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(1/3)

計畫編號：NSC 97-2622-E-029-003-CC2

執行期間：自民國 97 年 08 月 01 日起至民國 98 年 07 月 31 日

執行單位：東海大學 資訊工程與科學系

主 持 人：楊朝棟

參與學生：楊明峰 陳秋雄 林志豪

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。

本年度，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。

接下來第 1 年，計畫整合各階層元件，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。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。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 2 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

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 nest first year, we plan to integrate components of all layers by using Globus Tookit, 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. For evaluation, we plan to construct client applications for remotely controlling Workflow, and feedback the work history and results to GKM. Taking Open Source PACS for examples, we plan to conduct experiments on functionality and compatibility of the prototype.

In the nest second year, we 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。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。

本計劃若能承蒙貴會支持，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。

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。

※ 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。

Journal of Network and Computer Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids ☆

Chao-Tung Yang [a,*], Ming-Feng Yang [a,1], Wen-Chung Chiang [b]

[a] High-Performance Computing Laboratory, Department of Computer Science, Tunghai University, Taichung, Taiwan, ROC
[b] Department of Information Networking Technology, Hsiuping Institute of Technology, Taichung County, Taiwan, ROC

ARTICLE INFO

Article history:
Received 4 July 2008
Received in revised form
16 December 2008
Accepted 8 February 2009

Keywords:
Co-allocation
Data grid
Recursively adjusting
Parallel file transfer
Burst Mode

ABSTRACT

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. These schemes show that the more fragments used the more performance. In fact, some schemes can be applied to specific situations; however, most situations are not common actually. For example, how many blocks in a data set should be cut? For this issue, we proposed the anticipative recursively adjusting mechanism (ARAM) in a previous research work. Its best feature is performance tuning through alpha value adjustment. It relies on special features to adapt to various network situations in data grid environments. In this paper, 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.

1. Introduction

An increasing number of scientific applications, e.g., arising from Genomics, Proteomics, and Bioinformatics require exchanges of large volumes of data to support computation (Allcock et al., 2002; Czajkowski et al., 1999, 2001; Foster et al., 2001; Hoschek et al., 2000; Open Grid Forum; Stockinger et al., 2002; The Globus Alliance). Downloading large data sets from replica locations may result in different performance rates because replica sites may have different architectures, system loading, and network connectivity. Bandwidth quality is the most important factor affecting internet transfers between clients and servers, with download speeds being bounded by traffic congestion due to bandwidth limitations.

One method for improving download speeds uses replica selection techniques to determine the best replica locations (Chervenak et al., 2001, 2002; Czajkowski et al., 1999, 2001; Foster and Kesselman, 1997; Yang et al., 2005, 2008; Zhang et al., 2003; Vazhkudai and Schopf, 2002, 2003; Yang et al., 2006). However, downloading data sets from single best servers often results in ordinary transfer rates because bandwidth quality varies unpredictably due to the shared nature of the Internet.

Another method uses co-allocation (Vazhkudai, 2003) technology to download data. Co-allocation architectures were developed to enable clients to download data from multiple locations by establishing multiple connections in parallel, thus improving performance over single-server transfers and helping to alleviate the internet congestion problem (Yang et al., 2007b). Parallel downloading (Vazhkudai et al., 2002, 2001; Wang et al., 2006; Yang et al., 2007a) is a technique used to fetch and download files from multiple sources including Web servers, file servers, P2P nodes, etc. Parallel downloading has been integrated into many Internet applications and has become the core of many P2P systems. It speeds up download times and eliminates the server selection problem (Vazhkudai, 2003; Venugopal et al., 2006; Vazhkudai et al., 2002). Several co-allocation strategies were addressed in previous works (Mathis et al., 1997; Yang et al., 2007a), but drawbacks remain, such as faster servers having to wait for the slowest one to deliver its final block. As shown in Mathis et al. (1997) and Padhye et al. (1998), this may degrade network performance by repeatedly transferring the same block. Hence, it is important to minimize differences in finish times among servers, and to prevent the same blocks from being transferred over different links between servers and clients.

☆ This work is supported in part by the National Science Council, Taiwan, ROC, under Grant nos. NSC 96-2221-E-029-019-MY3 and NSC 97-2622-E-029-003-CC2.
* Corresponding author. Tel.: +886 4 23590415; fax: +886 4 23591567.
E-mail addresses: ctyang@thu.edu.tw (C.-T. Yang), orson@mail.hit.edu.tw (M.-F. Yang), wcchiang@mail.hit.edu.tw (W.-C. Chiang).
[1] Computer Center, Hsiuping Institute of Technology, Taichung County, Taiwan, ROC.

1084-8045/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2009.02.002

In our previous research work, we presented a method for regulating next-section workloads by continuously adjusting the workloads on selected replica servers. The anticipative recursively adjusting mechanism (ARAM) scheme (Yang et al., 2007a) measures the actual bandwidth performance during data file transfers, and, according to previous transfer finish rates, anticipates bandwidth statuses at the next transfer section. The basic idea is to assign less data to selected replica servers with greater network link performance variations since links with more bandwidth variations will have smaller effective bandwidths, as well as smaller transfer finish rates. The goal is to make the expected finish times of all servers be the same.

In this paper, we first present our new approach based on the ARAM co-allocation strategy for data grid environments. We have designed and implemented a TCP bandwidth estimation model and Burst Mode (BM) to enhancing the original ARAM algorithm. Workloads on all selected replica servers are still adjusted according to TCP throughputs and packet loss rates, and faster servers get double or even quadruple throughputs via Burst Mode enabling. Finally, we present Cyber Transformer, a useful toolkit for data grid users. Integrated with the Information Service, Replica Location Service, and Data Transfer Service, its simple, friendly GUI interface makes it easy for inexperienced users to manage replicas and download files in data grid environments. This tool integrates all strategies based on co-allocation architectures including our previous and proposed algorithms.

The remainder of this paper is organized as follows. Related background review and studies are presented in Section 2. Our new approach is outlined in Section 3. Experimental results and a performance evaluation of our scheme are presented in Section 4. Section 5 concludes this research article.

2. Background review and related work

2.1. Co-allocation architecture

The architecture proposed in Vazhkudai (2003) consists of three main components: an information service, a broker/co-allocator, and local storage systems. Fig. 1 shows co-allocation of data grid transfers, an extension of the basic template for resource management (Vazhkudai et al., 2001; Vazhkudai and Schopf, 2002) provided by the Globus Toolkit. Applications specify the characteristics of desired data and pass attribute descriptions to a broker. The broker queries available resources, gets replica locations from the Information Service (Czajkowski et al., 1999, 2001) and Replica Management Service (Czajkowski et al., 2001), then gets lists of physical file locations.

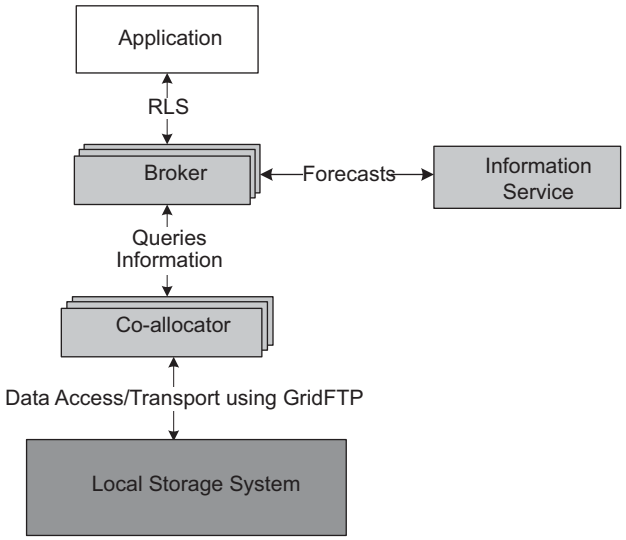

Fig. 1. Data grid co-allocation architecture.

2.1.1. Brute-force co-allocation

The Brute-force co-allocation scheme shown in Fig. 2 divides file sizes equally among available flows; it does not address bandwidth differences among various client–server links.

2.1.2. History-based co-allocation

The history-based co-allocation scheme shown in Fig. 3 keeps block sizes per flow proportional to predicted transfer rates, and disregards the influence of network variations between client and server.

2.1.3. Conservative load balancing

The conservative load balancing scheme shown in Fig. 4 divides requested data sets into k disjoint blocks of equal size. Available servers are allocated single blocks to deliver in parallel. Servers work in sequential order until all requested files are downloaded. Loadings on the co-allocated flows are automatically adjusted because the faster servers deliver larger file portions more quickly.

2.1.4. Aggressive load balancing

This method, shown in Fig. 5, 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

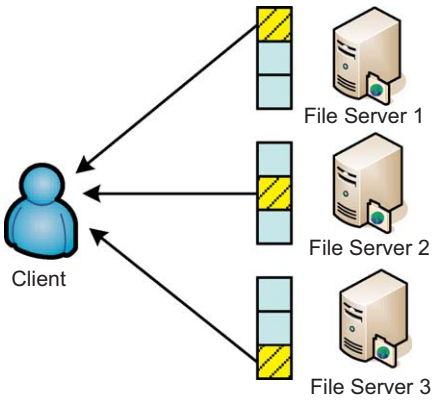

Fig. 2. The Brute-force co-allocation process.

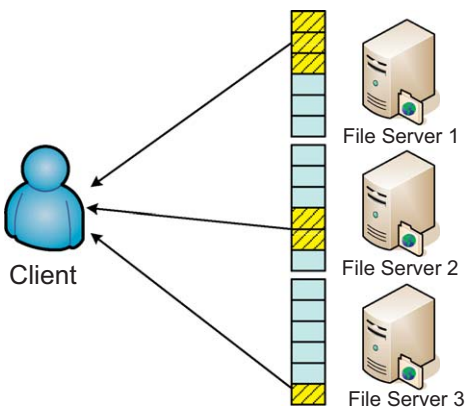

Fig. 3. The history-based co-allocation process.

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

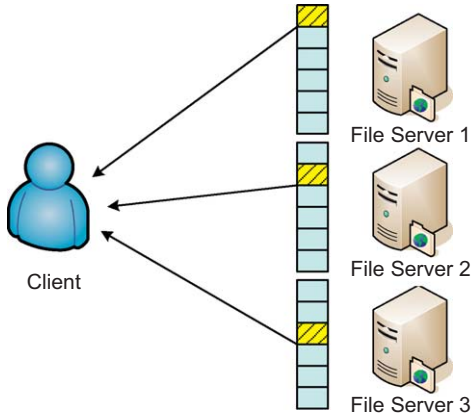

Fig. 4. The conservative load balancing process.

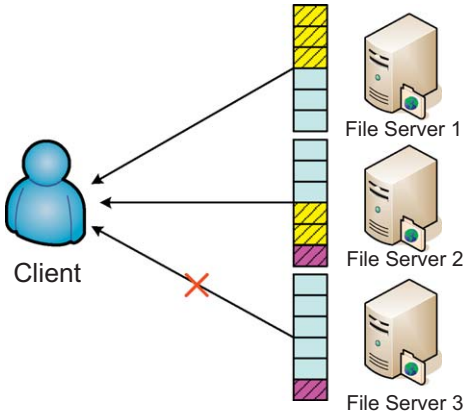

Fig. 5. The aggressive load balancing process.

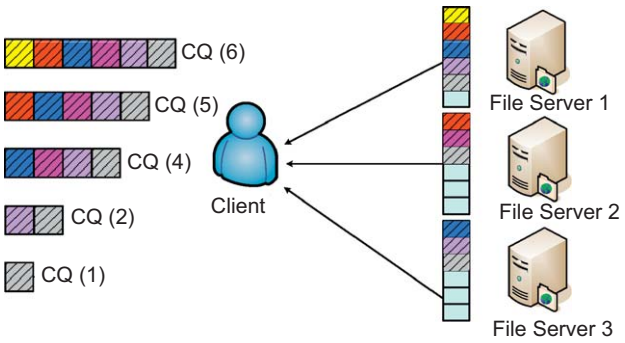

Fig. 6. The DCDA process.

data requested from slower servers or stopping requesting data from them altogether.

2.1.5. Dynamic co-allocation with duplicate assignments (DCDA)

The co-allocation strategies described above do not handle the shortcoming of faster servers having to wait for the slowest server to deliver its final block which, in most cases, wastes much time and decreases overall performance. Neither the prediction nor the heuristic approach, the DCDA scheme dynamically co-allocates duplicate assignments (Bhuvaneswaran et al., 2005, 2007) and copies nicely with changes in server speed performance, as shown in Fig. 6. The DCDA scheme is based on an algorithm that uses a circular queue. Let D be a data set and k the number of blocks of fixed size in the data set. D is divided into k disjoint blocks of equal size and all available servers are assigned to deliver blocks in parallel. When a requested block is received from a server, one of the unassigned blocks is assigned to that server. The co-allocator repeats this process until all blocks have been assigned. DCDA behaves well even when server links are broken or idled. The DCDA scheme is flawed, however, in that it consumes network bandwidth by repeatedly transferring the same blocks. This wastes resources and can easily cause bandwidth traffic jams in the links between servers and clients.

2.1.6. Recursively adjusting mechanism (RAM)

This co-allocation strategy is the most efficient approach to reducing the influence of network variations between clients and servers. However, idle times when faster servers are waiting for the slowest server to deliver its last block are still a major factor affecting overall efficiency that conservative load balancing and aggressive load balancing (Vazhkudai, 2003; The Globus Alliance), cannot effectively avoid. In real-world networking environments, a replica server's available bandwidth might change dynamically as a result of network configuration or load variations. Previous algorithms could not adapt to these dynamisms. Therefore, the greater the degree of bandwidth variation the greater the download times needed. Thus, overall efficiency depends on several factors. Our strategy can overcome such obstacles, and improve data transfer performance. The recursively adjusting mechanism works by continuously adjusting each replica server's workload to correspond to its real-time bandwidth during file transfers. The goal is to make the expected finish times of all servers the same. As Fig. 7 shows, when an appropriate file section is first selected, it is divided into proper block sizes according to the respective server bandwidths. The co-allocator then assigns blocks to servers for transfer. At this moment, it is expected that the transfer finish times will be consistent at $E(t_1)$. However, since server bandwidths may fluctuate during segment deliveries, actual completion times may vary (solid line, in Fig. 7). When the quickest server finishes its work at time t_1, the next section is assigned to the servers. This allows each server to finish its assigned workload by the expected time at $E(t_2)$. These adjustments are repeated until the entire file transfer is finished.

The main purpose of this algorithm is to select appropriate data sources and download from multiple data servers to a single-client resource. We proposed a recursively adjusting co-allocation scheme for parallel downloads from multiple replica servers to a single client. This is useful in cases like downloading music file segments and playing continuous music on a single-client resource. Our algorithms are mainly aimed at transferring parallel data segments from multiple servers to multiple clients for execution of parallel numerical applications on the clients.

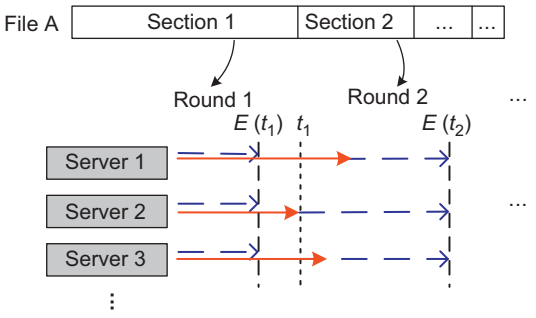

Fig. 7. The adjustment process.

