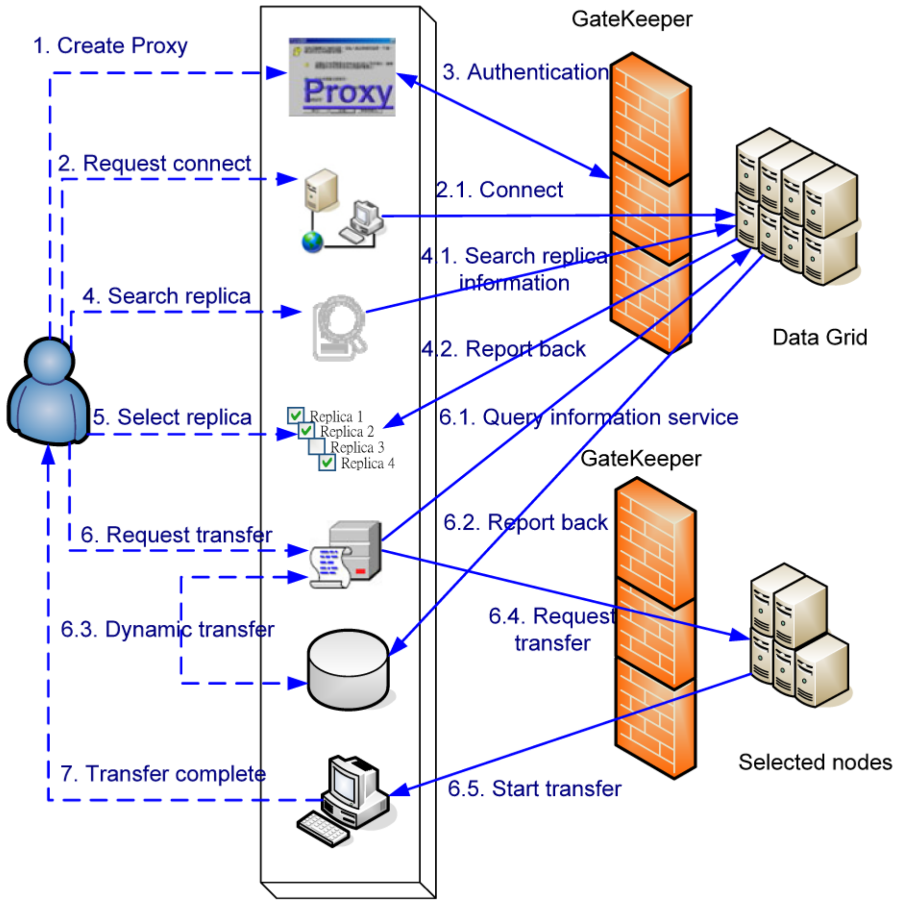
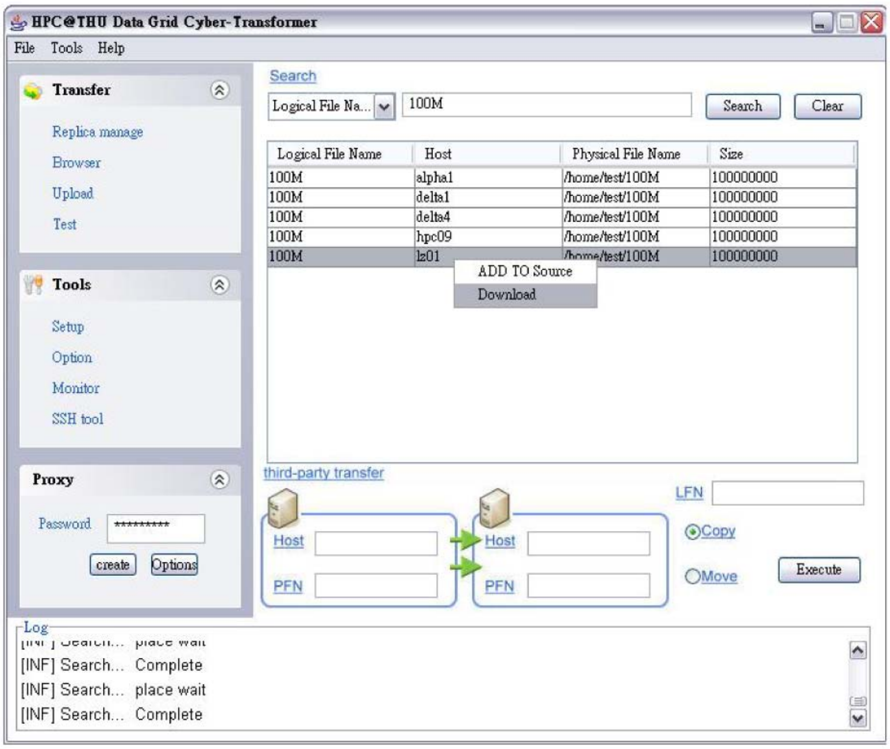


**Fig. 9** The cyber-transformer transaction flow

effort among problem-solving environments, science portals, grid middleware, and collaborative pilots.

