私立東海大學

資訊工程研究所

碩士論文

指導教授：楊朝棟　博士

基於應用網頁平台實現檔案平行傳輸效能之比較於網格環境與雲端環境

A Web-based Parallel File Transfer on Grid and Cloud Environments

研 究 生：羅裕翔

中華民國九十九年七月

摘要

從以往至今個人電腦或其它電腦裝置當個人需要使用特定應用程式之時，就必需在該個人電腦或其它電腦裝置上安裝該特定之應用程式，而當該電腦故障或使用者需要至其它個人電腦上需要使用該軟體之時則必需重新安裝，如此一來浪費使用者的無形時間或金錢等資源。近幾年以來由於 Web 的逢勃發展，許多應用軟體程式及後端資料儲存都逐轉換至網際網路來使用在早期一點 Web 語言 CGI 及近期有 JSP、ASP.net 及 PHP 語言，於是乎解決了以上使用者重覆安裝應用軟體程式與後端資料來源的問題。由於近年來逐漸的許多應用程式都慢慢轉換到網際網路上來使用，於是有了最近新興技術概念，如網格（Grid）與雲端（Cloud）計算服務是近這幾年來最熱門的話題。尤其是雲端計算服務更是近幾年被許多國家所被重視的未來產業之一，雲端計算服務一方面可以解決以上的問題，而另一方面也可以資源共享機制，本研究主題是應用 PHP 程式語言並且利用它開發小型 PACS 系統--MIFAS 及網格服務與雲端服務應用於後端資料來源平行傳輸的效能比較，本研究主要透過在以往網格研究應用於資料平行傳輸部份，並搭配以往所發展出的九種演算法以達加速傳輸效率做為我們的實驗組，另一方面我們也建置了小型雲端環境-HDFS (Hadoop Distributed File System)透過 HDFS over FTP 副本機制做為實驗對照組，我們將應用此兩者副本機制應用環境來做為我們檔案平行傳輸效能分析。

關鍵字：網格、雲端、平行傳輸

Abstract

Till to now personal computer or other computing device's application is standalone development, if one user wants to use some specific application in other computer or device then the computer user must be re-deploy or re-install the application in that computer. As result to improve this problem many application developer start re-design or translate application as Web application, early phases technology we common to see like as Common Gateway Interface (CGI), the near future we common to see like as Java Server Page (JSP), Active Server Page.Net (ASP.NET) and PHP Hypertext Processor (PHP). At this research we used PHP Hypertext Processor (PHP) Technology to develop a Web-base application call as "MIFAS", this application not only a min Picture Archive Communication System (PACS) but also we support Hadoop Distribute File System (HDFS) the cloud base storage as MIFAS backend storage. In recent years, the cloud computing service is one of great importance to the future of industry by many countries. Cloud computing services not only can solve the above problems, but also support resource sharing mechanism. This research theme is the grid service and cloud services back-end data sources used in the performance comparison of parallel transmission. This research study through the grid used in the past, parallel data transmission part, and with the nine previously developed algorithms to achieve speed transmission efficiency as our experimental group. On the other hand we also build a small cloud environment, Through HDFS over FTP as replica mechanism as experimental mapping group. We will use this mechanism for the two replica of application environment for us to do performance analysis of parallel transmission file.

Keywords: Grid, Cloud, Parallel Transferring

Acknowledgement

It’s time to thanks to all of those people who had helped me through the completion of this thesis. In particular, I would like to thank my adviser professor, Dr. Chao-Tung Yang, who introduced me to this topic and gave me a broad support and guidance. Professor Yang gave me a deep influence and inspiration. At there, I also like to thank my entire oral exam professor, Professor Wuu-Yang, Professor Chao-Chin Wu, Professor Cheng-Chung Chu and Professor Wen-Chung Shih, for their valuable comments.