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

The challenge in multiple server–multiple client scenarios is greater since server selections and data downloads on some clients can impact server selections and data transfer performance on other clients.

3. Our approach

3.1. Anticipative recursively adjusting mechanism (ARAM)

The recursively adjusting mechanism reduces file transfer completion times and idle times spent waiting for the slowest server. It also provides an effective scheme for reducing the cost of reassembling data blocks. However, our scheme did not consider the potential effect of server links broken or idled during file transfers. Therefore, we propose an efficient approach called the anticipative recursively adjusting mechanism to extend and improve upon recursively adjusting co-allocation mechanism (Yang et al., in press). The main idea of the ARAM is to assign transfer requests to selected replica servers according to the finish rates for previous transfers, and to adjust workloads on selected replica servers according to anticipated bandwidth statuses. In continuously adjusting selected replica server workloads, the anticipative recursively adjusting mechanism scheme measures actual bandwidth performance during data file transfers and regulates workloads by anticipating bandwidth statuses for subsequent transfers according to the finish rates for previously assigned transfers. The basic idea is to assign less work to selected replica servers on network links with greater performance variability. Links with more bandwidth variation will have smaller effective bandwidths, as well as smaller finish rates for assigned transfers. The goal is to have the expected finished times of all servers be the same. Our approach performs well, even when the links to selected replica servers are broken or idled. It also reduces the idle time wasted waiting for the slowest server. As appropriate file sections are selected, they are first divided into proper block sizes according to the respective server bandwidths, previously assigned file sizes, and transfer finish rates. Initially, the finish rate is set to 1. Next, the co-allocator assigns the blocks to selected replica servers for transfer. At this moment, it is expected that the transfer finish times will be consistent with $E(t_1)$. However, since server bandwidths may fluctuate during segment deliveries, actual completion times may differ from expected times $E(t_1)$ (solid lines in Figs. 8 and 9). When the fastest server finishes at time t_1, the size of unfinished transfer blocks (italic blocks in Figs. 8 and 9) is measured to determine the finish rate. Two outcomes are possible: the quickest server finish time t_1 may be slower than or equal to the expected time, $E(t_1)$, indicating that network link performance remained unchanged or declined during the transfer. In this case, the difference in transferred size between the expected time and actual completion time (italic block in Fig. 8) is then calculated.

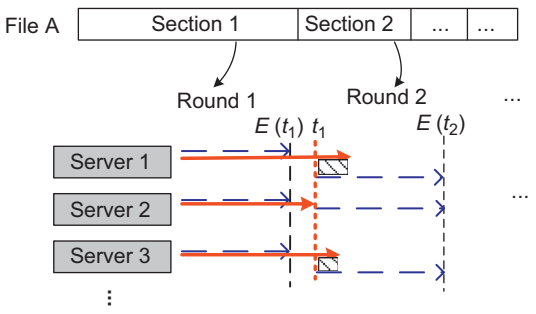

Fig. 8. Later-than-expected-time adjustment process.

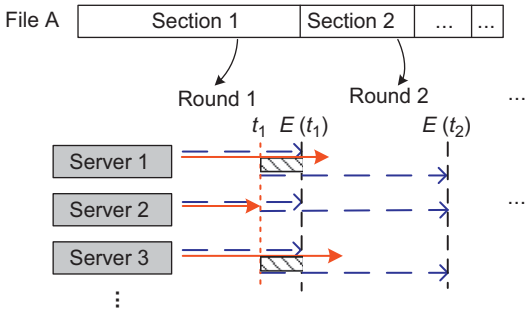

Fig. 9. Earlier-than-expected-time adjustment process.

The other outcome is that the quickest server finish time t_1 may be faster than the expected time, $E(t_1)$, indicating an excessively pessimistic anticipation of network performance, or an improvement in replica server network link performance during the transfer. The difference in transferred size between the expected time (italic block in Fig. 9) and earlier time is then measured. If the anticipated network performance was excessively pessimistic, it is adjusted for the next section. The next task is to assign proper block sizes to the servers along with respective bandwidths and previous finish rates, enabling each server to finish its assigned workload by the expected time, $E(t_2)$. These adjustments are repeated until the entire file transfer is finished.

Looking more closely at ARAM, some parameter definitions are shown below:

- A: file requested by user
- n: selected replica servers
- α: rate that determines how much of the section remains to be assigned
- T_j: allocated time for section j
- SE_j: allocated size for section j
- *UnassignedFileSize*: portion of file A not yet distributed for downloading
- *UnfinishedFileSize*: size of unfinished blocks assigned in previous rounds
- B_{ji}: real-time transfer rate from the selected replica server
- r_j: transfer finish rate
- r_{j-1}: server transfer finish rate for previously assigned delivered file
- B_j: bandwidth available for section j
- S_{ji}: block size per flow from SE_j for each server i at time T_j
- ET_{ji}: expected time for server i at section j
- RT_{ji}: real finish time for server i at section j
- TS_{ji}: actual transfer size at real finish time RT_{ji}
- r_{ji}: job finish rate

When a user requests file A from the data grid environment, the replica selection server responds with a list of all available servers defined as maximum performance data sets/servers. Data sets/servers for the co-allocator to transfer the file are selected, and the target file is then transferred from the chosen replica data sets/servers.

Assume that n replica servers are selected and S_i denotes server "i" for $1 \leqq i \leqq n$. A connection for file downloading is then built to each server.

The anticipative recursively adjusting mechanism process is as follows. A new section of a file to be allocated is first defined. The section size is shown as

$$SE_j = (UnassignedFileSize + TotalUnfinishedFileSize)\alpha, \quad 0 < \alpha \leqslant 1 \tag{1}$$

where SE_j denotes section j such that $1 \leqq j \leqq k$, assume k time is allocated for downloading and there are k sections, while T_j denotes the time allocated to section j. *UnassignedFileSize*, the portion of *File A* awaiting distribution for downloading is initially equal to total file size and *TotalUnfinishedFileSize* is equal to zero in the first round. α is the rate determining how much of the section remains to be assigned.

In the next step, SE_j is divided into several blocks and assigned to "n" servers. Each server has a real-time transfer rate to the selected replica server of B_{ji}. r_{j-1} denotes the server transfer finish rate for previously assigned files, where the initial value is 1. The block size per flow from SE_j for each server "i" at time T_j is S_{ji}:

$$S_{ji} = \frac{SE_{ji}(B_{ji} \times r_j - i)}{\sum_{i=1}^{n}(B_{ji} \times r_j - i)}, \quad 0 \leqslant r_j - i \leqslant 1 \tag{2}$$

$$B_j = \sum_{i=1}^{n}(B_{ji} \times r_j - i) \tag{3}$$

$$ET_{ji} = \frac{S_{ji}}{B_{ji}} \tag{4}$$

This fulfills our requirement to minimize the time faster servers must wait for the slowest server to finish. In some cases, network variations greatly degrade transfer rates. A faster channel may finish its assigned data blocks at real finish time RT_{ji}, or later or earlier than expected time ET_{ji}. Then TS_{ji} denoting the actual transfer size at real finish time RT_{ji} is given by

$$TS_{ji} = B_{ji} \times RT_{ji} \tag{5}$$

If the first finish time for RT_{ji} is earlier than expected time ET_{ji}, the reason may be an excessively pessimistic anticipation of network performance, or the network links used for improvement during the transfer. We compare the block sizes transferred between the earliest and expected times for each server chosen. If the transferred size TS_{ji} is greater than expected size S_{ji} at the first finish time, otherwise, the first finish time for RT_{ji} may be the result of the network link used remaining unchanged or deteriorating during the transfer:

$$r_{ji} = \begin{cases} \dfrac{TS_{ji}}{S_{ji}}, & RT_{ji} \geqslant ET_{ji} \\ 1, & RT_{ji} < ET_{ji}, \text{and } TS_{ji} \geqslant S_{ji} \end{cases} \tag{6}$$

The co-allocator then measures the bandwidth performance of each server, and estimates bandwidth statuses for the next transfer section in order to adjust workflows for the next session. At the same time, it eliminates server *UnfinishedFileSize* listings by summing them up for assignment to the next section.

After allocation, all selected replica servers continue transferring data blocks. When a faster selected replica server finishes its assigned data blocks, the co-allocator allocates an unassigned section of file *A*. Workflows are continually adjusted during the data block allocation process until the entire file has been allocated.

3.2. TCP bandwidth estimation model

TCP/UDP is one of the core protocols in the Internet protocol suite. TCP provides reliable, in-order delivery of a stream of bytes, making it suitable for applications such as GridFTP file transfers. Parallel TCP sockets is a generic "hack" that improves TCP throughputs during bulk data transfers by opening several TCP connections and striping the data files over them (Altman et al., 2006). In practice, it is often unclear how many sockets one needs to open in order to achieve satisfactory throughput, and opening too many connections may be undesirable for various reasons (Altman et al., 2006; Bolliger et al., 1999; Hacker and Athey, 2002; Padhye et al., 1998). The TCP Bandwidth Estimation Model (Hacker and Athey, 2002) as a function to assessing TCP packet loss rate, such as round trip time, maximum segment size, other miscellaneous parameters, etc.

$$TCP_{BW}(p) \approx \min\left(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{2bp/3} + T_0 \min(1, 3\sqrt{3bp/8})p(1+32p^2)}\right) MSS \tag{7}$$

- $TCP_{BW(p)}$: bytes transmitted per second
- MSS: maximum segment size
- W_{max}: maximum congestion window size
- RTT: round trip time
- b: number of transmitted data packets acknowledged by one acknowledgement (ACK) from the receiver (usually $b = 2$)
- T_0: timeout value
- p: packet loss ratio, number of retransmitted packets divided by the total number of packets transmitted
- C: a constant value, initially set to 1.0

In Eq. (7), $TCP_{BW(p)}$ represents bytes transmitted per second, and three factors need to be considered: *MSS*, *RTT*, and *p*. These represent overall TCP bandwidth. For TCP performance assessment, another researcher has simplified them into one:

$$BW \leqslant \frac{MSS}{RTT}\frac{C}{\sqrt{p}} \tag{8}$$

In Eq. (8), *MSS*, *RTT*, and *p* are the same variables used in Eq. (7), *C* is a constant factor, and *BW* represents the number of bytes transmitted per second.

Thus, how the TCP Bandwidth Estimation Model measures server bandwidth makes it more reliable and fair.

3.3. k-means algorithm

The *k*-means algorithm clusters n objects according to attributes into k partitions, $k < n$. It is similar to the expectation-maximization algorithm for Gaussian mixtures in that they both attempt to find natural cluster centers in data. Assuming object attributes form vector spaces, it tries to minimize total intra-cluster variance, or, the squared error function:

$$V = \sum_{i=1}^{k}\sum_{x \in S_i} \|x - m_i\|^2 \tag{9}$$

According to the *k*-means algorithm, where there are random k clusters S_i, $i = 1, 2, \ldots, k$, the Euclid distance of each x point to m_i in S_i, m_i is the cancroids or mean point of all the points $x \in S_i$. Eqs. (10)–(13) not only calculate Euclid distances by means of each S_i, but also recursively renew the mean point m_i depending on the cost function *V*. After calculations, 10 servers with different

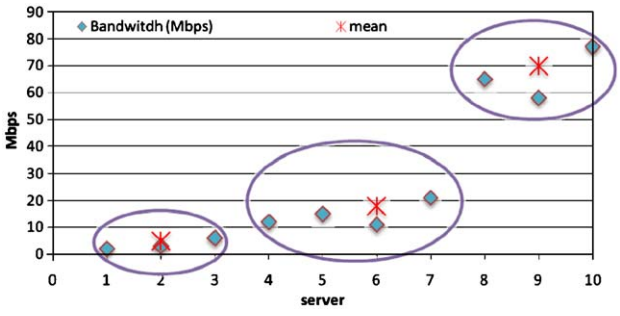

Fig. 10. 10 hosts classification according to bandwidth using *k*-means algorithm.

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

network bandwidths have been placed in three groups ($k = 3$). The simulation results are shown in Fig. 10:

- k: number of partitions
- x: number of points
- S_i: partition attributes form a vector space
- m_i: the mean point of all of S_i points
- $xBoolean_{ij}$: determines whether or not an x point belongs to S_i
- V: distance cost function
- d: distance between two point

$$m_i = \frac{\sum_{x \in S_i} d(x_i, m_i)}{|S_i|} \tag{10}$$

$$xBoolean_{ij} = \begin{cases} 1 \text{ if } \|x_j = S_i\|^2 \leqslant \|x_j = S_k\|^2 & \forall k \neq 1 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$V = \sum_{i=1}^{k} V_i = \sum_{i=1}^{k} \left(\sum_{k, x_j \in S_i} d(x_j, m_i) \right) \tag{12}$$

$$new(m_i) = \frac{1}{|S_i|} \sum_{k, x_j \in S_i} x_j \tag{13}$$

3.4. Burst Mode

Like many network accelerator methods, and multithreading, Burst Mode first splits one huge bandwidth into small pipelines all working at the same time. Burst Mode focuses on the fastest group of servers and can differentiate among the various candidate server network bandwidths. Second, BM chooses the faster one then others (as shown in Eqs. (10)–(13)). Ultimately, the BM has made single jobs into many, as shown in Fig. 11.

The k-means simulation results showed that fewer local replica servers are high efficiency than many remote replica servers. Accordingly, the main ideas in Burst Mode are to find the fastest server group, and to make it download via multithreading. BM also deals with cutting blocks properly for various data sets.

Burst Mode function is shown below:

- $N_i\,TCP_{BW}$: candidate server bandwidth
- FTS: the fastest group of servers

$$N_i\,TCP_{BW} = \frac{MSS}{RTT}\frac{C}{\sqrt{p}} \tag{14}$$

$$FTS = S_i \max\{S_1, S_2, \ldots, S_n\}, m_i \in S_i \tag{15}$$

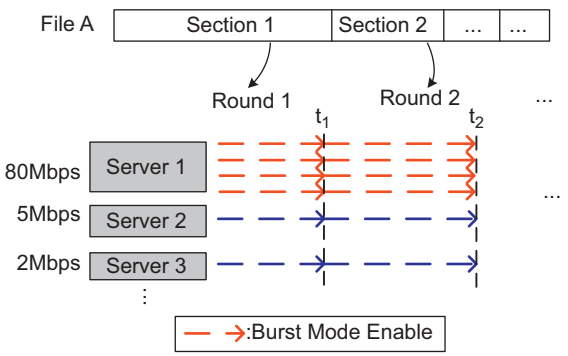

Fig. 11. Burst Mode enables higher bandwidths.