The Java-based application uses the Java CoG kit to connect to the grid system. The key characteristics include: GridProxyInit, a JDialog for submitting pass phrases to grids to extend certificate expiration dates, GridConfigureDialog uses the UITool in the CoG Kit to enable users to configure process numbers and host names of Grid servers, and GridJob, which creates GramJob instances. This class represents a simple gram job and allows for submitting jobs to a gatekeeper, canceling them, sending signal commands, and registering and unregistering from callbacks. The GetRSL, RSL provides a common interchange language to describe resources.

The Java CoG GridFTP API does not support downloading files in multiple streams and simultaneously writing them to the same file, which causes some combination overhead after all transmissions. Thus, we needed an effective method for writing to a file in parallel. To resolve the situation, we analyzed and rewrote the Java



Logical File Na... 100M

Password **********

| Logical File Name | Host | Physical File Name | Size |
|---|---|---|---|
| 100M | alpha1 | /home/test/100M | 100000000 |
| 100M | delta1 | /home/test/100M | 100000000 |
| 100M | delta4 | /home/test/100M | 100000000 |
| 100M | hpc09 | /home/test/100M | 100000000 |
| 100M | lz01 | /home/test/100M | 100000000 |

```
[INF] Search... place wait
[INF] Search... Complete
[INF] Search... place wait
[INF] Search... Complete
```

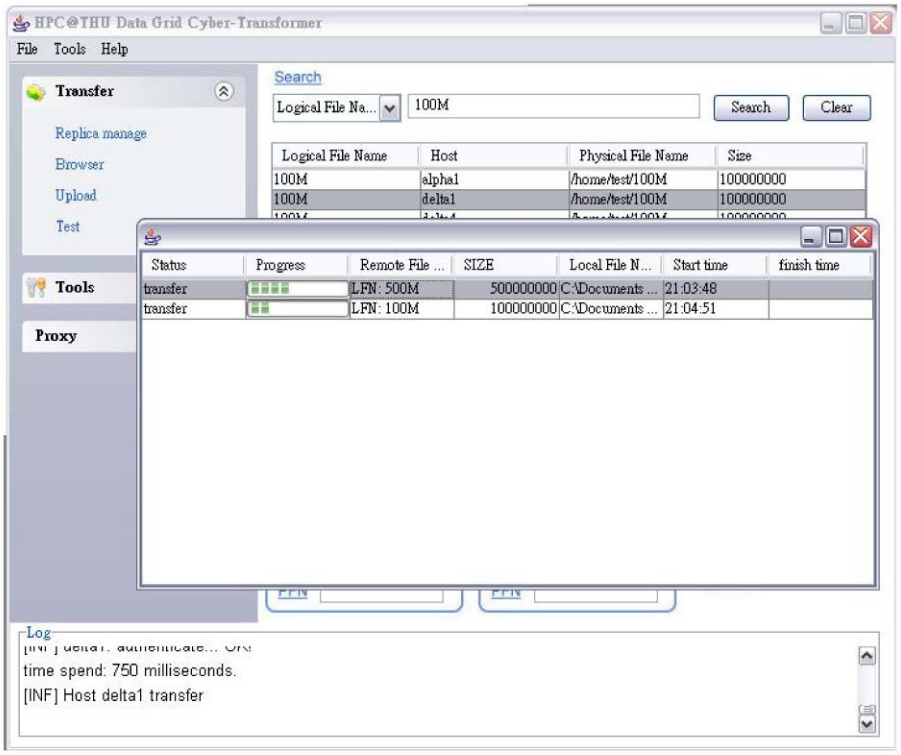
**Fig. 10** Searching for replicas to download

CoG GridFTP codes. We found that the GridFTP write file offers a class that can use **FileRandomIO**. The class code is listed below.

```
public synchronized void write(Buffer buffer) throws IOException {
    long bufOffset = buffer.getOffset();

    if (bufOffset == -1) {
       if (file.getFilePointer() != this.offset) {
       throw new IOException("Invalid offset: " + bufOffset);
       }
    } else {
        file.seek(bufOffset);
    }

    file.write(buffer.getBuffer(), 0, buffer.getLength());
    this.offset += buffer.getLength();
}
```



| Logical File Name | Host | Physical File Name | Size |
|---|---|---|---|
| 100M | alpha1 | /home/test/100M | 100000000 |
| 100M | delta1 | /home/test/100M | 100000000 |

| Status | Progress | Remote File ... | SIZE | Local File N... | Start time | finish time |
|---|---|---|---|---|---|---|
| transfer | 🟩🟩🟩🟩 | LFN: 500M | 500000000 | C:\Documents ... | 21:03:48 | |
| transfer | 🟩🟩 | LFN: 100M | 100000000 | C:\Documents ... | 21:04:51 | |

```
time spend: 750 milliseconds.
[INF] Host delta1 transfer
```