The growth and the success are not merely from one’s efforts. There are many people whom I would like to thank; the first I would to thank is my father and mother, they always support me and concern on period of my graduate school in Tung-Hai University; the second I would thank my good friend, Mr. Wen-Jen Hu, he also gave me many experimental technology support on my thesis; finally I would like to thank my classmate, Mr. Lung-Teng Chen, Mr. Wille Chou and other classmate help me construct experimental environment. My research would not have been completed without the help from them.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Introduction

Data Grid and Cloud enable the sharing, selection, and connection of a wide variety of geographically distributed computational and storage resources for content needed by large-scale data-intensive applications such as high-energy physics, bioinformatics, and virtual astrophysical observatories. In data grids, data replication and consistency refer to the same data being stored in distributed sites, and kept consistent when one or more copies are modified. As internet is usually the underlying network of a grid, network bandwidth plays as the main factor affecting file transfers between clients and servers. In this paradigm, there are still some challenges need to be solved, such as to reduce differences in finish times between selected replica servers; to avoid traffic congestion resulting from transferring the same blocks in different links among servers and clients; and to manage network performance variations among parallel transfers. The co-allocation architecture will be developed to enable parallel downloads of datasets from selected replica servers. Our algorithm is based on using the finish rates for previously assigned transfers to anticipate the bandwidth status for the next section to adjust workloads, and to reduce file transfer times in grid and cloud environments.

In this research we support a tool with user friendly Web Transformer Interface, will be implemented and used to manage file replications and download files on data grid and cloud (HDFS) environments in parallel. This Web tools will integrate more than eight kinds of algorithm methods including the previous research work and our proposed algorithms. We will conduct the experimental results to show that Web Interface Transformer can obtain high-performance file transfer speed and reduce the time cost of parallel downloading on grid and cloud computing platforms.

This Web Interface Transformer also support hospital PACS system and DICOM Viewer, the hospital doctors can operate the tool to view patients information, common to say "Hospital Information System" (HIS) also it can view patients CT,MRI, etc image information, common to say "PACS". This tool also can manage hospital image replica server destination and use which algorithm is best for file replica transform.

1.2 Thesis Contribution

Till to now hospital build a PACS need spend expensive fee, not only that the hospital IT engineer need to pay more time to maintenance back-end storage. In this research we not only development a PACS system but also build a cloud environment as our back-end replica files storage.

1.3 Thesis Organization

The remainder of this research is organized as follows. The Chapter 1 is Related work and background review, COA algorithm and studies are presented in Chapter 2, Using tools and technology in Chapter 3, MIFAS system demo are presented in Chapter 4, The Experimental results and a performance evaluation of our scheme are presented in Chapter 5 and final conclusion and future work are presented in Chapter 6.

Chapter 2 Background Review and Related Work

2.1 Replica Management

Replica management is an important issue for a number of scientific applications. Consider a data set that contains one pet byte (one thousand million megabytes) of experimental results for a particle physics application. While the complete data set may exist in one or possibly several physical locations, it is likely that few universities, research laboratories or individual researchers will have sufficient storage to hold a complete copy. Instead, they will store copies of the most relevant portions of the data set on local storage for faster access. Replica Management is the process of keeping track of where portions of the data set can be found. Replica management is the process of creating or deleting replicas at a storage site. Most often, these replicas are exact copies of the original files, created only to harness certain performance benefits. A replica manager typically maintains a replica catalog containing replica site addresses and the file instances.

2.2 Replica Selection

Replica selection is the process of choosing a replica from among those spread across the Grid, based on some characteristics specified by the application. One common selection criteria would be access speed Replica selection is used to choose replicas from among the sites comprising a Data Grid. The selection criteria depend on application characteristics. This mechanism enables Data Grid users to easily manage replicas of data sets at their sites for better performance. Much effort has been devoted to solving the replica selection problem. The replica selection process commonly consists of three steps: data preparation, preprocessing, and prediction. When a user requests access to a data set, the system determines an appropriate way to deliver the replica to the user. One issue concerning replica selection, predicting transfer times, involves inspecting many characteristics and is a complex piece of work.