The algorithm is listed below:

[Initialization]
Measure bandwidths and find the fastest servers using Eqs. (14) and (15).
BigBlockUnit set to 100 MB initially
[Allocate blocks to the fastest servers and download via multithreading.]
Step 1: Group m_i and rank the most powerful server *FTS*
Step 2: Allocate SE_j and download via multithreading
Step 3: Monitor job progress statuses
LOOP WHEN (*UnassignedFileSize* and total *UnfinishedFileSize* are greater than *BigBlockUnit* (initial *BigBlockUnit* = 100 MB))
THEN
{
IF (Job finish rate is just 100% ($r_{ji} = 1$) and *UnassignedFileSize* and total *UnfinishedFileSize* are greater than *BigBlockUnit*)
THEN
{
Let data transfer in multiple parts between client and *FTS* server
$SE_j = (UnassignedFileSize + TotalfinishedFileSize)\alpha$, $0 < \alpha \leqslant 1$ ($UnassignedFileSize + TotalUnfinishedFileSize$)
$\geqslant BigBlockUnit$
}
}
END LOOP;

3.5. Grid network congestion control

Grid network congestion control is concerned with controlling traffic entry into data grid networks to prevent congestive collapse by avoiding oversubscription of any grid node processing or link capacity and taking resource reduction steps, such as reducing packet sending rates when Burst Mode is active.

The modern theory of congestion control (Kelly, 2003; Mamatas et al., 2007), describes how individuals controlling their own pack lost rate can interact to achieve an optimal network-wide rate allocation. Examples of "optimal rate" allocation are max–min fair allocation and Kelly's (2003) suggestion of proportional fair allocation, although many others are possible. The mathematical expression (Eq. (16)) for optimal rate allocation is as follows. Let x_i be the rate of flow i. Let x, c and R be the corresponding vectors and matrix. Let $U(x)$ be an increasing, strictly convex function, called the utility, which measures how much benefit a user obtains by transmitting at rate x. The optimal rate allocation will then satisfy:

$$\max_x \sum_i U(x_i), \quad Rx \leqslant c \tag{16}$$

3.6. Anticipative recursively adjusting mechanism plus (ARAM+)

3.6.1. Assumptions

We outline our system design model assumptions below.

- All grid nodes are installed GlobusToolkit4 previously.
- All grid nodes are supporting Simple Network Management Protocol (SNMP).
- The time for transferring, stopping/assigning processes, and calculating TCP_{BW} to selected replica servers is negligible.

3.6.2. Anticipative recursively adjusting mechanism plus (ARAM+)

The ARAM+ is not merely inherited from ARAM. It has been enhanced also in the following two areas: its TCP Bandwidth

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

Estimation Model (TCPBEM) and its Burst Mode. ARAM+ continually adjusts the workloads on selected replica servers by measuring actual bandwidth performance via TCPBEM during data file transfers and, according to previous job finish rates, and adjusting alpha values for subsequent transfer sections.

Some interesting ideas have arisen from P2P networks and distributed denial-of-service (DDoS) attacks. As is well known, P2P networking is share based; it shares data and downloads in parallel, more numbers of share point get more speedup. Another typical example is DDoS attacks that occur when multiple compromised systems flood the bandwidth or resources of a targeted system. We have combined these elements in our approach. The multithreading in the Burst Mode design came from DDoS attacks, BM "floods" the target replica server bandwidth to speed up download performance. The other idea from P2P networking was applied to ARAM+. It pre-selects many candidate replicas from various servers, then chooses appropriate servers and allocates only enough workload to fit server capacities.

Both of our previous works (Vazhkudai et al., 2001; Wang et al., 2006; Yang et al., 2005, 2007b, in press), the anticipative recursively adjusting mechanism and recursively adjusting mechanism (RAM) were based on co-allocation architecture and relied on tuning alpha values by hand to adapt to specific data grid situations. The ARAM+ uses the same strategies, but differs in that alpha values are tuned dynamically.

ARAM+ adapts to real-time network statuses and calculates appropriate alpha α values continually with TCPBEM $TotalTCP_{BW}$, to ensure good download flexibility and to speed up overall performance. The equations are as follows:

•$TotalTCP_{BW}$: overall bytes transmitted per second

$$TotalTCP_{BW} = \sum_{i-1}^{N} \frac{MSS}{RTT} \frac{C}{\sqrt{p}} \tag{17}$$

$$\alpha = 1 - \left(\frac{1}{TotalTCP_{BW}^{0.2}} \right), \quad 0 < \alpha \leqslant 1 \tag{18}$$

3.6.3. ARAM+ algorithm

[Initialization]

Current bandwidths for all candidate servers are measured using the TCP Bandwidth Estimation Model (TCPBEM) and calculating appropriate alpha values with Eqs. (14) and (15).

[Allocating blocks to selected servers]
LOOP WHEN (*UnassignedFileSize* and total *UnfinishedFileSize* is greater than zero)
THEN
{
IF (UnassignedFileSize and Total *UnfinishedFileSize* are greater than $TotalTCP_{BW}$)
THEN
{*
IF (*UnassignedFileSize* and Total *UnfinishedFileSize* multiplied by α are greater then $TotalTCP_{BW}$)
THEN
{
Define new section for allocation
$SE_j = (UnassignedFileSize + TotalUnfinishedFileSize)\alpha$, $0 < \alpha \leqslant 1$
}
ELSE
{
Define finial section
$SE_j = UnassignedFileSize + TotalUnfinishedFileSize$
}
}
END LOOP;
Step 1: Define new section for allocation SE_j
Step 2: Monitor all selected replica servers
Step 3: Allocate blocks to selected replica servers, according to the TCP_{BW} of the selected replica server, and the previous finish rates R_{j-1} for the selected replica server (initial $R_0 = 1$)
Step 4: Monitor all download flows
LOOP WHEN (The fastest flow finishes its assigned data blocks)
THEN
{
IF (First finish time for RT_{ji} is earlier then expected time ET_{ji} and transferred size TS_{ji} is greater than expected size S_{ji}) THEN
{
The $r_{ji} = 1$
}
ELSE
{
Measure the finish rate for the previously delivered file ($0 \leqslant r_{ji} \leqslant 1$)
}

$$r_{ji} = \begin{cases} \dfrac{TS_{ji}}{S_{ji}}, & RT_{ji} \geqslant Et_{ji} \\ 1, & RT_{ji} < ET_{ji}, \text{ and } TS_{ji} \geqslant S_{ji} \end{cases}$$

}
END LOOP;

4. Experimental

4.1. Our grid environment: Tiger grid

The experiments in this work were conducted and evaluated on the TigerGrid, which consists of more than 100 processors distributed over 10 clusters located at 5 educational institutions (Tunghai University—THU, National Taichung University—NTCU, Hsiuping Institute of Technology—HIT, National Dali Senior High School—DALI, Lizen High School—LZSH, and Tungs' Taichung Metro Harbor Hospital—TUNG). A logical diagram of the Tiger grid network environment is shown in Fig. 12. Fig. 13 shows statuses for all machines used in the grid testbed on one monitor page.

They are interconnected by the 1 Gbps Taiwan Academic Network (TANET). The Tiger grid platform is built around 60 computing nodes, more than 224 CPUs with differing speeds, and total storage of more than 5 TB. All the institutions are in Taiwan, at least 10 km from THU. All machines have Globus 4.0.7 or above installed.

We performed wide-area data transfer experiments using Cyber Transformer, our GridFTP GUI client tool, on our co-allocation testbed at Tunghai University (THU), Taichung City, Taiwan, and fetched files from replica servers at National Da-Li Senior High School (DL), Li-Zen High School (LZ), Tungs' Taichung Metro Harbor Hospital (TUNG), and Hsiuping Institute of Technology School (HIT). These institutions are all in Taichung, Taiwan, 10–30 km from THU.

4.2. Our experimental tool: Cyber Transformer

In a previous work Yang et al. (2006), we gave experimental results for Cyber Transformer, a powerful new toolkit for replica

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

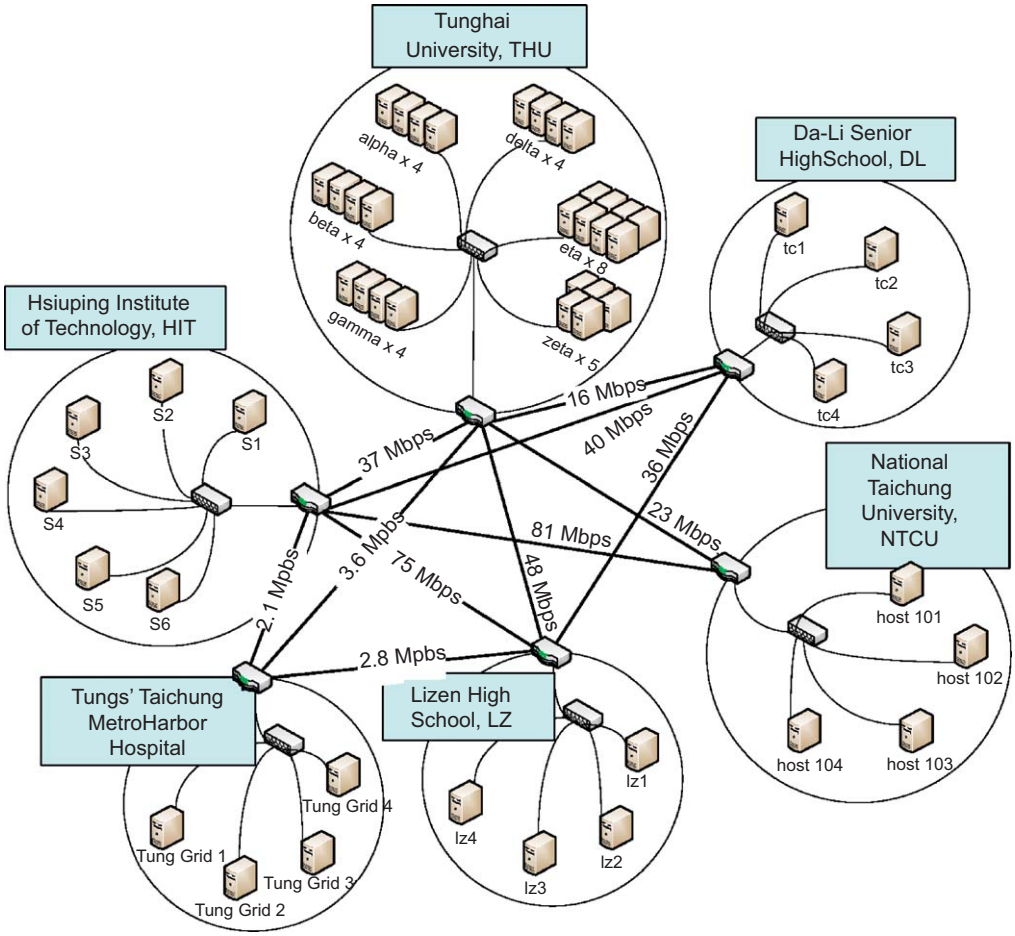

Fig. 12. Tiger grid network.

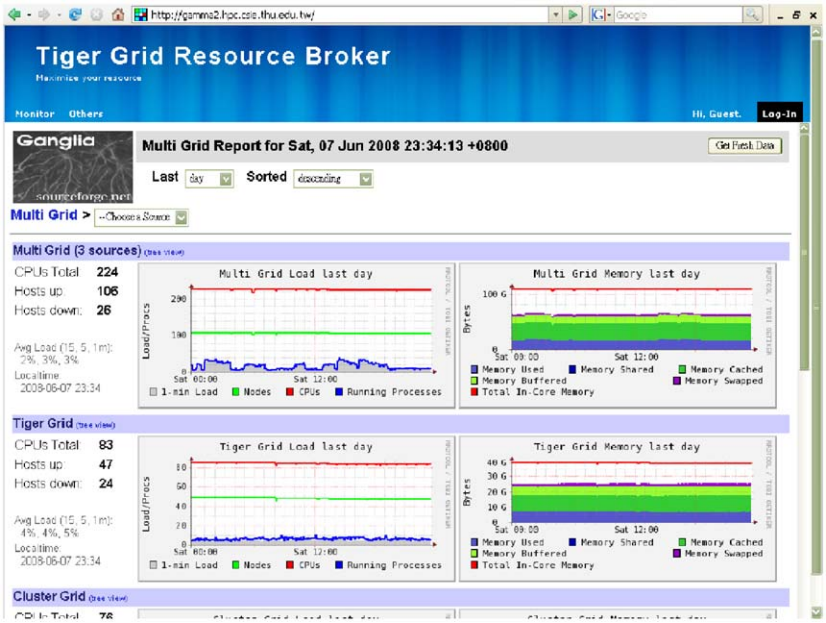

Fig. 13. Tiger grid resources.

management and data grid environment data transfers. It can accelerate data transfer rates, and also manage replicas over 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 source host names. When users search for files using logical file names, Cyber Transformer queries the Replica Location Services to find all corresponding replicas, and

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

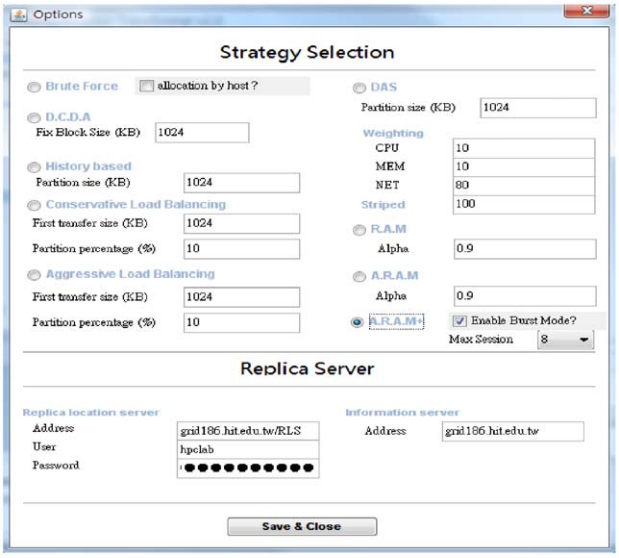

Fig. 14. Parallel download strategy selection.

directs the replica sources to start parallel transfers. Cyber Transformer users can easily gather replica resources and combine them into single entities with the "strategy selection" user interface, accomplishing the task with various parallel download strategies, as shown in Fig. 14.

4.3. Experimental results and analyses

An experiment and a case design were devised to test Burst Mode, our proposed approach to speeding up local and remote performance, and dynamically adjusting alpha values to adapt to variable network situations. Details of the test cases we designed are shown in Fig. 15.

4.3.1. Case study—"cross-grid" vs. "local grid" replica selects and transfers

We designed two scenarios to verify the efficiency of enabling Burst Mode. All test cases are listed in Tables 1 and 2.

Generally, more replicas and local placement will yield better parallel file transfer performance. Our results, shown in Figs. 16 and 17, show that we found more replicas remotely so user performance improvement was not obvious, even worse than the few replica found locally. However, Burst Mode function could get more performance even two copies only (refer to scenario: Rx2_local).

4.3.2. Case study—RAM and ARAM vs. ARAM+

RAM (Yang et al., 2007c) and ARAM (Yang et al., 2007a) both used constant alpha values; our approach, ARAM+, relied on dynamic alpha values to adapt to data grid network link fluctuations. The case study for RAM and ARAM is listed in Table 4. We set the constant alpha values at 0.9, 0.5, and 0.1 for comparison with ARAM+, and replicas were selected from inside and outside regions. In order to distinguish among replica locations, these two kinds of replica selection plans are listed in Table 3.