**Fig. 11** The file download process

The class, **RandomAccessFile** in **file.seek(bufOffset)** provides a function that can change the write pointer. This allowed us to write another class to inherit **FileRandomIO** from Java CoG, and overwrite the method, public synchronized **void write(Buffer buffer)**. We also added a method to change the write pointer. This gave us more transmission time to write data to a file at the same time. All streams write to assigned file positions, not to the beginning of the file. That does not affect other streams transferring data and writing files. The new code after our changes is listed below.

```
protected long filePointer;

/**
* set write pointer
* @param filePointer write pointer
*/

public void setFileOffset(long filePointer) {
this.filePointer = filePointer;
}
```

```
/**
* Overwrite FileRandomIO,enable to change write pointer
*/

public synchronized void write(Buffer buffer) throws IOException {
    long bufOffset = buffer.getOffset();

    if (bufOffset == -1) {
        if (file.getFilePointer() != this.offset) {
            throw new IOException("Invalid offset: " + bufOffset);
        }
        } else {
            file.seek(filePointer + bufOffset);
        }

        file.write(buffer.getBuffer(), 0, buffer.getLength());

        this.offset += buffer.getLength();
    }
}
```

Because the network environment is the key point affecting file transfers, we need to measure the bandwidth between the client and a desired node. Using another tool to measure the network environment between a client and a GridFTP server would be very inefficient; so, the need is to measure the current transfer speed according to the transmission volume during file transfers. The transfer part of Java CoG offers the interface, **DataSink**, which allows applications to decide which methods to use when writing files. We added some code to measure file transfer speed to this interface, and the resulting code is listed below.

```
      DataSink lo_DataSink = new DataSink() {

          public synchronized void write(Buffer buffer) throws
                  IOException {
              long size = buffer.getLength();

              fileRandomIO.write(buffer);

              long end = System.currentTimeMillis();
              rate = size / (end - start);
              start = end;
          }

          public void close() throws IOException {
              fileRandomIO.close();
          };
      };
```

The boldface type shows that the key point in measuring the current transfer speed is using time difference and file-writing duration. This gives the transfer speed during file transfers, thus there is no separate reassembly time cost. We overcome one co-allocation shortcoming, and the completion time is just the sum of the authentication and data transmission times.

## 5 Experimental results and analysis

In this section, we discuss the performance of our recursive-adjustment co-allocation strategy. We evaluate four co-allocation schemes: (1) brute-force (Brute), (2) history-based (History), (3) conservative load balancing (Conservative), (4) aggressive load balancing (Aggressive) and (5) dynamic adjustment strategy (DAS). We analyze the performance of each scheme by comparing their transfer finish times, and the total idle time faster servers spent waiting for the slowest servers to finish delivering the last block. We also analyze overall performances in the various cases.

### 5.1 Input parameters

We used the following experiments to determine input parameters for the three factors in our strategy: CPU idle state, memory free space, and network bandwidth, and assign ratios to each of the factors.

At first, to determine the effect of network bandwidth on transfer rates, we measured average rates using various bandwidth ratios. As Fig. 12 shows, there was little difference for small file sizes, however, as the file size was increased, a curve became apparent. The transfer rate decreased at bandwidth ratios smaller than 0.6; the peak transfer rate occurred at a ratio of 0.8. This means that we set $R^{\mathrm{CPU}}$, $R^{\mathrm{MEM}}$, and $R^{\mathrm{BW}}$ in the ratio 0.1:0.1:0.8.

In the second experiment, we assessed the effect of CPU computing power on transfer rates. We used three machines with different CPU types, memory sizes fixed