2.3 Globus Toolkit and GridFTP

In the situations where replicas are selected according to access times, Grid information services can provide information about network performance and perhaps the ability to reserve network bandwidth, while the metadata repository can provide information about file sizes. This enables selectors to rank all of the existing replicas to determine which one will yield the fastest data access time. Alternatively, selectors can consult the same information sources to determine whether there is a storage system that would yield better performance if a replica was created on it. In deciding where to create new copies, we consider these system factors: network bandwidth, CPU loading, and I/O state, to calculate the number of replicas. In this work, we used the grid middleware Globus Toolkit as our Data Grid infrastructure. The Globus Toolkit provides solutions for such considerations as security, resource management, data management, and information services. One of its primary components is MDS (Monitor and Discovery System), which is the information services component of Globus Toolkit and provides information about the available resources on the Grid and their status.

2.4 Java CoG Kit

The Java CoG (Commodity of Grid) Kit provides access to Grid services through Java via higher-level framework. Components providing client and limited server side capabilities are included. The Java CoG Kit provides a framework for utilizing the many Globus services as part of the Globus toolkit. Many of the classes are provided as pure Java implementations. Thus, writing client-side applets without installing the Globus toolkit is possible. The Java CoG Kit combines Java technology with Grid Computing to develop advanced Grid Services and accessibility to basic Globus resources. Java CoG Kits allow Grid users, Grid application developers, and Grid administrators to use, program, and administer Grids from a higher-level framework. The Java CoG Kits are good examples. These kits allow for easy and rapid Grid application development. They encourage collaborative code reuse and avoid the duplication of effort among problem solving environments, science portals, Grid middleware, and collaborator pilots. The Java CoG Kit, for example provided the implementation of Java-based GSI, gridFTP, myProxy, GRAM client implementations. Today the Java CoG Kit provides additional functionality beyond that of the Globus toolkit. Among this functionality you will find the concept of Grid abstractions and providers, workflows, and support for portal developers.

2.5 Co-Allocation Scheme

The architecture proposed in consists of three main components: an information service, a broker/co-allocator, and local storage systems. Figure 2-1 shows co-allocation of data grid transfers, an extension of the basic template for resource management provided by the Globus Toolkit. Applications specify the characteristics of desired data and pass attribute descriptions to a broker. The broker queries available a resource, gets replica locations from the Information Service and Replica Management Service, then gets lists of physical file locations.