In our next experiment, two scenarios, sets A and B, are listed in Table 4 and used to accentuate the advantages of the Burst Mode method and dynamic alpha value adjustment. Overall performances in Scenario B have obviously been improved over those in Scenario A. The total amounts of TCP bandwidth in

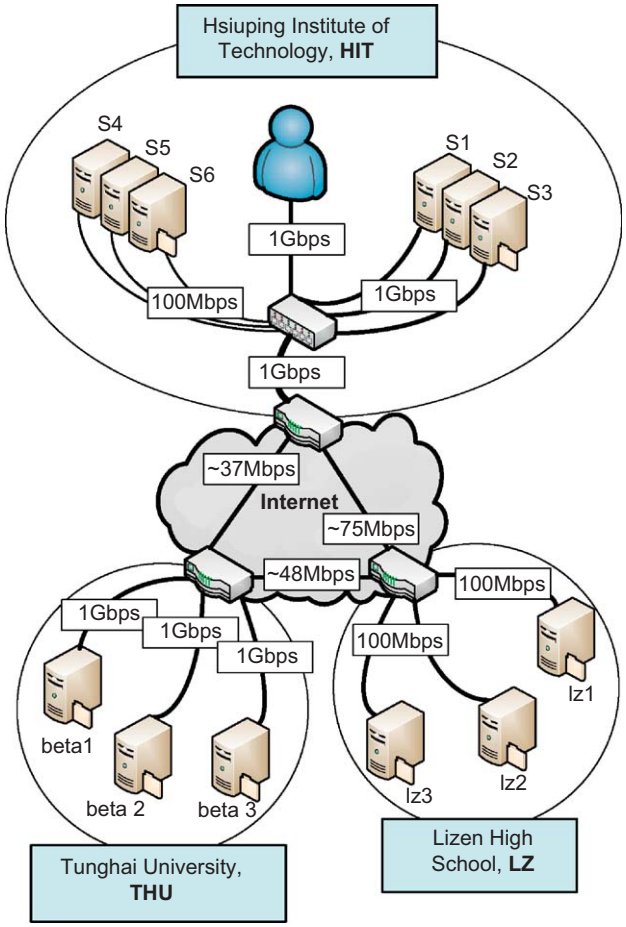

Fig. 15. Scenarios for our test-bed of Tiger grid.

Table 1
Scenario for replica local or not.

Scenario	Replica server list
ARAMplus_4: non-local	THU-S1, S2; LZ1, 2
ARAMplus_4: local-1	HIT-S1, S2; THU-beta1, beta2
ARAMplus_4: local-2	HIT-S1, S2; THU-beta 1; LZ-1
ARAMplus_4: local-3	HIT-S1, S2; LZ-1, 2
ARAMplus_4: all-local	HIT-S1, S2, S3, S4

Table 2
Scenario for various replica numbers and selections.

Scenario	Replica server list
R × 6_non-local	LZ-1, 2, 3; THU-beta 1, beta 2, beta 3
R × 6_local	HIT-S1, S2, S3, S4, S5, S6
R × 2_local	HIT-S1, S2
R × 2_non-local-THU	THU-S1, S2
R × 2_non-local-LZ	LZ-S1, S2

Scenario A differed slightly, but there were significant differences in Scenario B. In all these case studies, especially in Scenario B, Burst Mode yielded huge performance improvements, as shown in Figs. 18 and 19.

4.3.3. Case study—comparison of 9 co-allocation schemes

To evaluate the performance of our proposed technique, we implemented the following nine co-allocation schemes: Brute-

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

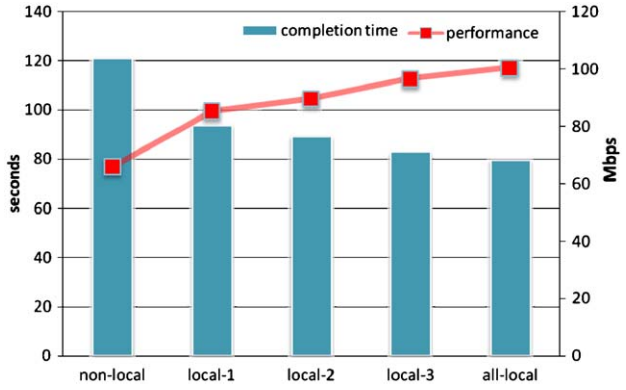

Fig. 16. Effects of various replica locations on performance results.

Fig. 17. Effects of various replica numbers and selections on performance results.

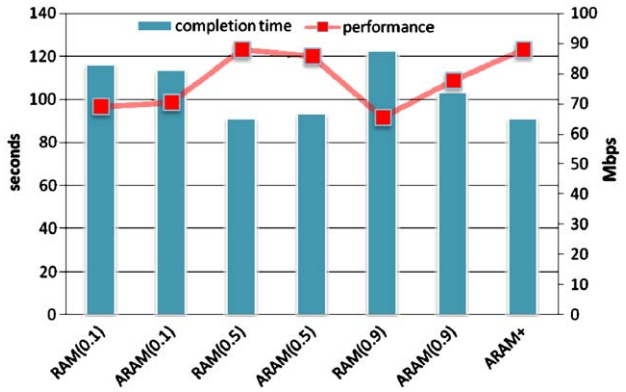

Fig. 18. Performance results for scenario A.

force (Brute), history-based (history), conservative load balancing (conservative), aggressive load balancing (aggressive), dynamic co-allocation with duplicate assignments (DCDA), recursively adjusting mechanism (RAM), dynamic adjustment strategy (DAS), anticipative recursively adjusting mechanism (ARAM), and anticipative recursively adjusting mechanism plus (ARAM+). Using the case setups listed in Table 3 for each scheme, we analyzed their performance by comparing transfer finish times and overall performance, as shown Figs. 20 and 21.

We found that ARAM+ performed better than the others. An interesting outcome shows the Brute scheme's "local" performance differed greatly from its "mixed" performance. ARAM+ is

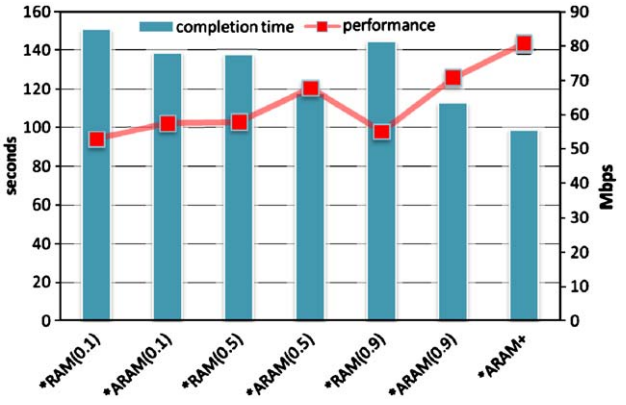

Fig. 19. Performance results for scenario B.

Table 3
Replica placement and selection plan.

Mix	HIT-S1, S2; LZ-1, 2; THU-beta1, beta2
Local	HIT-S1, S2, S3, S4, S5, S6

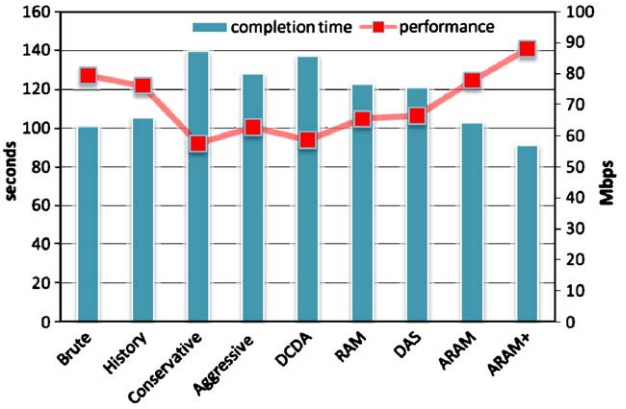

Fig. 20. Comparing 9 schemes on "local" cases.

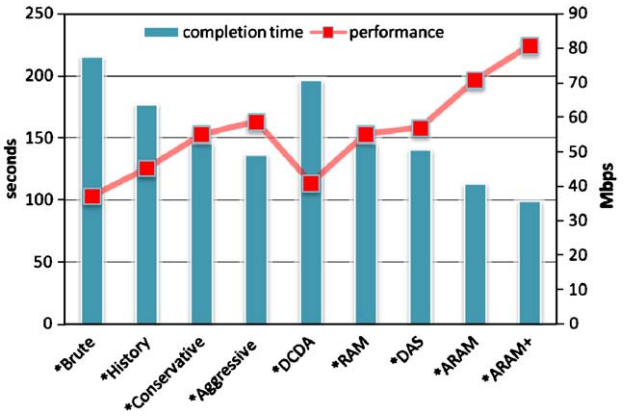

Fig. 21. Comparing 9 schemes on "mixed" cases.

comparable to Brute or any others. The advantages of ARAM+ are the following:

- ARAM+ uses TCP bandwidth measurement technology, reliability and accuracy of the best.
- ARAM+ can enhance GridFTP to become multiplexing.

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

Table 4
Scenario for alpha value tuning.

Scenario A	Scenario B
RAM(0.1)_local	RAM(0.1)_mix
ARAM(0.1)_local	ARAM(0.1)_mix
RAM(0.5)_local	RAM(0.5)_mix
ARAM(0.5)_local	ARAM(0.5)_mix
RAM(0.9)_local	RAM(0.9)_mix
ARAM(0.9)_local	ARAM(0.9)_mix
ARAM+_local	ARAM+_mix

- ARAM+ used *k*-means for classifying numbers grid node. It quickly finds out the most efficient computing nodes.
- ARAM+ gives the longest amount of computing job to powerful grid node but small data set could ignore some advance option, for example, dynamic α, server classification (*k*-mean) algorithm and congestion control.
- ARAM+ can really adapt to different grid environments, rather than to just specific experiments designed grid system.

5. Conclusion

Co-allocation architectures can be used to enable parallel transfers of data files from multiple replicas in data grids, which mean all replicas stored in the various grid sites. Many schemes based on the Co-Allocation Model 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 these schemes, the applicable piece fragments achieve more performance. In fact, some schemes can be applied to specific situations; however, most situations are not common actually. For this issue, we propose the anticipative recursively adjusting Mechanism plus (ARAM+), based on ARAM. The best part is performance tuning through continual dynamic alpha value adjustment. It relies on special features to adapt to various network situations in data grid environments. The TCP Bandwidth Estimation Model was used to evaluate dynamic link states in our experiments by detecting TCP throughputs and packet lost rates between grid nodes. TCP Bandwidth Estimation Model also can be more reliable and fair than ARAM and any other scheme. Burst Mode function truly can increase transfer rates and speed up total performance especially considering congestion control. The ARAM+ not only adapts to the worst network links, but also speeds up the overall performance especially in wide-area grid networks.

References

Allcock B, Bester J, Bresnahan J, Chervenak A, Foster I, Kesselman C, et al. Data management and transfer in high-performance computational grid environments. Parallel Computing 2002;28(5):749–71.

Altman Eitan, Barman Dhiman, Tuffin Bruno, Vojnovic Milan. Parallel TCP sockets: simple model, throughput and validation. In: INFOCOM 2006, April 2006.

Bhuvaneswaran RS, Katayama Y, Takahashi N. Dynamic co-allocation scheme for parallel data transfer in grid environment. In: Proceedings of first international conference on semantics, knowledge, and grid (SKG 2005), 2005. p. 17.

Bhuvaneswaran RS, Katayama Y, Takahashi N. A framework for an integrated co-allocator for data grid in multi-sender environment. IEICE Transactions on Communications 2007;E90-B(4):742–9.

Bolliger Juerg, Gross Thomas, Hengartner Urs. Bandwidth modelling for network-aware applications. In: INFOCOM '99, March 1999.

Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S. The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. Journal of Network and Computer Applications 2001;23(3):187–200.

Chervenak A, Deelman E, Foster I, Guy L, Hoschek W, Iamnitchi A, et al. Giggle: a framework for constructing scalable replica location services. In: Proceedings of the 2002 ACM/IEEE conference on supercomputing, November 2002. p. 1–17.

Czajkowski K, Foster I, Kesselman C. Resource co-allocation in computational grids. In: Proceedings of the eighth IEEE international symposium on high performance distributed computing (HPDC-8 '99), August 1999.

Czajkowski K, Fitzgerald S, Foster I, Kesselman C. 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. p. 181–94.

Foster I, Kesselman C. Globus: a metacomputing infrastructure toolkit. International Journal of High Performance Computing Applications 1997;11(2):115–28.

Foster I, Kesselman C, Tuecke S. The anatomy of the grid: enabling scalable virtual organizations. International Journal of High Performance Computing Applications 2001;15(3):200–22.

Hacker Thomas J, Athey Brian D. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network, parallel and distributed processing symposium. In: Proceedings international, IPDPS 2002, 10.1109/IPDPS.2002.1015527.

Hoschek W, Jaen-Martinez J, Samar A, Stockinger H, Stockinger K. Data management in an international data grid project. In: Proceedings of the first IEEE/ACM international workshop on grid computing-grid 2000, Bangalore, India, December 2000.

Kelly Frank. Fairness and stability of end-to-end congestion control. European Journal of Control 2003:159–76.

Mamatas Lefteris, Harks Tobias, Tsaoussidis Vassilis. Approaches to congestion control in packet networks. Journal of Internet Engineering 2007;1(1):2.

Mathis M, Semke J, Mahdavi J, Ott T. The macroscopic behavior of the TCP congestion avoidance algorithm. Computer Communication Review 1997;27(3).

Open Grid Forum. 〈http://www.ogf.org/〉.

Padhye J, Firoiu V, Towsley D, Kurose J. Modeling TCP throughput: a simple model and its empirical validation. In: ACMSIGCOMM, September 1998.

Stockinger H, Samar A, Allcock B, Foster I, Holtman K, Tierney B. File and object replication in data grids. Journal of Cluster Computing 2002;5(3):305–14.

The Globus Alliance. 〈http://www.globus.org/〉.

Vazhkudai S. Enabling the co-allocation of grid data transfers. In: Proceedings of fourth international workshop on grid computing, 17 November 2003. p. 44–51.

Vazhkudai S, Schopf J. Predicting sporadic grid data transfers. In: Proceedings of 11th IEEE international symposium on high performance distributed computing (HPDC-11 '02), July 2002. p. 188–96.

Vazhkudai S, Schopf J. Using regression techniques to predict large data transfers. International Journal of High Performance Computing Applications (IJHPCA) 2003;17(3):249–68.

Vazhkudai S, Tuecke S, Foster I. Replica selection in the globus data grid. In: Proceedings of the first international symposium on cluster computing and the grid (CCGRID 2001), May 2001. p. 106–13.

Vazhkudai S, Schopf J, Foster I. Predicting the performance of wide area data transfers. In: Proceedings of the 16th international parallel and distributed processing symposium (IPDPS 2002), April 2002. p. 34–43.

Venugopal S, Buyya R, Ramamohanarao K. A taxonomy of data grids for distributed data sharing, management, and processing. ACM Computing Surveys 2006;38(1) (Article 3).

Wang CM, Hsu CC, Chen HM, Wu JJ. Efficient multi-source data transfer in data grids. In: Proceedings of the sixth IEEE international symposium on cluster computing and the grid (CCGRID '06), 16–19 May 2006. p. 421–4.

Yang CT, Chen CH, Li KC, Hsu CH. Performance analysis of applying replica selection technology for data grid environments. In: PaCT 2005, Lecture Notes in Computer Science, vol. 3603. Berlin: Springer; 2005. p. 278–87.