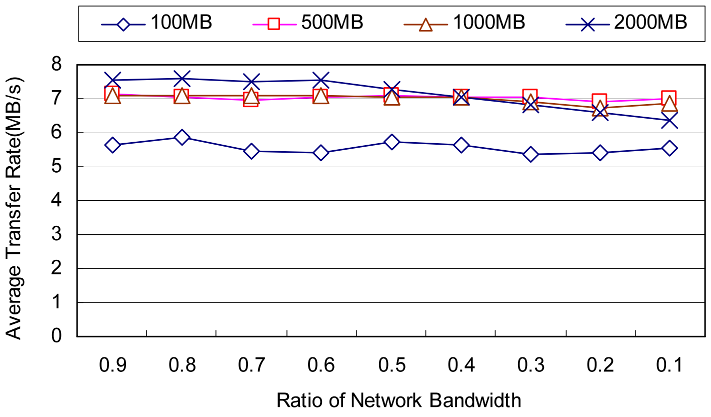


**Fig. 12** The partition size evaluation result

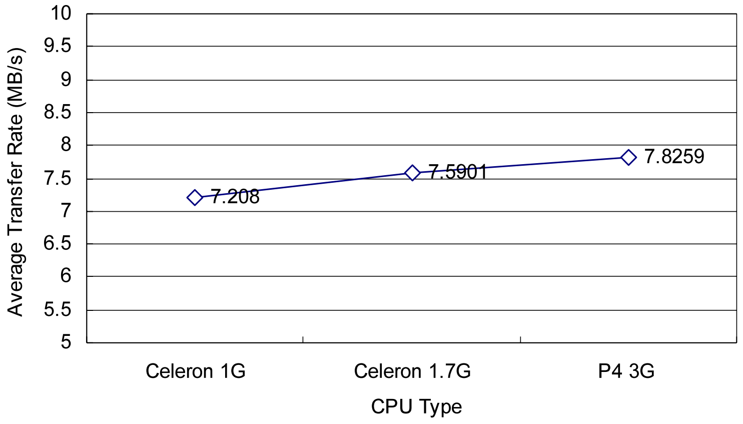


**Fig. 13** Comparison of different CPU types

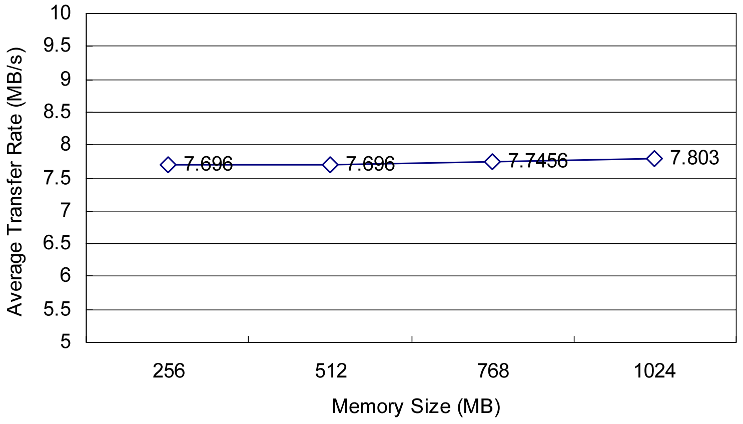


**Fig. 14** Comparison of different memory sizes

at 512 MB, and the same bandwidth: M1, an Intel Celeron 1G; M2, an Intel Celeron 1.7G; and M3, a P4 3G. We set the three machines up as GridFTP sites and measured the average transfer rates for a 500 MB file download from each site to determine performance. The results in Fig. 13 show that the more powerful CPUs performed better, but the increase in efficiency was not proportional to the increase in CPU computing power.

In the third experiment on memory size, we used one machine with different memory sizes as a GridFTP server to measure the relation between memory size and transfer rate performance. The results in Fig. 14 clearly show that increasing memory size has no obvious effect on transfer performance.

The results show that more CPU computing power and larger memory size will improve transfer rates, but not by much. We believe that bandwidth is the most im-

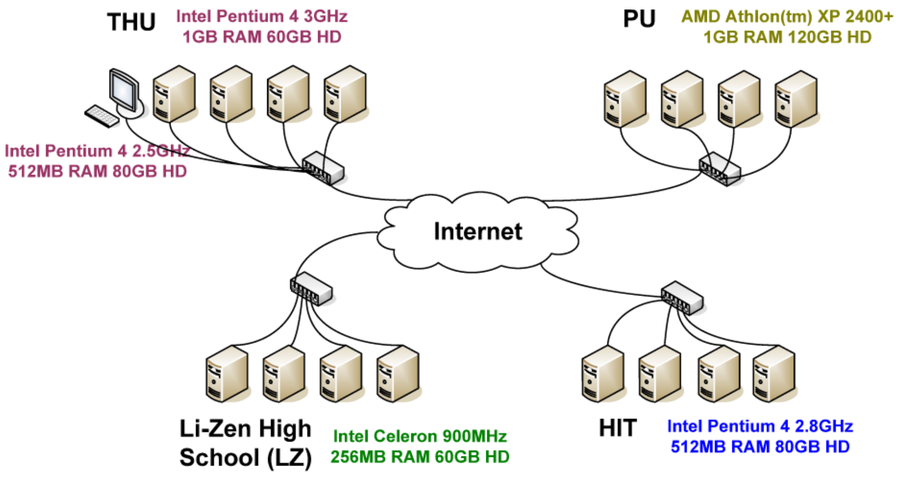


**Fig. 15** Our data grid testbed

portant factor affecting transfer rate, and that the bandwidth ratio should be set larger than the other two factors. CPU power and memory size can be used to make a difference when the bandwidths of several servers are very close.

### 5.2 Experimental environments

We performed wide-area data transfer experiments using our GridFTP GUI client tool. We executed our co-allocation client tool on our testbed at Tunghai University (THU), Taichung City, Taiwan, and fetched files from four selected replica servers: one at Providence University (PU), one at Li-Zen High School (LZ), and the other one at Hsiuping Institute of Technology School (HIT). All these institutions are in Taichung, Taiwan, and each is at least 10 km from THU. Figure 15 shows our data grid testbed, and Table 1 is the detailed listing. All servers had Globus 3.2.1 or above installed.

In the following experiments, we set $R^{\mathrm{CPU}}, R^{\mathrm{MEM}}$, and $R^{\mathrm{BW}}$ in the ratio 0.1:0.1:0.8. We experimented with file sizes of 10 MB, 50 MB, 100 MB, 500 MB, 1000 MB, 1500 MB, and 2000 MB. For comparison, we measured the performance of conservative load balancing on each size using the same block numbers.

Table 2 shows average transmission rates between THU and each replica server. These numbers were obtained by transferring files of 100 MB, 500 MB, 1000 MB, and 2000 MB from a single replica server using our GridFTP client tool, and each number is an average over several runs.