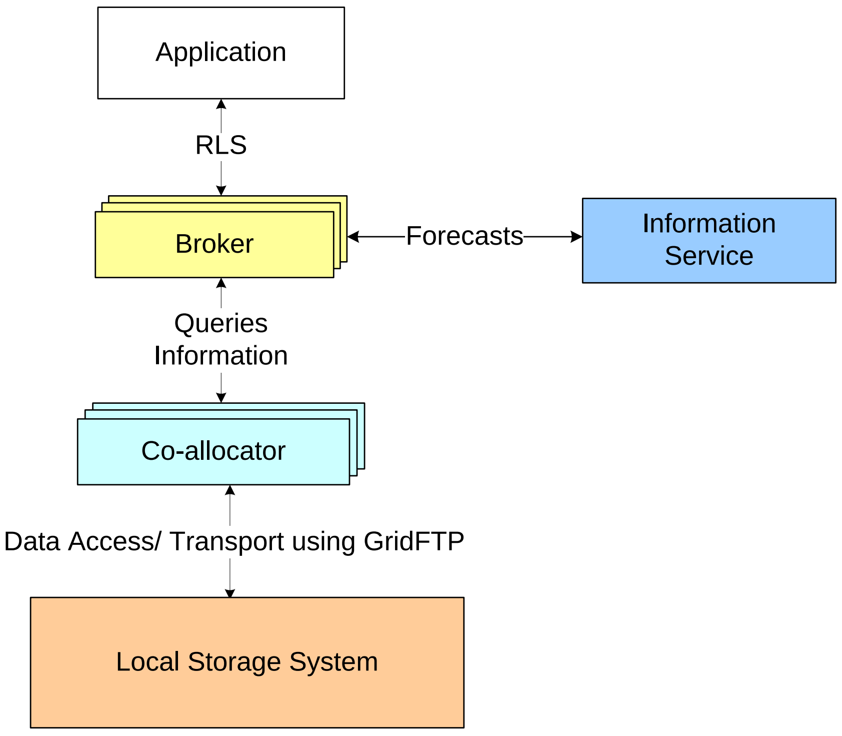


Figure 2-1: Co-allocation Architecture

Chapter 3 Algorithm in COA

3.1 Algorithm on Grid COA Review

A good algorithm will direct affect co-allocation transform and download files speed and computer node performance, in this research we will used 9 kinds of algorithm to compare file download speed and performance. Before introduce those co-allocation algorithms we will spared some algorithm as some group, like as static and dynamic co-allocation algorithm (seeing Figure 3-1). In this research we totally have 9 kinds of co-allocation algorithm such as “Brute-Force”, “History-base”, “Conservative”, “Aggressive”, “DCDA”, “DAS”, “RAM”, “ARAM” and “ARAM+”. The “Brute-Force” and “History-base” are spared as static co-allocation algorithm the others are spared as dynamic co-allocation algorithm. The difference of static and dynamic co-allocation algorithm just spared as resource whether dynamic dispatch or not, at below we will introduce them detail.