Yang CT, Yang IH, Chen CH, Wang SY. Implementation of a dynamic adjustment mechanism with efficient replica selection in co-allocation data grid environments. Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC 2006) – Distributed Systems and Grid Computing Track, France, April 23–27, 2006. pp. 797–804.

Yang CT, Chi YC, Han TF, Hsu CH. Redundant parallel file transfer with anticipative recursively-adjusting scheme in data grids. Distributed and Parallel Computing: 7th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP 2007, Lecture Notes in Computer Science, vol. 4494, 2007a. p. 242–53.

Yang CT, Yang IH, Li KC, Wang SY. Improvements on dynamic adjustment mechanism in co-allocation data grid environments. Journal of Supercomputing 2007b;40(3):269–80.

Yang CT, Wang SY, Fu CP. 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, September 17–20. Berlin: Springer; 2007c. p. 61–70, ISSN 1611-3349.

Yang CT, Yang MF, Chiang WC. Implementation of a cyber transformer for parallel download in co-allocation data grid environments. 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. 242–53.

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

Yang CT, Yang IH, Wang SY, Hsu CH, Li KC. A recursively-adjusting co-allocation scheme with cyber-transformer in data grids. Future Generation Computer Systems, in press (available online 21 January 2007).

Yang L, Schopf J, Foster I. Improving parallel data transfer times using predicted variances in shared networks. In: Proceedings of the fifth IEEE international symposium on cluster computing and the grid (CCGrid '05), 9–12 May 2005. p. 734–42.

Zhang X, Freschl J, Schopf J. 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. p. 270–82.

Please cite this article as: Yang C-T, et al. Enhancement of anticipative recursively adjusting mechanism for redundant parallel file transfer in data grids. J Network Comput Appl (2009), doi:10.1016/j.jnca.2009.02.002

MIFAS: Medical Image File Accessing System in Co-allocation Data Grids*

Chao-Tung Yang† Chiu-Hsiung Chen Ming-Feng Yang
High-Performance Computing Laboratory
Department of Computer Science and Information Engineering
Tunghai University, Taichung, Taiwan R.O.C.
ctyang@thu.edu.tw

Wen-Chung Chiang
Department of Information Networking Technology
Hsiuping Institute of Technology
Taichung County, Taiwan ROC
wcchiang@mail.hit.edu.tw

Abstract

We have encountered two challenges when using the PACS system. First, PACS users are limited to certain bandwidths and locations. Second, Web PACS machine replacement is too costly, management is difficult, and better image stability is needed. There are also speed variations for different users at different locations. For instance, radiologists use medical image workstations with direct access to the PACS information system and so have a greater speed for querying and file retrieval. Physicians, on the other hand, use web browsers with no direct access to the PACS information system, which leads to slower network speeds. Physicians also affect one another in overall network speed by processing queries and file retrievals via web browser. There are also insufficient network bandwidth concerns. These often arise when exchanging medical images with other hospitals or downloading large numbers of images. Since these large file volumes are transferred via WANs, insufficient network bandwidths limit upload and download speeds. And if the Web PACS breaks down, the hospital must ask professional engineers for replacements, and spend large amounts of money about a NT$ million or more per unit. This is not only troublesome for system managers, but also costly for the hospital.

1. Introduction

Hospitals are increasingly acquiring 2D, 3D, and 4D medical imaging devices, along with treatment and surgery simulators. These produce image files ranging from several MB to several hundred MB. High-level medical images, such as 64/128-slice CT scans, 3.0T MRI, and PET often exceed one hundred MB or more.

Advancing technology has led to the development of many high-quality imaging devices, resulting in the collection of massive amounts of medical image files unaccompanied by sufficient handling infrastructure. Consequently, Picture Archiving and Communication Systems (PACS) are unable to provide efficient query response services. Processing queries and file retrievals causes slowdowns in the overall Web PACS network. And conventional access methods for large numbers of image exchanges and downloads affect transfer times where bandwidth is limited.

Today, the related issues of medical imaging and file transfer speed demand attention to viewing and processing images and videos [1, 3].

We first present our new method for processing medical image queries, which is based on the Co-allocation [13, 14, 15, 16] strategy for data grid environments. A data grid is defined as a collaboration of distributed resources across institutional borders. The system we designed and implemented is called the Medical Image File Accessing System for Co-allocation Data Grids (MIFAS). It uses the Co-allocation model to get images in parallel download [7] on independently or another site.

The remainder of this paper is organized as follows. Background review and studies are presented in Section 2. Cyber Agent Transformer design and implementation are given in Section 3. Medical image, experiments, results and performance evaluation are presented in Section 4. Section 5 concludes this research article.

2. Background

* This work is supported in part by the 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.
† The corresponding address

978-0-7695-3473-2/08 $25.00 © 2008 IEEE
DOI 10.1109/APSCC.2008.267

2.1. Medical Images

Medical images usually provide human body information to assist disease diagnosis. Medical imaging refers to the techniques and processes that use special equipments to create images of different body areas for clinical purposes (medical procedures seeking to reveal, medical diagnose or examined disease) or medical science study (including normal anatomy and function). As a discipline and in its widest sense, it is part of biological imaging and incorporates radiology, radiological sciences, endoscopy, thermography, medical photography and microscopy (e.g. for human pathological investigations).

Measurement and recording techniques, which are not primarily designed to produce images, such as electroencephalogram (EEG), magneto-encephalography (MEG) and others, but for data susceptible to be represented as maps, can be seen as forms of medical imaging.

In clinical applications, medical imaging is also known as radiology or "clinical imaging". Diagnostic radiography designates the technical aspects of medical imaging and especially the acquisition of medical images. The radiologic technologists or physicians are responsible for acquiring medical images of diagnostic quality, and performing radiological interventions.

In the fields of Medicine, Medical Engineering, Medical Physics and Bioformatics, Medical Imaging is usually defined as the technology of image formation, retrieval and storage with the research and development of instrumentation. As for the research on medical image application and interpretation, it is classified as radiology, other relevant medical sub-disciplines, or areas of medical science (neuroscience, cardiology, psychology, and etc.) Many techniques developed for medical imaging also have scientific and industrial applications.

Medical imaging is often perceived to designate the set of techniques that noninvasively produce images of the internal body. In this restricted sense, medical imaging can be seen as the solution of mathematical inverse problems. This means that cause is inferred from the observed signal. In the case of ultrasonic device, the probe consists of ultrasonic pressure waves and echoes inside the tissue to show the structure of internal body. In the case of projection radiography, the probe is X-ray radiation which is absorbed at different rates in different tissue types such as bone, muscle and fat.

2.2. Data Grid

Grid computing or grid clusters is a technology closely related to cluster computing. The key differences (by definitions which distinguish the two at all) between grids and traditional clusters are that grids connect collections of computers which do not fully trust each other, or which are geographically dispersed. Grids are thus more like a computing utility than like a single computer. In addition, grids typically support more heterogeneous collections than are commonly supported in clusters.

Grid Computing started as a generalization of Cluster Computing, promising to deliver large scale levels of parallelism to high-performance applications by crossing administrative boundaries. Moreover, the use of computational and data resources in high-performance applications, undertaken over Grid infrastructure, have started to become a reality. Today, we face the large challenge of making on-demand access to any computational service.

2.3. Co-allocation Model

The architecture proposed [10, 11, 13] consists of three main components: an information service, a broker/co-allocator, and local storage systems. Figure 1 shows co-allocation of Grid Data transfers, which is an extension of the basic template for resource management [3] provided by the Globus Toolkit. Applications specify the characteristics of desired data and pass attribute descriptions to a broker. The broker searches for available resources, and gets replica locations from the Information Service [2] and Replica Management Service [8] and replica selection [9, 12]; then, obtains the lists of physical file locations.

We have implemented the following eight co-allocation schemes: Brute-Force (Brute), History-based (History), Conservative Load Balancing (Conservative), Aggressive Load Balancing (Aggressive) [6], Dynamic Co-allocation with Duplicate Assignments (DCDA), Recursively-Adjusting Mechanism (RAM), Dynamic Adjustment Strategy (DAS), and Anticipative Recursively-Adjusting Mechanism (ARAM).

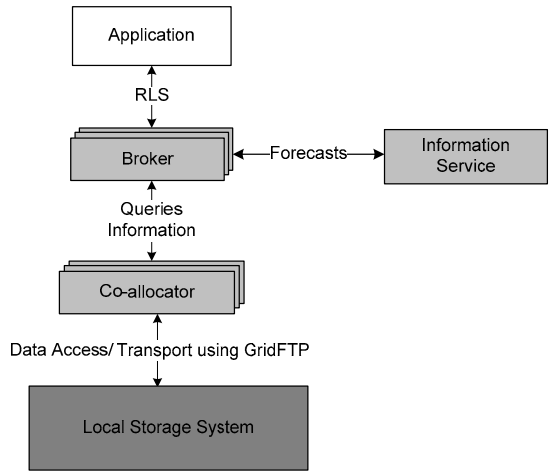

Figure 1: Data Grid Co-allocation Architecture

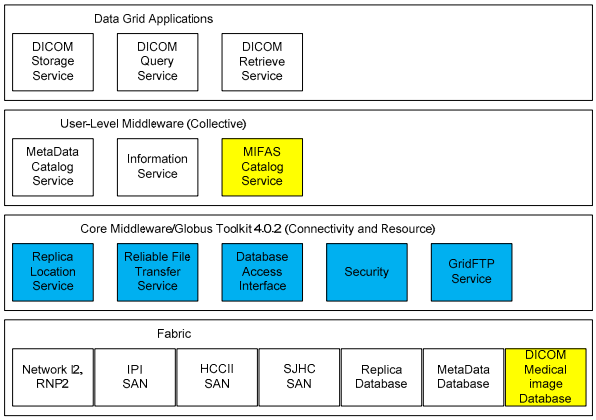

Figure 2: Overview Architecture of MIFAS in Co-allocation Data Grid. Yellow: developed at HPCLab; Blue: Globus Toolkit

2.4 Image Processing Program ImageJ

ImageJ [4] is a public domain image processing software based on Java. It is developed by the National Institutes of Health. ImageJ can run on Windows, Mac OS, Mac OS X, Linux, and Sharp PDA, and other platforms.

This application can display, edit, analyze, process, save and print 8-bit, 16-bit and 32-bit images. It can read many image formats including TIFF, GIF, JPEG, BMP, DICOM, FITS and raw. It supports stacks, a series of images that share a single window. It is multithreaded, so time-consuming operations such as image file reading can be performed in parallel with other operations.

ImageJ is a free open source software, supporting custom upgrade, edit and plug-in. ImageJ has built-in editor and java compiler. With any IDE, users can directly process images using ImageJ.

3. System Design and Implementation

3.1. System Architecture

Our proposed solution, MIFAS in Co-allocation Data Grid, was developed using grid computing technology, and integrating Co-allocation with Globus Toolkit 4.xx. We incorporated desktop PCs and servers in the data grid, then used them to run the data grid components. In previous experiments, data grid nodes used high-speed network bandwidth. As recommended, we proposed a system architecture that does not interfere with theirs. Descriptive medical image information (metadata) about logical data items is stored in the MIFAS Catalog Service. The four-layer architecture of the Data Grid is shown in Figure 2.

3.2. System Flow

3.2.1. System Workflow

Our design for the Co-allocation grid is as shown in Figure 3. Every Client node access point uses Cyber Agent to enter the Co-allocation data grid and manage queries and image retrievals, as with the Web-based Enquiries PACS. Overall, the greatest benefit of our method is that it speeds up query accesses and image retrievals. It also provides security for queries and image retrievals in the data grid environment.

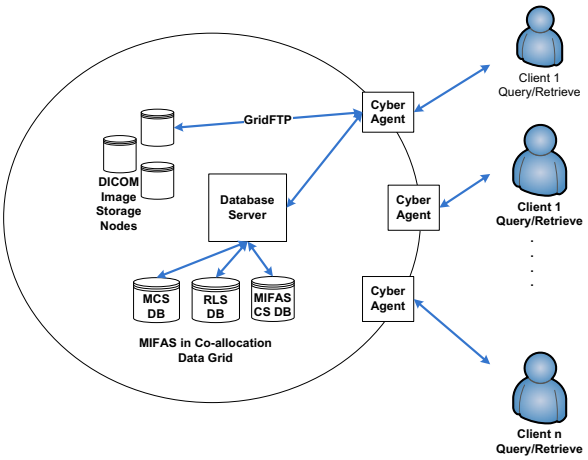

Figure 3: Workflow Overview of Medical Image in Co-allocation Data Grid

3.2.2. The Cyber Agent Transformer

In a previous work [15], we gave experimental results for Cyber Agent Transformer, a powerful new toolkit for replica management and data transfers in data grid environments. It not only accelerates data

transfer rates, but also manages replicas over 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 Agent Transformer can be invoked with either the logical file name of a data file or a list of replica source host names. When users search for files by logical file name, Cyber Agent Transformer searches Replica Location Services to find all corresponding replicas, and notifies each source to start parallel transfers. The file is then gathered from replica sources and finally combined into a single file.

3.3. GUI and System Operations

We developed a user-friendly GUI for Cyber Agent Transformer to help users unfamiliar with downloading and managing files in data grid environments. It was implemented in the Java CoG library, and can be run on any operating system with JVM. The entire set up and operation process is shown below. Such as authentication setup, strategy selection, user tools, message box.

We designed tools to assist users in downloading medical image files, and setting up some environment configurations. Figure 4 shows the user tools.

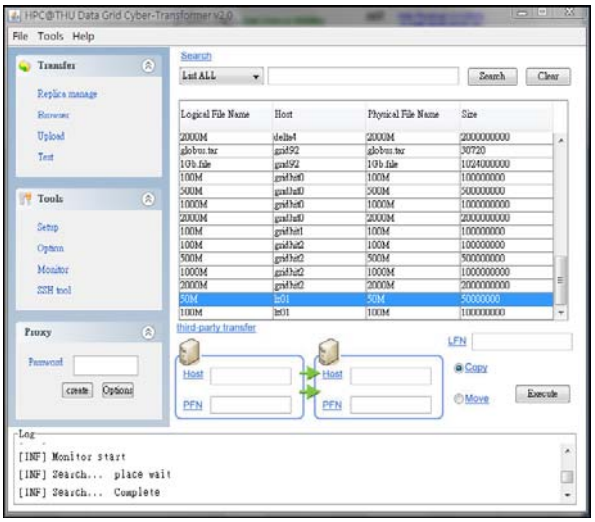

Figure 4: User Tools

4. Experimental Environments and Results

4.1. Cross-hospital PACS Architecture

Using the TIGER Grid system, we tried to simulate a PACS system serving two or more hospitals, and performed several experiments on issues of concern.