### 5.3 Results and analysis

We examined the effect of faster servers waiting for the slowest server to deliver the last block for each scheme. Figure 16 shows total idle times for various file sizes. Note that our Dynamic Adjustment Strategy performed significantly better than the other

**Table 1** Detailed list of our Data Grid testbeds

| Site | Hostname | CPU Type | Clock (MHz) | RAM | NIC | Linux kernel | Globus version |
|---|---|---|---|---|---|---|---|
| THU | delta1 | Intel Pentium 4 | 3001 | 1 GB | 1 G | 2.6.12 | 4.0.1 |
| THU | delta2 | Intel Pentium 4 | 3001 | 1 GB | 1 G | 2.6.12 | 4.0.1 |
| THU | delta3 | Intel Pentium 4 | 3001 | 1 GB | 1 G | 2.6.12 | 4.0.1 |
| THU | delta4 | Intel Pentium 4 | 3001 | 1 GB | 1 G | 2.6.12 | 4.0.1 |
| LZ | lz01 | Intel Celeron | 898 | 256 MB | 10/100 | 2.4.20 | 3.2.1 |
| LZ | lz02 | Intel Celeron | 898 | 256 MB | 10/100 | 2.4.20 | 3.2.1 |
| LZ | lz03 | Intel Celeron | 898 | 384 MB | 10/100 | 2.4.20 | 3.2.1 |
| LZ | lz04 | Intel Celeron | 898 | 256 MB | 10/100 | 2.4.20 | 3.2.1 |
| HIT | gridhit0 | Intel Pentium 4 | 2800 | 512 MB | 10/100 | 2.6.12 | 3.2.1 |
| HIT | gridhit1 | Intel Pentium 4 | 2800 | 512 MB | 10/100 | 2.6.12 | 3.2.1 |
| HIT | gridhit2 | Intel Pentium 4 | 2800 | 512 MB | 10/100 | 2.6.12 | 3.2.1 |
| HIT | gridhit3 | Intel Pentium 4 | 2800 | 512 MB | 10/100 | 2.6.12 | 3.2.1 |
| PU | hpc09 | AMD Athlon XP | 1991 | 1 GB | 1 G | 2.4.22 | 3.2.1 |
| PU | hpc10 | AMD Athlon XP | 1991 | 1 GB | 1 G | 2.4.22 | 3.2.1 |
| PU | hpc11 | AMD Athlon XP | 1991 | 1 GB | 1 G | 2.4.22 | 3.2.1 |
| PU | hpc12 | AMD Athlon XP | 1991 | 1 GB | 1 G | 2.4.22 | 3.2.1 |

**Table 2** GridFTP end-to-end transmission rates from THU to various servers

| Replica server | Average transmission rate |
|---|---|
| HIT | 61.5 Mbits |
| LZ | 49.5 Mbits |
| PU | 26.7 Mbits |

schemes on every file size. These results demonstrate that our approach efficiently reduces the differences in servers finish times.

Figure 17 shows total completion times in a detailed cost-structure view. Servers were at PU, LZ, and HIT, with the client at THU. The first three bars for each file size denote the time to download the entire file from single server, while the other bars show co-allocated downloads using all three servers. Our co-allocation strategy finished the jobs faster than the other strategies, and there was no combination time cost. Thus, we may infer that the main gains our technology offers as a result of the modifications we presented in Sect. 4.3 are no combination time, and faster transmission than other co-allocation strategies.

Table 3 lists all experiments we performed and the sets of replica servers used. The results in Fig. 18 show that using co-allocation technologies yielded no improvement for smaller file sizes such as 10 MB. They also show that in most cases, overall performance increased as the number of co-allocated flows increased. We observed that for our testbed and our co-allocation technology, overall performance reached its highest value in the DAS2_2 case. However, in the DAS3 case, when we added one flow to the set of replica servers, the performance did not increase. On the con-