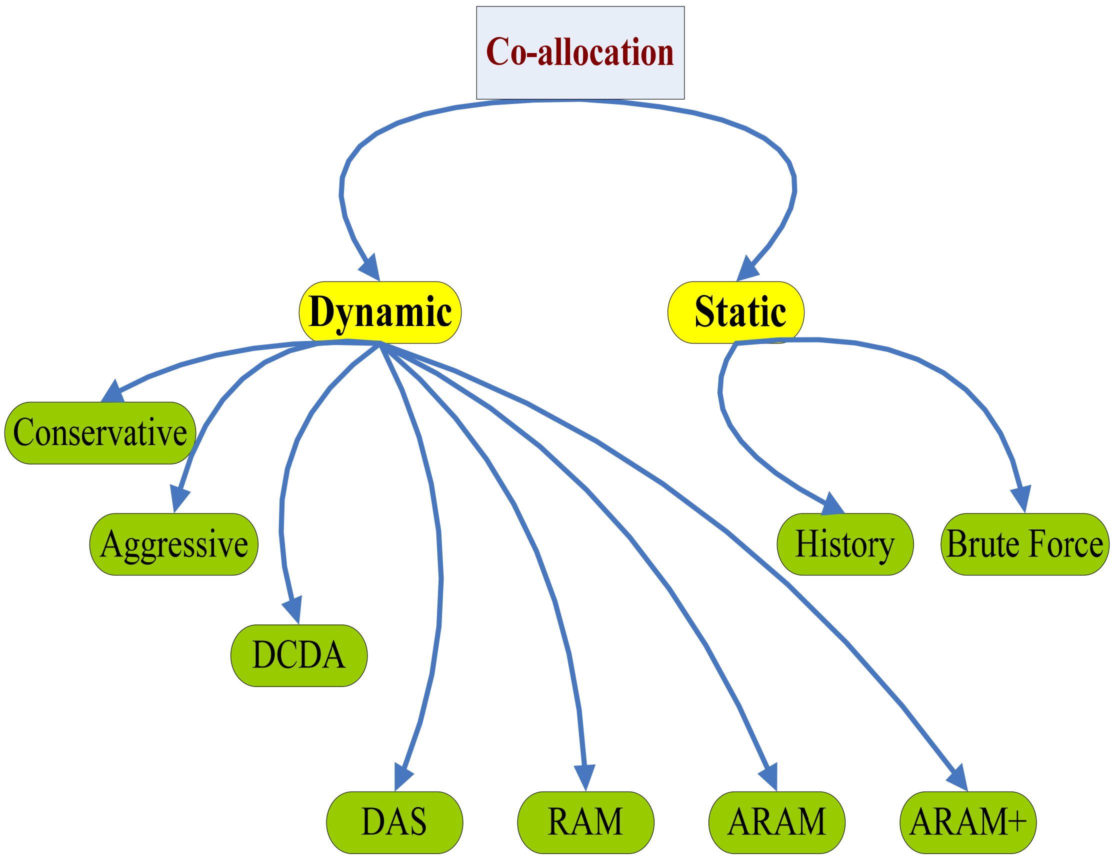


Figure 3-1: Static COA and Dynamic COA

3.2Brute Force Co-Allocation

Brute-Force Co-Allocation: The Brute-Force Co-Allocation scheme works by dividing files equally among “n” available flows (locations). Thus, if the data to be fetched is size, “s”, then this technique assigns to each flow a data block of size, “s/n”. For instance, if there are three source, the target file will be divided into three blocks equally (seeing Figure 3-2). And each source provides one block for the client. With this technique, although all the available servers are utilized, bandwidth differences among the various client-server links and not exploited.

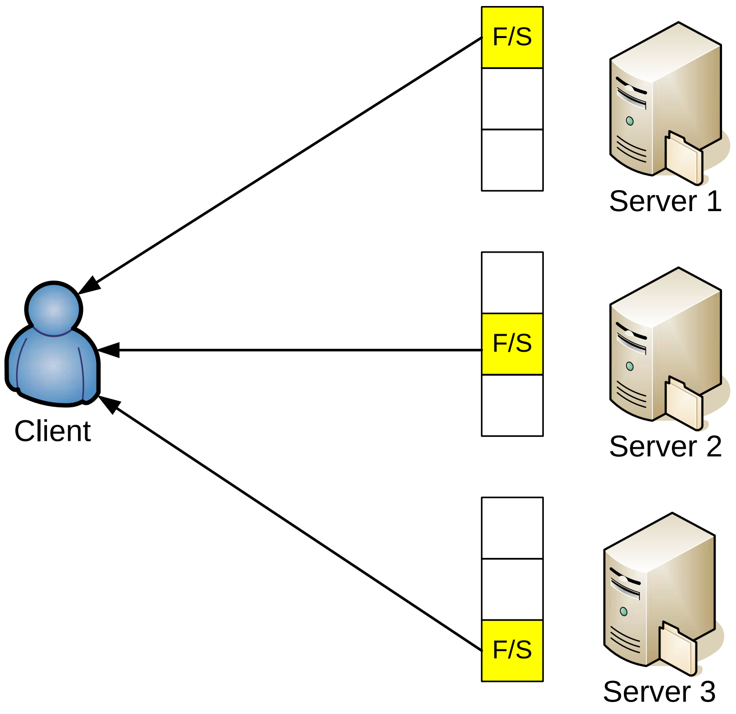


Figure 3-2: Co-Allocation of Brute Force

3.3 History-base Co-Allocation

The History-based Co-Allocation (Figure 3-3) scheme keeps block sizes per flow proportional to transfer rates predicted by the previous results of file transfer results. In history-based allocation scheme, the block size per flow is commensurate to its predicted transfer rate, decided based on a previous history of GridFTP transfers. Thus, the file-range distribution is based on the predicted merit of the flow. If these predictions are not accurate enough, renegotiations of flow sizes might be necessary as slower links can get assigned larger portions of data, which could be weight heavily on the eventual bandwidth achieved.

With the history-based approach, client divides the file into "*n*" disjoint blocks, corresponding to "*n*" servers. Each server "*i*", $1 \le i \le n$, has a predicted transfer rate of "B_i" to the client. In theory then, the aggregate bandwidth "*A*" achievable by the client for the entire download is $A = \sum_{i=1}^{i=n} B_i$. For each server "*i*", $1 \le i \le n$, and for the data to be fetched is size of, "*S*", the block size per flow is $S_i = \frac{B_i}{A} \times S$.