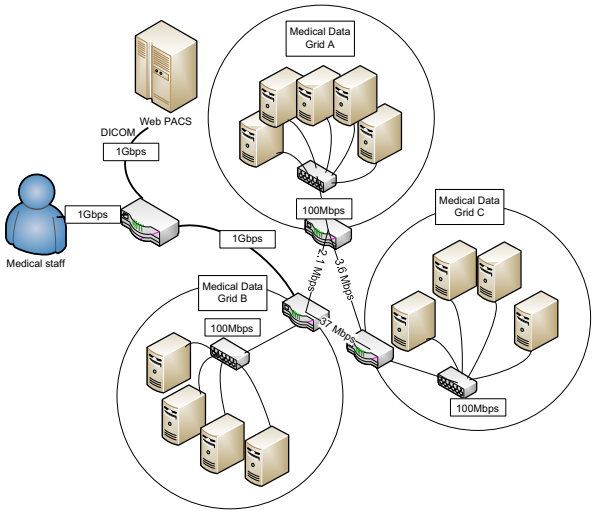

Figure 5: Cross-hospital PACS Architecture

4.2 Compare query and retrieve using ARAM in local Grid node and Web PACS

In this experiment, we simulated a Web PACS system in a local Grid node and used the best transfer method, ARAM, to do other comparison tests. Physicians may need to search and retrieve the files listed in Table 1 for diagnosis or to compare medical cases. These files are usually X-Ray images, CT scans, or series' of CT scans. In order to compare the difference in data retrieval performance between the Web PACS system and Cyber Agent Transformer, we customized test-bed A, as shown in Figure 6. Cyber Agent Transformer retrieved images via parallel-download from Medical Data Grid B (Data Flow B, Figure 6), whereas the Web PACS system retrieved from the Web PACS (Data Flow A, Figure 6). The times for the MIFAS Co-allocation ARAM and Web PACS are shown in Figures 7 and 8. The results show the performance in end-to-end query and retrieval of the first medical image by ARAM was better for all sizes than Web PACS. And the average transfer time was better than Web PACS. We then tested retrieving Image J from the Medical Data Grid, as shown in Figure 9.

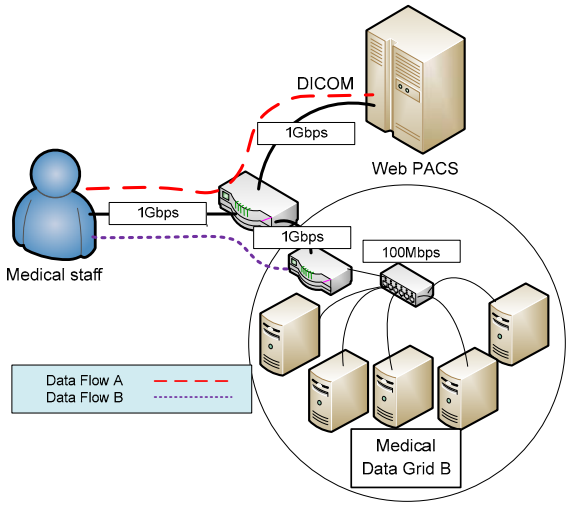

Figure 6: Our Test-bed A

Table 1: Query and Retrieve for X-Ray and CT

Image Query and Retrieve	Image Data
CT 42 512*512	~22MB
X-Ray Chest 5 2320*2828	~65MB
A series of CT case 180 512*512	~79MB

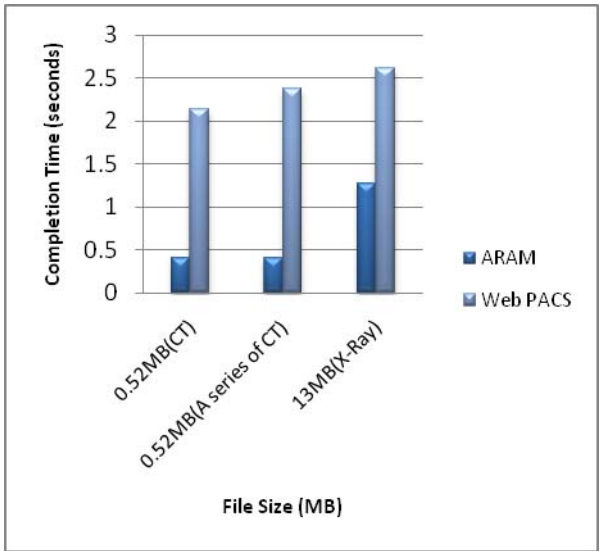

Figure 7: Compare Query/Retrieval Times for the First Image from Local Grid Node and Web PACS using ARAM

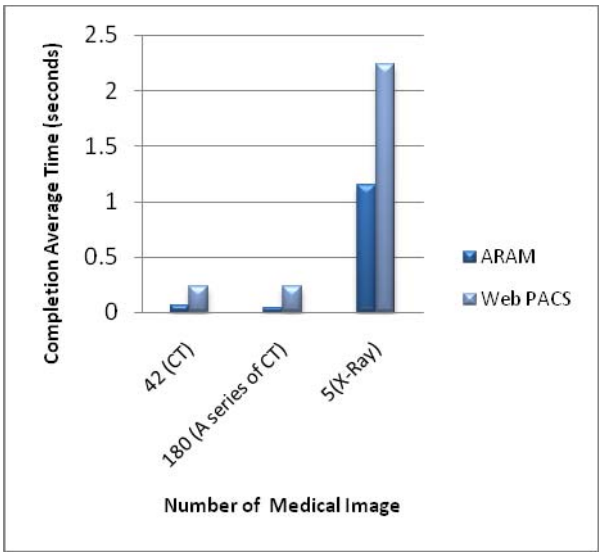

Figure 8: Compare Average ARAM Transfer Times from Local Grid Node and Web PACS

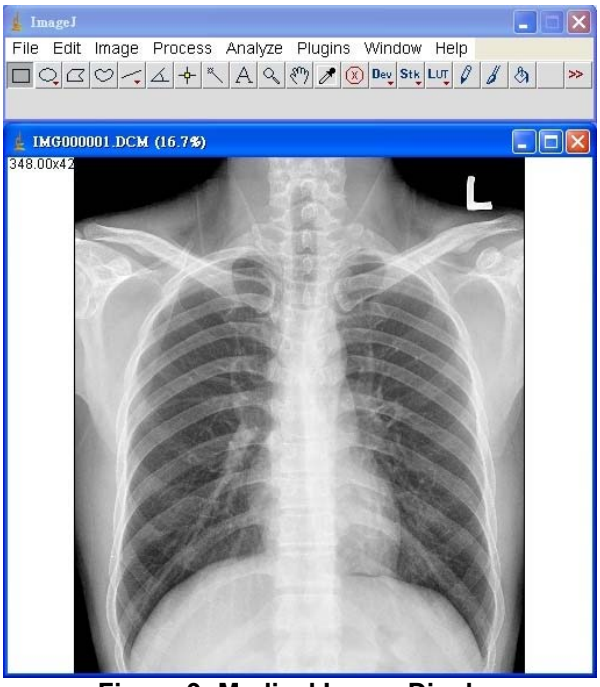

Figure 9: Medical Image Display

5. Conclusions

We can enhance quality of two important aspects of the overall health care environment. For users, we provide a fast, secure, stable, reliable system for obtaining medical images. Co-allocation architecture enables parallel downloading from a data grid. It can also speed up downloads and overcome network faults. For managers, we provide easy management, reduced expense, and increased medical image system stability.

In this paper, we reported on successfully moving medical images on the MIFAS Co-allocation data grid. We proposed a means of integrating a medical image file accessing system with a co-allocation data grid to improve medical image query, retrieval, exchange, and download speeds. Our user experiments showed ARAM to be the best among the eight Co-allocation schemes. We found that parallel downloading via File Transfer Protocols yields better performance than single-point downloading. ARAM also overcomes the problem of broken network links. It completes transfer jobs by continuing from the previous point.

Furthermore, we enhanced security with a data grid authentication environment: User Certificate, Private Key, Certificate Authority (CA) File, and Proxy File. In conclusion, Medical Image File Accessing in a Co-allocation Data Grid provides users with a reliable and secure environment for processing queries and medical image retrievals efficiently.

References

[1] V. Breton, R. Medina, J. Montagnat, "Datagrid, prototype of a biomedical Grid", Methods Inf. Med., vol. 42, no. 2, pp. 143-147, 2003.

[2] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, and M. Ripeanu, B. Schwarz, H. Stockinger, K. Stockinger, and B. Tierney. "Giggle: A Framework for Constructing Scalable Replica Location Services," in Proc. SC, pp. 1-17, 2002.

[3] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets," Journal of Network and Computer Applications, 23(3), pp. 187-200, 2001.

[4] ImageJ ,. http://rsb.info.nih.gov/ij/

[5] N. E. King , B. Liu , Z. Zhou , J. Documet , H.K. Huang "The Data Storage Grid: The Next Generation of Fault-Tolerant Storage for Backup and Disaster Recovery of Clinical Images" Medical Imaging 2005: PACS and Imaging Informatics, edited by Osman M. Ratib,Steven C. Horii, Proceedings of SPIE Vol. 5748, pp. 208-217, 2005.

[6] The Globus Alliance, http://www.globus.org/

[7] S. Vazhkudai, J. Schopf, and I. Foster, "Predicting the Performance of Wide Area Data Transfers," Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), pp. 34-43, April 2002.

[8] S. Vazhkudai, "Enabling the Co-Allocation of Grid Data Transfers," Proceedings of Fourth International Workshop on Grid Computing, pp. 44-51, 17 November 2003.

[9] S. Vazhkudai, S. Tuecke, and I. Foster, "Replica Selection in the Globus Data Grid," Proceedings of the 1st International Symposium on Cluster Computing and the Grid (CCGRID 2001), pp. 106-113, May 2001.

[10] C.T. Yang, Y.C. Chi, T.F. Han and C.H. Hsu, "Redundant Parallel File Transfer with Anticipative Recursively-Adjusting Scheme in Data Grids", ICA3PP 2007, pp. 242–253, 2007.

[11] C.T. Yang, C.H. Chen, K.C. Li, and C.H. Hsu, "Performance Analysis of Applying Replica Selection Technology for Data Grid Environments," PaCT 2005, Lecture Notes in Computer Science, vol. 3603, pp. 278-287, Springer-Verlag, September 2005.

[12] L. Yang, J. Schopf, and I. Foster, "Improving Parallel Data Transfer Times Using Predicted Variances in Shared Networks," Proceedings of the fifth IEEE International Symposium on Cluster Computing and the Grid, (CCGrid '05), pp. 734-742, 9-12 May 2005.

[13] C.T. Yang, S.Y. Wang, and C.P. Fu, "A Dynamic Adjustment Mechanism for Data Transfer in Data Grids," Network and Parallel Computing: IFIP International Conference, NPC 2007, Lecture Notes in Computer Science, vol. 4672, pp. 61-70, Springer, ISSN 1611-3349, September 17-20, 2007.

[14] C.T. Yang, S.Y. Wang, C.H. Lin, M.H Lee, and T.Y. Wu, "Cyber Transformer: A Toolkit for Files Transfer with Replica Management in Data Grid Environments," Proceedings of the Second Workshop on Grid Technologies and Applications (WoGTA'05), pp. 73-80, December 2005.

[15] C.T. Yang, I.H. Yang, K.C. Li, and C.H. Hsu "A Recursively-Adjusting Co-Allocation Scheme in Data Grid Environments," ICA3PP 2005 Algorithm and Architecture for Parallel Processing, Lecture Notes in Computer Science, vol. 3719, pp. 40-49, Springer-Verlag, October 2005.

[16] C.T. Yang, I.H. Yang, K.C. Li, and S.Y. Wang, "Improvements on Dynamic Adjustment Mechanism in Co-Allocation Data Grid Environments," The Journal of Supercomputing, Springer, vol. 40, no. 3, pp. 269-280, 2007.

2008 14th IEEE International Conference on Parallel and Distributed Systems

Enhancement of Anticipative Recursively-Adjusting Mechanism for Redundant Parallel File Transfer in Data Grids*

Chao-Tung Yang† Ming-Feng Yang Lung-Hsing Cheng

High-Performance Computing Laboratory
Department of Computer Science and Information Engineering
Tunghai University, Taichung, Taiwan ROC
ctyang@thu.edu.tw

Wen-Chung Chiang

Department of Information Networking Technology
Hsiuping Institute of Technology
Taichung County, Taiwan ROC
wcchiang@mail.hit.edu.tw

Abstract

In data grid, co-allocation architecture can be used to enable parallel transferring of data file from multiple replicas which stored in the different grid sites. Some schemes base on co-allocation model were proposed and used to exploit the different transfer rates among various client-server network links and to adapt dynamic rate fluctuations by dividing data into fragment. These schemes showed the more fragments used the more performance conducted when data transfer in parallel with evidence. In our previous work, we propose a scheme named Anticipative Recursively-Adjusting Mechanism (ARAM) in previous research work. The best thing is performance tuning through the alpha value, it's rely on special feature to adapt different network situations in a data grid environment. In this paper, the TCP Bandwidth Estimation Model (TCPBEM) is used to evaluate dynamic link state by detect TCP throughput and packet lost rate between grid nodes. We integrate the model into ARAM, called Anticipative Recursively-Adjusting Mechanism Plus (ARAM+), that can be more reliable and reasonable then previous one. In the meanwhile, we also design a Burst Mode which could increase transfer rate of ARAM+. This approach not only adapts worst network link but also speedup the overall performance.

1. Introduction

Data grids gather distributed resources to solve large-size dataset management problems, and enable the selection, sharing, and connection of a wide variety of geographically distributed computational and storage resources to deal with large-scale data-intensive application requests [2, 8, 9, 10, 11, 14, 17, 18, 19, 20, 32, 33]. Most data grid applications, for instance, high-energy physics, bioinformatics, and virtual astrophysical observatories, and so on, simultaneously access and execute large numbers of data files in the Grid environment.

An increasing number of scientific applications ranging from Genomics, and Proteomics, and Bioinformatics to support computational require exchange large volume of data, therefore downloading large datasets from replica locations may result in varied performance rates because replica sites may have different architectures, system loading, and network connectivity. Bandwidth quality is the most important factor affecting internet transfers between clients and servers, and download speeds are bounded by traffic congestion due to bandwidth limitations.

One method for improving download speeds uses replica selection techniques to determine the best replica locations [28]. However, by downloading datasets from the single best server often results in ordinary transfer rates, because bandwidth quality varies unpredictably due to the shared nature of the Internet. Another method uses co-allocation [27] technology to download data.

Co-allocation architectures were developed to enable clients to download data from multiple locations by establishing multiple connections in parallel, thus improving performance as compared to the single server case and alleviating the internet congestion problem [27]. Parallel downloading [22, 23, 25, 26] is a technique used to fetch and download files from multiple sources including Web servers, file servers, P2P nodes, etc. Parallel downloading has been integrated into many Internet applications and has become the core of many P2P systems. It speeds up download time and eliminates the server selection problem [21, 23, 24]. In previous works [15, 27], several co-allocation strate-

* This work is supported in part by the National Science Council, Taiwan R.O.C., under grants no. NSC 96-2221-E-029-019-MY3 and NSC 97-2622-E-029-001-CC2.
† Corresponding author.

1521-9097/08 $25.00 © 2008 IEEE
DOI 10.1109/ICPADS.2008.48

gies were addressed. However there are still drawbacks in these approaches, such as: faster servers wait for the slowest one to deliver its final block. As shown in [15, 16], this may degrade network performance by repeatedly transferring the same block. Hence, it is important to minimize the differences in finishing time among different servers, and to prevent the same block from being transferred in different links between servers and clients.