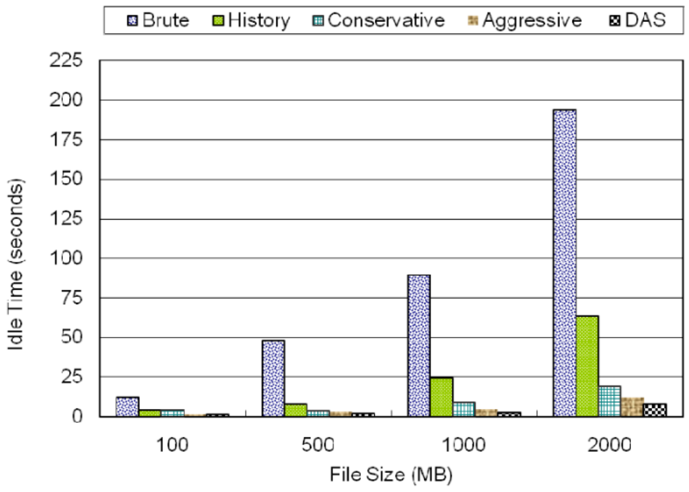


**Fig. 16** Idle times for various methods; servers at PU, LZ, and HIT

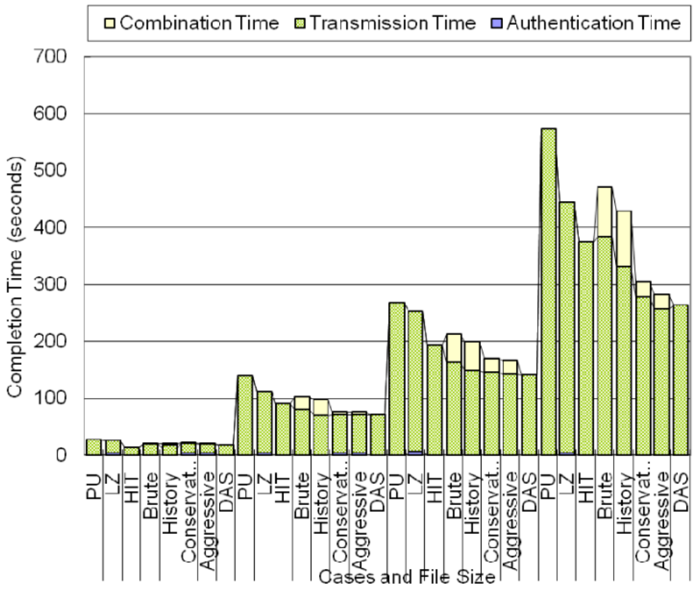


**Fig. 17** Completion times for various methods; servers are at PU, LZ, and HIT

trary, it decreased. We can infer that the co-allocation efficiency reached saturation in the DAS2_2 case, and that additional flows caused additional overhead and reduced overall performance because the PU file site had worse network bandwidth, and DAS2_1 choosing PU for file transfer led to the differences between DAS2_1 and DAS2_2. This means that more download flows do not necessarily result in higher

**Table 3** The sets of replica servers for all cases

| Case | Replica servers |
|---|---|
| PU1 | PU |
| LZ1 | LZ |
| DAS2_1 | PU, LZ |
| DAS2_2 | LZ, HIT |
| DAS3 | PU, LZ, HIT |

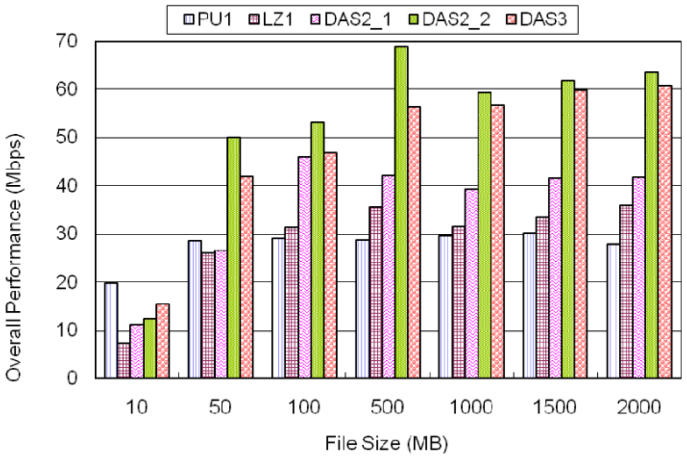


**Fig. 18** Overall performances for various sets of servers

performance. We must choose appropriate numbers of flows to achieve optimum performance.

In the final experimentation, two data transfer scenarios are performed by using our cyber-transformer for file download in parallel. Scenario one is used to conduct the file transfer performance by downloading 1 GB data in LAN environment in THU. And scenario two is used to conduct the file transfer performance by downloading 2 GB data in WAN environment (THU-HIT-LZ-PU) in a real grid. The performance is shown in Fig. 19.

By comparing testing result of the two scenarios there was something difference, as shown in Fig. 20. With less network transformation interference, the overall performance will be decreased 41% and 39%, respectively, when the testing environment was changed from LAN to WAN with brute-force and history schemes. In contrast to the aggressive and our DAS schemes, due to the main design consideration of those two schemes was to reduce link down and exactly dispatch future working load when progressing, there was no obvious performance decrease when the testing environment changed from LAN to WAN.

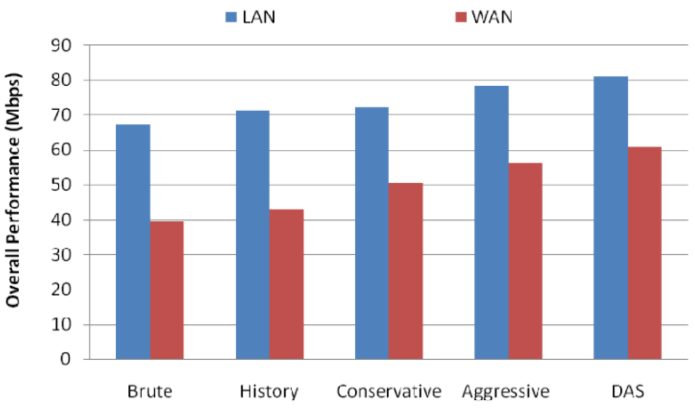
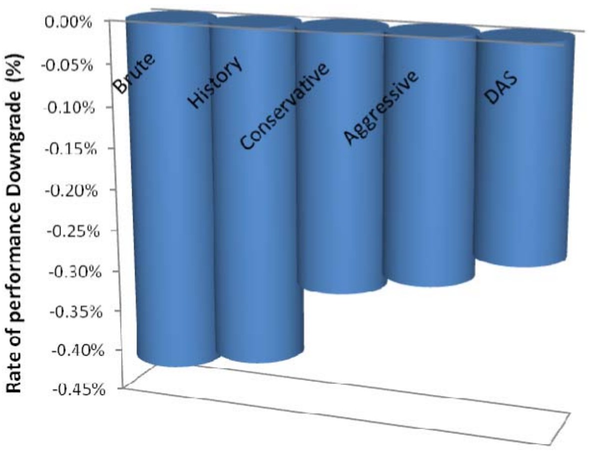