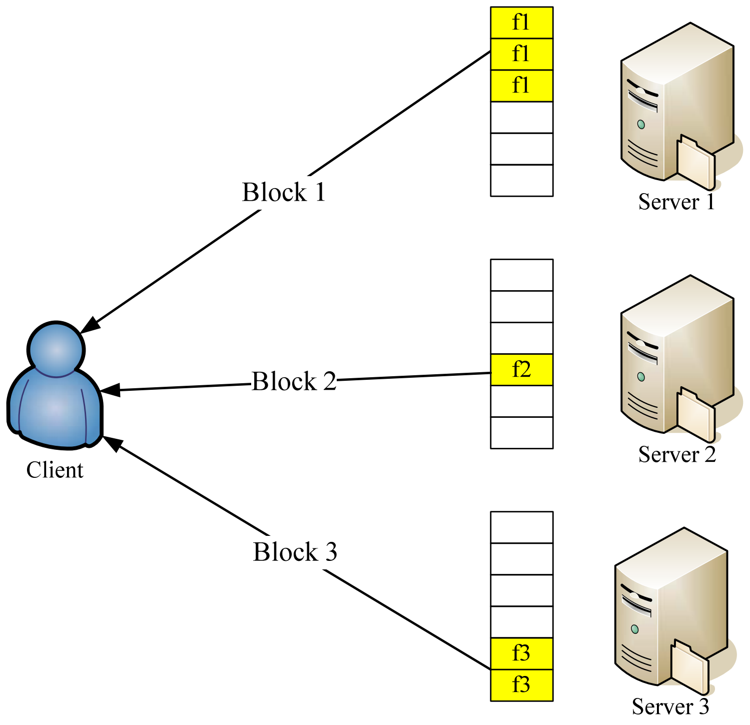


Figure 3-3: Co-Allocation of History Base Scheme

3.4 Conservative Load Balancing Co-Allocation

Conservative Load Balancing (seeing Figure 3-4): One of the proposed dynamic co-allocation algorithms is Conservative Load Balancing. The Conservative Load Balancing dynamic co-allocation strategy divides requested datasets into "*k*" disjoint blocks of equal size.

Available servers are assigned single blocks to deliver in parallel. When a server finishes delivering a block, another is requested, and so on, until the entire file is downloaded. The loadings on the co-allocated flows are automatically adjusted because the faster servers will deliver more quickly providing larger portions of the file.

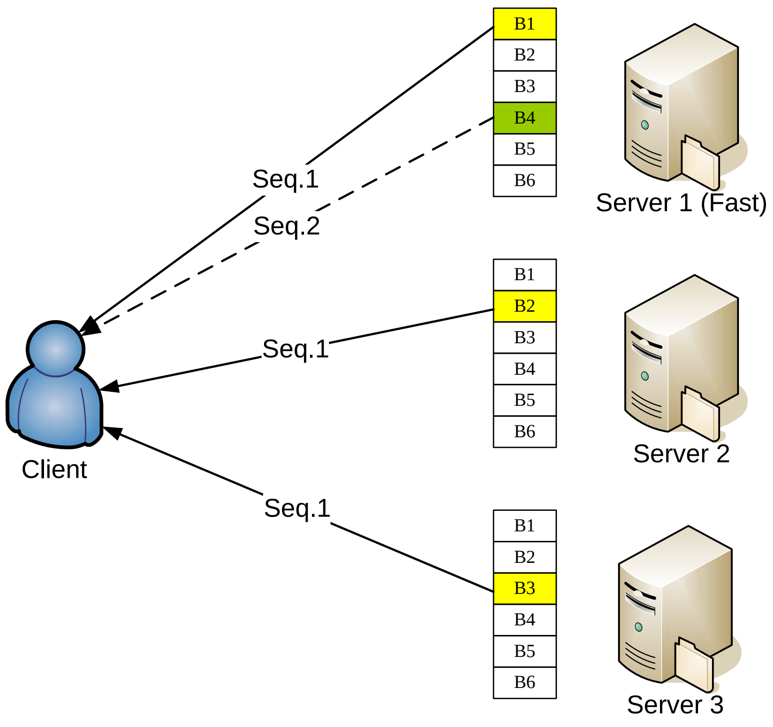


Figure 3-4: Co-Allocation of Conservative Scheme

3.5 Aggressive Load Balancing Co-Allocation

This scheme adds functions that change block size in deliveries with two method, the first gradually increasing the amounts of data requested from faster servers, second reducing the amounts of data requested from slower servers or ceasing to requesting data from them altogether (seeing Figure 3-5).