In our previous research work, by means of continuous adjusting the workload of each selected replica server, the Anticipative Recursively-Adjusting Mechanism (ARAM) scheme measures the actual bandwidth performance during the term of transferring data file, and according to the previous assigned transfer size finished rate, anticipates bandwidth status at the next transfer section to regulate the workload on the next section. The basic idea is to assign less data on the selected replica server with performance of a greater variability network link. In other words, for a link with more variable bandwidth, effective bandwidth will be smaller, and the finished rate of the previous assigned transfer size would be smaller as well. The goal is to make the expected finished time of each server to be the same.

In this paper, we first present our new approach based on ARAM co-allocation strategy in data grid environment, we have design and implement TCP bandwidth estimation model and Burst mode to enhancing original algorithm for ARAM, which mean all selected replica server will continue to adjust work load by TCP throughput and packet lost rate, in the mean time faster servers will get double or even quadruple throughput through Burst mode enable.

The remainder of this paper is organized as follows. Related background review and studies are presented in section 2. Our new approach is outlined in section 3. Experimental results and a performance evaluation of our scheme are presented in section 4. Section 5 concludes this research article.

2. Background review and related work

The architecture proposed in [29] consists of three main components: an information service, a broker/co-allocator, and local storage systems. Figure 1 shows co-allocation of Grid Data transfers, which is an extension of the basic template for resource management [7] provided by the Globus Toolkit. Applications specify the characteristics of desired data and pass attribute descriptions to a broker. The broker queries available resources and gets replica locations from the Information Service [6] and Replica Management Service [31], then gets lists of physical file locations.

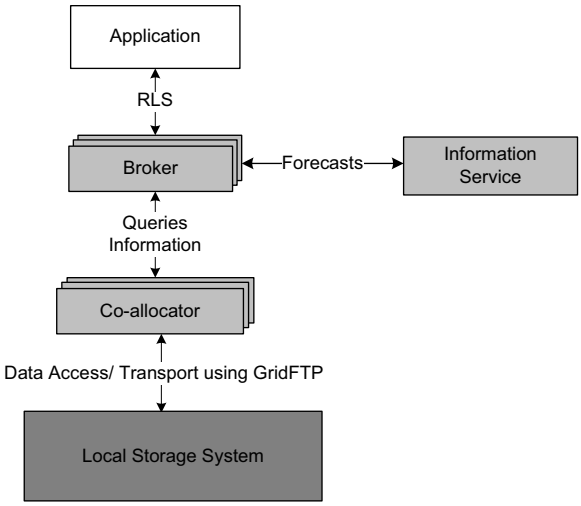

Figure 1: Data grid co-allocation architecture

In [21], the authors propose architecture for co-allocating Grid data transfers across multiple connections by exploiting the partial copy feature of GridFTP. They supply strategies such as Brute-Force, History-based, and two Dynamic Load Balancing techniques, conservative and aggressive, for allocating data blocks. Several co-allocation strategies presented in previous works are described below.

Brute-Force Co-Allocation [21]: The Brute-Force Co-allocation scheme divides file sizes equally among available flows; it does not address bandwidth differences among various client-server links.

History-based Co-Allocation [21]: The History-based Co-allocation scheme keeps block sizes per flow proportional to predict transfer rates, and disregards the influence of network variations between client and server.

Conservative Load Balancing [21]: The Conservative load balancing scheme divides requested datasets into k disjoint blocks of equal size. Available servers are allocated single blocks to deliver in parallel. Servers work in sequential order until all requested files are downloaded. Loadings on the co-allocated flows are automatically adjusted because the faster servers deliver larger file portions more quickly.

Aggressive Load Balancing [21]: This method 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.

Dynamic Co-allocation with Duplicate Assignments (DCDA) [3, 4]: The co-allocation strategies described above do not handle the shortcoming of faster servers having to wait for the slowest server to deliver its final block which, in most cases, wastes much time and de-

creases overall performance. Neither prediction nor heuristics approaches, the DCDA scheme dynamically co-allocates duplicate assignments and copes nicely with changes in server speed performance, as shown in Figure 2. The DCDA scheme is based on an algorithm that uses a circular queue. Let D be a dataset and k the number of blocks of fixed size in the dataset. D is divided into k disjoint blocks of equal size and all available servers are assigned to deliver blocks in parallel. When a requested block is received from a server, one of the unassigned blocks is assigned to that server. The co-allocator repeats this process until all blocks have been assigned. DCDA behaves well even when server links are broken or idled. The DCDA scheme is flawed: it consumes network bandwidth by repeatedly transferring the same blocks. This wastes resources and can easily cause bandwidth traffic jams in the links between servers and clients.

3. Our Approach

3.1. Anticipative Recursively-Adjusting Mechanism (ARAM)

Grid Recursively-Adjusting mechanism can reduce file transfer completion times and idle times spent waiting for the slowest server. It also provides an effective scheme for reducing the cost of reassembling data blocks. However, our scheme did not consider the potential effect of server links broken or idled during file transfers. Therefore, we propose an efficient approach called the Anticipative Recursively-Adjusting Mechanism (ARAM) to extend and improve upon Recursive-Adjustment Co-Allocation [12]. The main idea of the ARAM is to assign transfer requests to selected replica servers according to the finish rates for previous transfers, and adjusts workloads on selected replica servers according to anticipated bandwidth statuses. By continuously adjusting selected replica server workloads, the Anticipative Recursively-Adjusting Mechanism scheme measures actual bandwidth performance during data file transfers and regulates workloads by anticipating bandwidth statuses for subsequent transfers according to the finish rates for previously assigned transfers.

The basic idea is to assign less work to selected replica servers on network links with greater performance variability. Links with more bandwidth variation will have smaller effective bandwidth, and the finish rates for previous assigned transfers will be smaller as well. The goal is to have the expected finished times of all servers be the same. Our approach performs well, even when the links to selected replica servers are broken or idled. It also reduces the idle time wasted waiting for the slowest server. As appropriate file sections are selected, they are first divided into proper block sizes according to the respective server bandwidths, previously assigned file sizes, and transfer finish rates.

Initially, the finish rate is set to 1. Next, the co-allocator assigns the blocks to selected replica servers for transfer. At this moment, it is expected that the transfer finish times will be consistent with $E(t_1)$. However, since server bandwidths may fluctuate during segment deliveries, the actual completion times may differ from the expected time $E(t_1)$ (solid lines in Figures 2 and 3). When the fastest server finishes at time t_1, the size of unfinished transfer blocks (italic blocks in Figures 2 and 3) is measured to determine the finish rate. Two outcomes are possible: the quickest server finish time t_1 may be slower than or equal to the expected time, $E(t_1)$, indicating that network link performance remained unchanged or declined during the transfer. In this case, the difference in transferred size between the expected time and actual completion time (italic block in Figure 2) is then calculated.

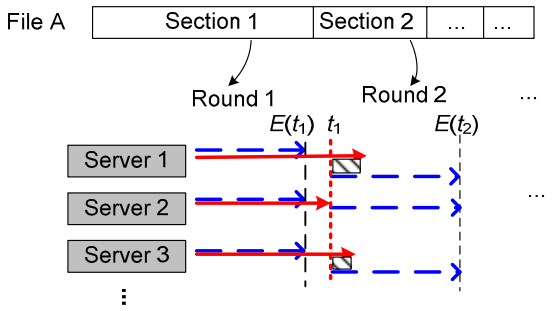

Figure 2: Later-than-expected-time adjustment process

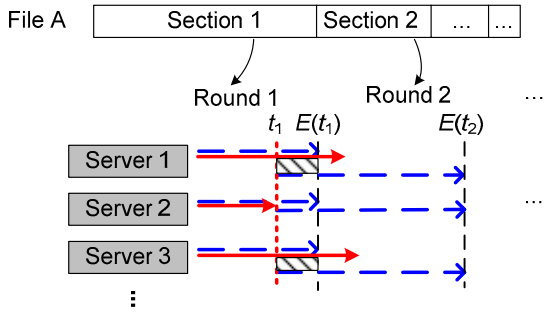

Figure 3: Earlier-than-expected-time adjustment process

The other outcome is that the quickest server finish time t_1 may be faster than the expected time, $E(t_1)$, indicating an excessively pessimistic anticipation of network performance, or an improvement in replica server network link performance during the transfer. The difference in transferred size between the earliest and the

expected time (italic block in Figure 3) is then measured. If the anticipated network performance was excessively pessimistic for the previous transfer, it is adjusted for the next section. The next task is to assign proper block sizes to the servers along with respective bandwidths and previous finish rates, enabling each server to finish its assigned workload by the expected time, $E(t_2)$. These adjustments are repeated until the entire file transfer is finished.

3.2. TCP Bandwidth Estimation Model

TCP/UDP is one of the core protocols of the Internet protocol suite. TCP provides reliable, in-order delivery of a stream of bytes, making it suitable for applications like file transfer such as for example GridFTP. Parallel TCP sockets is a generic "hack" to improve throughput attained by TCP for bulk data transfers by opening several TCP connections and striping the data file over them [1]. In practice, it is often unclear how many sockets one needs to open in order to achieve satisfactory throughput since opening too many connections may be undesirable for various reasons [1, 5, 13, 16]. The TCP Bandwidth Estimation Model [13] as a function of packet loss, round trip time, maximum segment size, along with a handful of other miscellaneous parameters across a wide range of packet losses. Now we understand how TCP Bandwidth Estimation Model to measure server bandwidth, its make more reliable and reasonable.

3.3. K-means algorithm

The k-means algorithm is an algorithm to cluster n objects based on attributes into k partitions, $k < n$. It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data. It assumes that the object attributes form a vector space. The objective it tries to achieve is to minimize total intra-cluster variance. The simulation results were shown as Figure 4.

3.4. Burst Mode

Just like many network accelerators method or multithreading, Burst Mode (BM) could make one huge bandwidth split into small pipeline to working at the same time. Burst Mode focus on the fastest group of all servers, first, BM can differentiate varieties network bandwidth of candidate servers, second, BM choose the faster one then others. Finally, BM makes one single job into many in apropos, as shown in Figure 5.

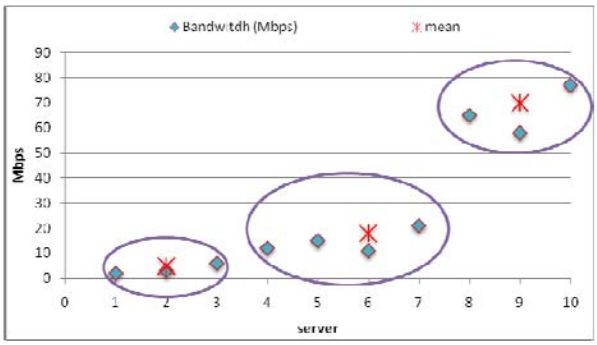

Figure 4: Grouping number of servers in various bandwidths

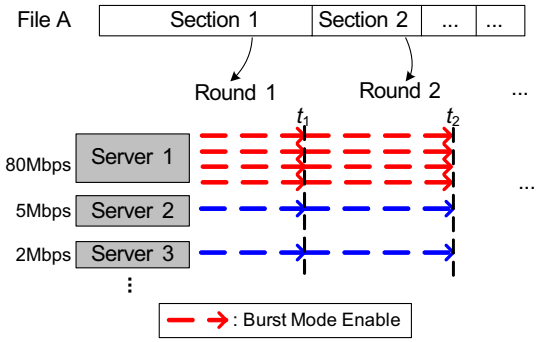

Figure 5: Higher bandwidth enable Burst Mode

In the previous experimental results showed that few replica servers in locally was high-efficiency than many of remote replica servers. According it, main idea in Burst Mode (BM) not only find out the fastest server group but also make them download in multithreading. In the mean time, BM could deal cutting block properly with various datasets.

3.5. Anticipative Recursively-Adjusting Mechanism Plus (ARAM+)

3.5.1. Assumptions: We outline the assumptions for our system design model and list as following:

- High-speed networking equipment with full-duplex transmission has been seen everywhere.
- Grouping and distinguishing amount of heterogeneous servers determine by end-to-end network bandwidth. Higher network bandwidth could also support multithreading to speedup total performance.
- According to Moore's Law, modern computer computing abilities is powerful then before in continually, such as the new hard drive, 10 Gigabit Ethernet backbone and multi-core CPU. Each I/O device will be act the key of performance.

- The time in transferring processes of stopping/assigning and calculated TCP_{BW} to the selected replica server is negligible.

3.5.2. Anticipative Recursively-Adjusting Mechanism Plus: Anticipative Recursive-Adjustment Mechanism Plus (ARAM+) base on ARAM, it's not only inheritance from ARAM but also enhance two features: 1) TCP Bandwidth Estimation Model (TCPBEM), 2) Burst Mode (BM). ARAM+ by means of continuous adjusting the workload of each selected replica server, which measures the actual bandwidth performance by TCPBEM during the term of transferring data file, and according to the previous job finished rate to adjust alpha value at next transfer section. There are some interesting idea from P2P network and distributed denial-of-service (DDoS) attack, as we know; P2P network base on share-based, its sharing data and download in parallel each other but closely demander will higher priority. Another typical example is DDoS attack; it occurs when multiple compromised systems flood the bandwidth or resources of a targeted system. We have combined those two characters in our approach. The multithreading in Burst Mode (BM) deign idea came from DDoS attack, BM "flooding" the bandwidth of target replica server to make download performance speedup. Another idea from P2P network apply to ARAM+, its pre-selection many candidates replica from different server, then elected properly servers and allocated only just work load to fit server capacity.

Compare to previous work [24, 25, 27, 29, 30], Anticipative Recursively-Adjusting Mechanism (ARAM) and Recursively-Adjusting Mechanism (RAM) both are designed base on Co-Allocation Architecture and also rely on tuning alpha value by hand to adapt specific situation in Data Grid. The ARAM+ does the same way with those strategies, but different in tuning alpha value dynamically.

3.5.3. ARAM+ Algorithm:

[Initialization]
Total candidate servers current bandwidth measuring using TCP Bandwidth Estimation Model (TCPBEM) and calculation the appropriate alpha value by equation 14 and 15.
[Allocation of blocks to the selected servers]
LOOP WHEN (*UnassignedFileSize* and total *UnfinishedFileSize* are greater than zero) THEN
{
IF (*UnassignedFileSize* and Total *UnfinishedFileSize* are greater than $TotalTCP_{BW}$) THEN
{
IF (*UnassignedFileSize* and Total *UnfinishedFileSize* multiplied by α greater then $TotalTCP_{BW}$) THEN
{
Define new section to be allocated
$SEj = (UnassignedFileSize + TotalUnfinishedFileSize) * \alpha, \quad 0 < \alpha \leq 1$
}
ELSE
{
Define finial section
$SEj = UnassignedFileSize + TotalUnfinishedFileSize$
}
}
END LOOP;

Step 1. Define new section to be allocated *SEj*
Step 2. Monitor each selected replica server
Step 3. Allocate blocks to each selected replica server, according to the TCP_{BW} of the selected replica server, and the previous finished rate *Rj*-1 of the selected replica server (Initial R_0=1)
Step 4. Monitor each download flow

LOOP WHEN (The fastest flow finishes its assigned data blocks) THEN
{
IF (The first finished time of *RTji* is earlier then expect time *ETji* and the transferred size *TSji* is greater than the expected size *Sji*) THEN
{
The *rji*=1
}
ELSE
{
Measure the finished rate of the previous assigned file size to be delivered ($0 \leq rji \leq 1$)
}

$$rji = \begin{cases} \dfrac{TSji}{Sji}, RTji \geq ETji \\ 1, RTji < ETji, and\ TSji \geq Sji \end{cases}$$

}
END LOOP;

4. Experimental Results

4.1. Experimental results and analyses

The experiment and case design to testify our propose approach, Burst Mode (BM) to speedup performance in locally/remotely and dynamic truing alpha value to adapt uncertain network situation especially. We have design some test case as follows and detail as shown in Figure 6.