**Fig. 19** Comparison of cyber-transformer transmission rate between LAN and WAN

**Fig. 20** The rate of overall performance downgrade between LAN and WAN



## 6 Conclusions

Using the parallel-access approach to downloading data from multiple servers reduces transfer times and increases server resilience. The co-allocation architecture provides a co-ordinated agent for assigning data blocks. A previous work showed that the dynamic co-allocation scheme leads to performance improvements. However, it cannot handle the idle time of faster servers having to wait for the slowest server to deliver its final block. This paper proposes the dynamic adjustment strategy (DAS) to improve file transfer performances using the co-allocation architecture in data grids. In our approach, the workloads on selected replica servers are continuously adjusted during data transfers, and our approach can also reduce the idle times spent waiting for the slowest servers, and thus decrease file transfer completion times.

We also developed a new toolkit, called cyber-transformer that enables even inexperienced users to easily monitor replica source site statuses, manage replicas, and download files from multiple servers in parallel. Experimental results show the effectiveness of our proposed technique in improving transfer times and reducing overall idle time spent waiting for the slowest servers. We also discussed the cost of combination time and provided an effective improvement. In future work, we will investigate providing more functions for our user-friendly interface, for example, auto parameters input and auto scan to find better replica servers for downloading. We also plan to improve replica management, especially on the problem of replica consistency.

**Acknowledgements** This work is supported in part by National Science Council, Taiwan R.O.C., under grants no. NSC 96-2221-E-029-019-MY3 and NSC 97-2622-E-029-003-CC2.

## References

1. Allcock B, Tuecke S, Foster I, Chervenak A, Kesselman C (2000) Protocols and services for distributed data-intensive science. In: ACAT2000 proceedings, pp 161–163
2. Allcock B, Bester J, Bresnahan J, Chervenak A, Foster I, Kesselman C, Meder S, Nefedova V, Quesnel D, Tuecke S (2001) Secure, efficient data transport and replica management for high-performance data-intensive computing. In: Proceedings of the eighteenth IEEE symposium on mass storage systems and technologies, pp 13–28
3. Allcock B, Bester J, Bresnahan J, Chervenak A, Foster I, Kesselman C, Meder S, Nefedova V, Quesnel D, Tuecke S (2002) Data management and transfer in high-performance computational grid environments. Parallel Comput 28(5):749–771
4. Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S (2001) The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. J Netw Comput Appl 23:187–200
5. Chervenak A, Deelman E, Foster I, Guy L, Hoschek W, Iamnitchi A, Kesselman C, Kunszt P, Ripeanu M (2002) Giggle: a framework for constructing scalable replica location services. In: Proceedings of supercomputing 2002, Baltimore, MD
6. Czajkowski K, Foster I, Kesselman C (1999) Resource co-allocation in computational grids. In: Proceedings of the eighth IEEE international symposium on high performance distributed computing (HPDC-8'99), August 1999
7. Czajkowski K, Fitzgerald S, Foster I, Kesselman C (2001) Grid information services for distributed resource sharing. In: Proceedings of the tenth IEEE international symposium on high-performance distributed computing (HPDC-10'01), August 2001, pp 181–194
8. Donno F, Gaido L, Ghiselli A, Prelz F, Sgaravatto M (2002) DataGrid Prototype 1. In: Proceedings of the TERENA networking conference, June 2002. http://www.terena.nl/conferences/tnc2002/Papers/p5a2-ghiselli.pdf
9. Foster I, Kesselman C (1997) Globus: a metacomputing infrastructure toolkit. Int J Supercomput Appl High Perform Comput 11(2):115–128
10. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the grid: enabling scalable virtual organizations. Int J Supercomput Appl High Perform Comput 15(3):200–222
11. Hoschek W, Jaen-Martinez J, Samar A, Stockinger H, Stockinger K (2000) Data management in an international data grid project. In: First IEEE/ACM international workshop on grid computing—Grid 2000, Bangalore, India, December 2000
12. IBM Red Books, Introduction to grid computing with Globus. IBM Press. http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf
13. Open Grid Forum, http://www.ogf.org/
14. Stockinger H, Samar A, Allcock B, Foster I, Holtman K, Tierney B (2002) File and object replication in data grids. J Clust Comput 5(3):305–314
15. SYSSTAT utilities home page, http://perso.wanadoo.fr/sebastien.godard/
16. The Globus Alliance, http://www.globus.org/
17. Vazhkudai S (2003) Enabling the co-allocation of grid data transfers. In: Proceedings of fourth international workshop on grid computing, November 2003, pp 41–51
18. Vazhkudai S, Schopf J (2002) Predicting sporadic grid data transfers. In: Proceedings of 11th IEEE international symposium on high performance distributed computing (HPDC-11 '02), July 2002, pp 188–196