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.

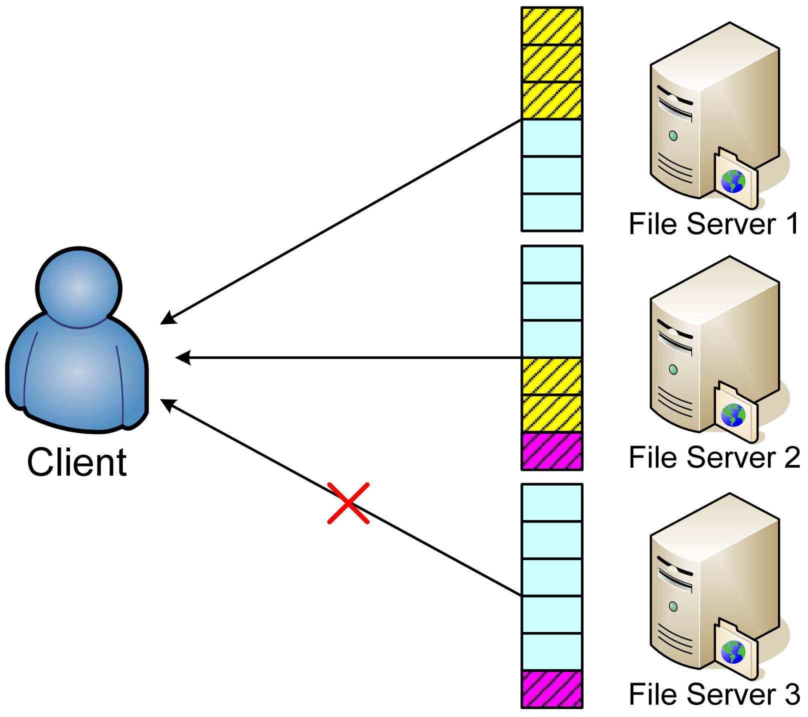


Figure 3-5: Co-Allocation of Aggressive Scheme

3.6 D.C.D.A Co-Allocation Algorithm Scheme

The DCDA is abbreviated from Dynamic Co-Allocate Duplicate Assignment, this co-allocation scheme cope with change in server speed performance nicely; The DCDA scheme is based on an algorithm that uses a circular queue. For instance (seeing Figure 3-6) 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.

The DCDA behaves well even when server links are broken or idled, but is also has some flawed, however, in that it consumers 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.