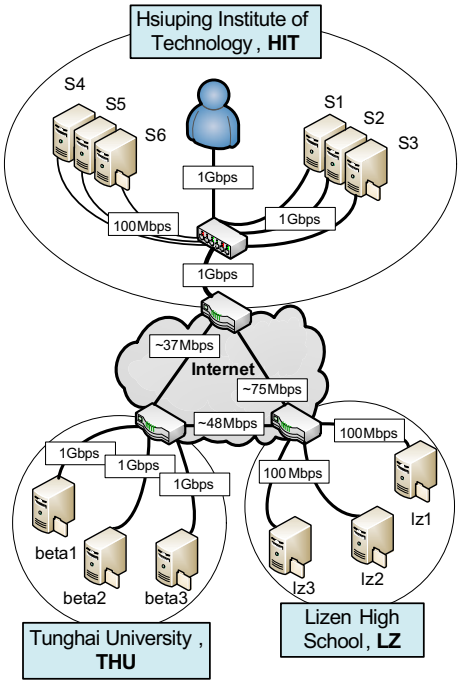

Figure 6: Scenarios for our test-bed of TIGER

4.2. Case study - Replica selects and transfers in "Across Grid" vs. "Local Grid"

To verify the efficiency of Burst Mode been enabling or not, we design two scenarios to compare, all test case were list in Table 1 and Table 2.

Table 1: Scenario for replica in locally or not

Scenario	Replica server list
ARAMplus_4: non-local	THU-S1, S2; LZ1, 2
ARAMplus_4: local-1	HIT-S1, S2; THU-beta1, beta2
ARAMplus_4: local-2	HIT-S1, S2; THU- beta 1; LZ-1
ARAMplus_4: local-3	HIT-S1, S2; LZ-1, 2
ARAMplus_4: all-local	HIT-S1, S2, S3, S4

Table 2: Scenario for different replica number and selection

Scenario	Replica server list
Rx6_non-local	LZ-1, 2, 3; THU- beta 1, beta 2, beta 3
Rx6_local	HIT-S1, S2, S3, S4, S5, S6
Rx2_local	HIT-S1, S2
Rx2_non-local-THU	THU-S1, S2
Rx2_non-local-LZ	LZ-S1, S2

In general case, more number and placement locally of replicas will gain better performance for parallel transform file. From our results shown as Figure 7 and Figure 8, we found more number of replicas remotely from user performance improvement wasn't obvious even worst then few replica in locally. Meanwhile parallel download in locally, Burst Mode enable meet two copied of replica could better than six copied on unbalanced bandwidth.

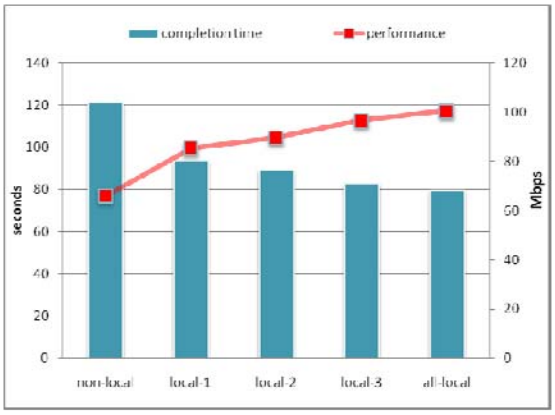

Figure 7: The performance result affected by different replica location

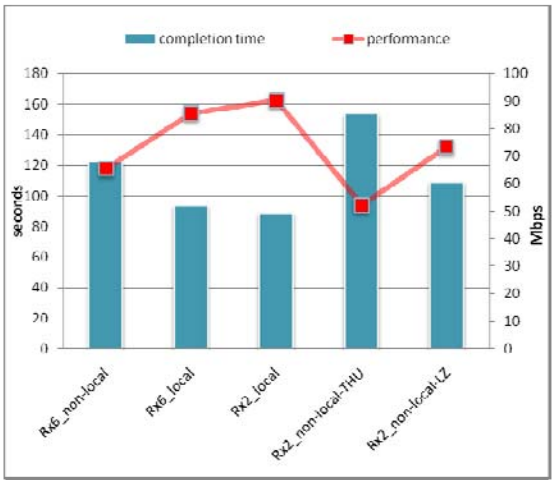

Figure 8: The performance result affected by different number of replica and selection

4.3. Case study - RAM and ARAM vs. ARAM+

Compare to RAM [32] and ARAM [26] both were using constant alpha value, our approach the ARAM+ could rely on dynamic alpha value to adapt fluctuant network link in Data Grid. The case study (list in Table 4) of RAM and ARAM, we set constant alpha value to be 0.9, 0.5 and 0.1 compare to ARAM+, also replica placement and selection from inside and outside reign. In order to distinguish replica location between inside and outside reign, these two kinds of replica selection plan list as follow table.

Table 3: Replica placement and selection plan

mix	HIT-S1, S2; LZ-1, 2; THU-beta1, beta2
local	HIT-S1, S2, S3, S4, S5, S6

Table 4: Scenario for alpha value tuning

Scenario A	Scenario B
RAM(0.1)_local	RAM(0.1)_mix
ARAM(0.1)_local	ARAM(0.1)_mix
RAM(0.5)_local	RAM(0.5)_mix
ARAM(0.5)_local	ARAM(0.5)_mix
RAM(0.9)_local	RAM(0.9)_mix
ARAM(0.9)_local	ARAM(0.9)_mix
ARAM+_local	ARAM+_mix

In order to accentuation the advantages of Burst Mode method and dynamic alpha value adjustment, there are two scenario set A and B in our next experiment. The total performances in Scenario B have obviously been improving then Scenario A. In Scenario A, the total amount of TCP bandwidth in these case were not quire difference, but in Scenario B were significant difference. All those case study especially in Scenario B, the Burst Mode were affect total performance hugely improvement, as shown in Figure 9 and 10.

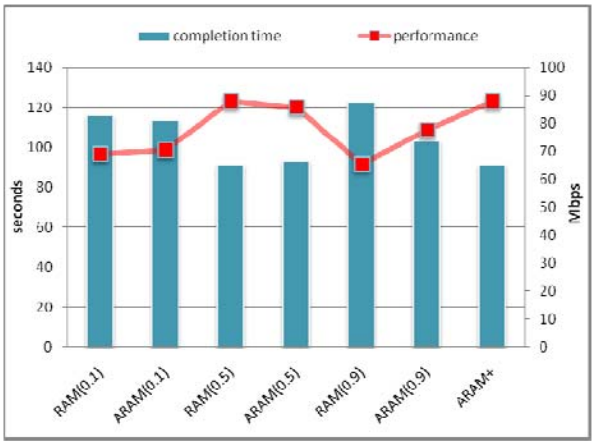

Figure 9: The performance result with scenario A

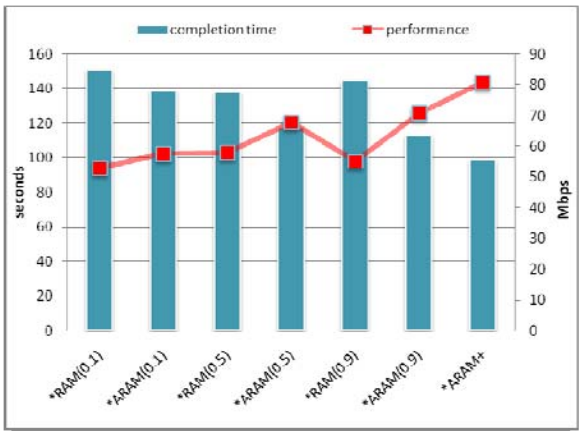

Figure 10: The performance result with scenario B

5. Conclusions

The Co-Allocation Architecture can be used to enable parallel transferring of data file from multiple replicas in Data Grid, which mean all replicas stored in the different grid sites. There are many schemes base on Co-Allocation Model were proposed and used to exploit the different transfer rates among various client-server network links and to adapt dynamic rate fluctuations by dividing data into fragment. These schemes showed the more fragments used the more performance conducted when data transfer in parallel with evidence. In fact, some of schemes can be applied for specific situations; however, most situations are not common actually. For this issue, we have proposed a scheme named Anticipative Recursively-Adjusting Mechanism Plus (ARAM+) based on ARAM. The best part is performance tuning through the dynamic alpha value adjust in continually, it's rely on special feature to adapt different network situations in a data grid environment. In our experimental results, the TCP Bandwidth Estimation Model is used to evaluate dynamic link state by detect TCP throughput and packet lost rate between grid nodes, we also found that can be more reliable and reasonable then ARAM and others. In the meanwhile, the Burst Mode which can be increase transfer rate with evidence. The ARAM+ not only adapts worst network link but also speedup the overall performance especially in wide area grid network.

References

[1] E. Altman, D. Barman, B. Tuffin and M. Vojnovic´ "Parallel TCP Sockets: Simple Model, Throughput and Validation", INFOCOM 2006, ISBN: 1-4244-0221-2, April 2006.

[2] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D.Quesnel, and S. Tuecke, "Data management and transfer in high-performance computational grid environments," Parallel Computing, vol. 28, no. 5, pp. 749-771, May 2002.

[3] R.S. Bhuvaneswaran, Y. Katayama, and N. Takahashi, "Dynamic Co-allocation Scheme for Parallel Data Transfer in Grid Environment," Proceedings of First International Conference on Semantics, Knowledge, and Grid (SKG 2005), pp. 17, 2005.

[4] R.S. Bhuvaneswaran, Y. Katayama, and N. Takahashi, "A Framework for an Integrated Co-allocator for Data Grid in Multi-Sender Environment," IEICE Transactions on Communications, vol. E90-B, no. 4, pp. 742-749, 2007.

[5] J. Bolliger, T. Gross, and U. Hengartner, "Bandwidth modelling for network-aware applications," In INFOCOM '99, ISBN: 0-7803-5417-6, March 1999.

[6] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, and M. Ripeanu, "Giggle: A Framework for Constructing Scalable Replica Location Services,"

Proceedings of the 2002 ACM/IEEE conference on Supercomputing, pp. 1-17, November 2002.

[7] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets," Journal of Network and Computer Applications, 23(3), pp. 187-200, 2001.

[8] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10'01), pp. 181-194, August 2001.

[9] K. Czajkowski, I. Foster, and C. Kesselman. "Resource Co-Allocation in Computational Grids," Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC-8'99), August 1999.

[10] I. Foster, C. Kesselman, S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." International Journal of High Performance Computing Applications, Vol. 15, No. 3, pp. 200-222, 2001.

[11] I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," International Journal of High Performance Computing Applications, Vol. 11, No. 2, pp. 115-128, 1997.

[12] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data Management in an International Data Grid Project," Proceedings of the First IEEE/ACM International Workshop on Grid Computing - Grid 2000, Bangalore, India, December 2000.

[13] T.J. Hacker and B.D. Athey, "The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network," Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, 10.1109/IPDPS.2002.1015527.

[14] Open Grid Forum, http://www.ogf.org/

[15] M. Mathis, J. Semke, J. Mahdavi and T. Ott. "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm." Computer Communication Review, volume 27, number3, July 1997.

[16] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. "Modeling TCP throughput: A simple model and its empirical validation," ACMSIGCOMM, September 1998.

[17] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney, "File and Object Replication in Data Grids," Journal of Cluster Computing, 5(3), pp. 305-314, 2002.

[18] The Globus Alliance, http://www.globus.org/

[19] S. Vazhkudai and J. Schopf, "Using Regression Techniques to Predict Large Data Transfers," International Journal of High Performance Computing Applications (IJHPCA), vol. 17, no. 3, pp. 249-268, August 2003.

[20] S. Vazhkudai and J. Schopf, "Predicting Sporadic Grid Data Transfers," Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11 '02), pp. 188-196, July 2002.

[21] S. Vazhkudai, "Enabling the Co-Allocation of Grid Data Transfers," Proceedings of Fourth International Workshop on Grid Computing, pp. 44-51, 17 November 2003.

[22] S. Vazhkudai, S. Tuecke, and I. Foster, "Replica Selection in the Globus Data Grid," Proceedings of the 1st International Symposium on Cluster Computing and the Grid (CCGRID 2001), pp. 106-113, May 2001.

[23] S. Vazhkudai, J. Schopf, and I. Foster, "Predicting the Performance of Wide Area Data Transfers," Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), pp. 34-43, April 2002.

[24] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing," ACM Computing Surveys, vol. 38, issue 1, Article 3, March 2006.

[25] C.M. Wang, C.C. Hsu, H.M. Chen, and J.J. Wu, "Efficient Multi-Source Data Transfer in Data Grids," Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06), pp. 421-424, 16-19 May 2006.

[26] C.T. Yang, Y.C. Chi, T.F. Han and C.H. Hsu, "Redundant Parallel File Transfer with Anticipative Recursively-Adjusting Scheme in Data Grids", ICA3PP 2007, pp. 242–253, 2007.

[27] C.T. Yang, I.H. Yang, and C.H. Chen, "Improve Dynamic Adjustment Mechanism in Co-Allocation Data Grid Environments," Proceedings of the 11th Workshop on Compiler Techniques for High-Performance Computing (CTHPC-11' 05), pp. 189-194, 17-18 March 2005.

[28] C.T. Yang, I.H. Yang, K.C. Li, and S.Y. Wang, "Improvements on Dynamic Adjustment Mechanism in Co-Allocation Data Grid Environments," The Journal of Supercomputing, Springer Netherlands, vol. 40, no. 3, pp. 269-280, June 2007.

[29] C.T. Yang, C.H. Chen, K.C. Li, and C.H. Hsu, "Performance Analysis of Applying Replica Selection Technology for Data Grid Environments," PaCT 2005, Lecture Notes in Computer Science, vol. 3603, pp. 278-287, Springer-Verlag, September 2005.

[30] C.T. Yang, I.H. Yang, K.C. Li, and C.H. Hsu "A Recursively-Adjusting Co-Allocation Scheme with Cyber-Transformer in Data Grids," Future Generation Computer Systems, Elsevier B.V., 2008.

[31] C.T. Yang, I.H Yang, C.H Chen and S.Y Wang, "Implementation of a Dynamic Adjustment Mechanism with Efficient Replica Selection in Co-Allocation Data Grid Environments," 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.

[32] L. Yang, J. Schopf, and I. Foster, "Improving Parallel Data Transfer Times Using Predicted Variances in Shared Networks," Proceedings of the fifth IEEE International Symposium on Cluster Computing and the Grid, (CCGrid '05), pp. 734-742, 9-12 May 2005.

[33] X. Zhang, J. Freschl, and J. Schopf, "A Performance Study of Monitoring and Information Services for Distributed Systems", Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12 '03), pp. 270-282, August 2003.