19. Vazhkudai S, Schopf J (2003) Using regression techniques to predict large data transfers. Int J High Perform Comput Appl (IJHPCA) 17:249–268
20. Vazhkudai S, Schopf J, Foster I (2002) Predicting the performance of wide area data transfers. In: Proceedings of the 16th international parallel and distributed processing symposium (IPDPS 2002), April 2002, pp 34–43
21. Vazhkudai S, Tuecke S, Foster I (2002) Replica selection in the Globus data grid. In: Proceedings of the 1st international symposium on cluster computing and the grid (CCGRID 2001), May 2001, pp 106–113
22. Venugopal S, Buyya R, Ramamohanarao K (2006) A taxonomy of data grids for distributed data sharing, management, and processing. ACM Comput Surv 38(1):1–53
23. Wolski R, Spring N, Hayes J (1999) The network weather service: a distributed resource performance forecasting service for metacomputing. Future Gener Comput Syst 15(5–6):757–768
24. Yang CT, Shih PC, Chen SY (2006) A domain-based model for efficient network information on grid computing environments. IEICE Trans Inf Syst E89-D(2):738–742. Special issue on parallel/distributed computing and networking
25. Yang CT, Yang IH, Chen CH, Wang SY (2006) Implementation of a dynamic adjustment mechanism with efficient replica selection in co-allocation data grid environments. In: Proceedings of the 21st annual ACM symposium on applied computing (SAC 2006)—distributed systems and grid computing track, April 23–27, 2006, pp 797–804
26. Yang CT, Wang SY, Fu CP (2007) A dynamic adjustment mechanism for data transfer in data grids. In: Network and parallel computing: IFIP international conference, NPC 2007, September 17–20. Lecture notes in computer science, vol 4672. Springer, Berlin, pp 61–70
27. Yang CT, Yang IH, Li KC, Wang SY (2007) Improvements on dynamic adjustment mechanism in co-allocation data grid environments. J Supercomput 40(3):269–280
28. Zhang X, Freschl J, Schopf J (2003) A performance study of monitoring and information services for distributed systems. In: Proceedings of 12th IEEE international symposium on high performance distributed computing (HPDC-12 '03), August 2003, pp 270–282

**Chao-Tung Yang** is a professor of Computer Science at Tunghai University in Taiwan. He was born on November 9, 1968, in Ilan, Taiwan, R.O.C. and received the B.Sc. degree in Computer Science from Tunghai University, Taichung, Taiwan, in 1990, and the M.Sc. degree in Computer Science from National Chiao Tung University, Hsinchu, Taiwan, in 1992. He received the Ph.D. degree in Computer Science from National Chiao Tung University in July 1996. He won the 1996 Acer Dragon Award for an outstanding Ph.D. Dissertation. He has worked as an Associate Researcher for ground operations in the ROCSAT Ground System Section (RGS) of the National Space Program Office (NSPO) in Hsinchu Science-based Industrial Park since 1996. In August 2001, he joined the faculty of the Department of Computer Science at Tunghai University. He got the excellent research award by Tunghai University in 2007. In 2007 and 2008, he got the Golden Penguin Award by Industrial Development Bureau, Ministry of Economic Affairs, Taiwan. His researches have been sponsored by Taiwan agencies National Science Council (NSC), National Center for High Performance Computing (NCHC), and Ministry of Education. His present research interests are in grid and cluster computing, parallel and multi-core computing, and Web-based applications. He is a member of both the IEEE Computer Society and ACM.

**Shih-Yu Wang** received the B.Sc. degree in Computer Science at Tunghai University, Taichung, Taiwan, in July 2004. He received the M.Sc. degree in Computer Science at Tunghai University, Taichung, Taiwan, in July 2006. He also works at Industrial Technology Research Institute in Hsinchu City, Taiwan. His research interests include data grid, grid computing, and cluster computing.

**William Cheng-Chung Chu** is the Dean of the Engineering College, a Professor of the Department of Computer Science, and the Director of Software Engineering and Technologies Center of Tunghai University. He had served as the Dean of Research and Development Office at Tunghai University from 2004 to 2007, Taiwan. From 1994 to 1998, he was an Associate Professor at the Department of Information Engineering and Computer Science at Feng Chia University. He was a Research Scientist at the Software Technology Center of the Lockheed Missiles and Space Company, Inc., where he received special contribution awards in 1992 and 1993 and a PIP award in 1993. In 1992, he was also a visiting scholar at Stanford University. He is serving as the Associate Editor for Journal of Software Maintenance and Evolution (JSME) and Journal of Systems and Software (JSS). His current research interests include software engineering, embedded systems, and e-learning. Doctor Chu received his M.Sc. and Ph.D. degrees from Northwestern University in Evanston, Illinois, in 1987 and 1989, respectively, both in computer science. He has edited several books and published over 100 referred papers and book chapters, as well as participated in many international activities, including organizing international conferences.