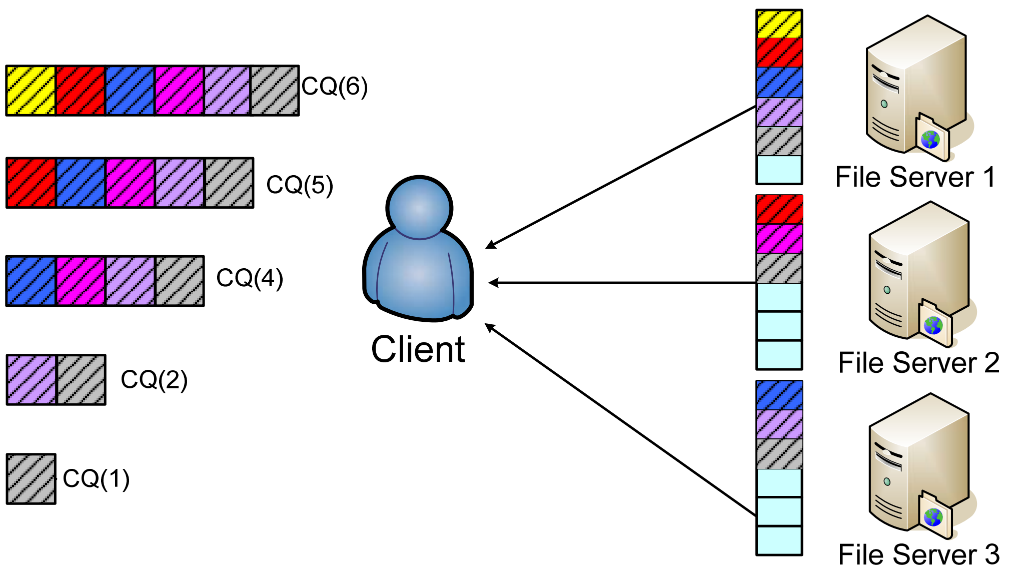


Figure 3-6: Co-Allocation of DCDA Scheme

3.7 R.A.M Co-Allocation Algorithm Scheme

The RAM is abbreviated from Recursively-Adjusting Mechanism, this co-allocation scheme works by continuous adjusting each replica server's workload to correspond to its real-time bandwidth during file transfers. This scheme's goal is to make the expected finish times of all servers the same. Seeing Figure 3-7 when an appropriate file section is first selected, it is divided into proper block sizes according to the respective server bandwidths. The co-allocation then assign 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 Figure 3-7). 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 work-load 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. 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.

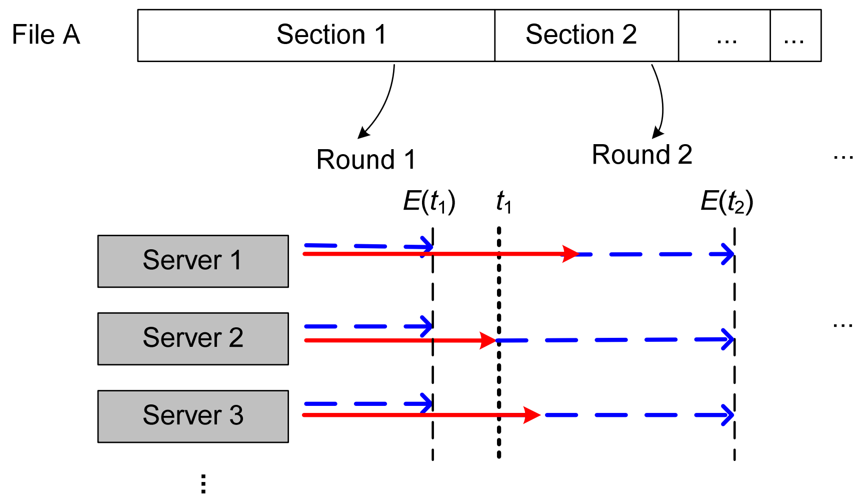


Figure 3-7: Co-Allocation of RAM Scheme

3.8 D.A.S Co-Allocation Algorithm Scheme

The DAS is abbreviated from Dynamic Adjustment Strategy This Co-allocation technology enables the clients to download data from multiple locations by establishing multiple connections in parallel.

This Co-allocation proposed a replica selection cost model and a replica selection service to perform replica selection. This Co-allocation proposes a new data transfer strategy based on this model. It consists of three phases: (1) initial phase, (2) steady phase (3) completion phase. To determine the initial block size, an upper bound that is dependent on the relation between the client's maximum bandwidth and the number of replica sources. Though multiple replicas can be downloaded in parallel, the gathered portions of files from different links must be transferred to the client in a single link. It is clear that the client's bandwidth could be bottleneck in co-allocation architecture.

3.9 ARAM Co-Allocation Algorithm Scheme

The ARAM is abbreviated from Anticipative Recursively-Adjusting Mechanism. This scheme 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, this 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 previous 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. This scheme 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. (Seeing Figure 3-8) to explain this scheme work: 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 Figure 3-8 and Figure 3-9). When the fastest server finishes at time t_1, the size of unfinished transfer blocks (italic blocks in Figure 3-8 and Figure 3-9) is measured to determine the finish rate.

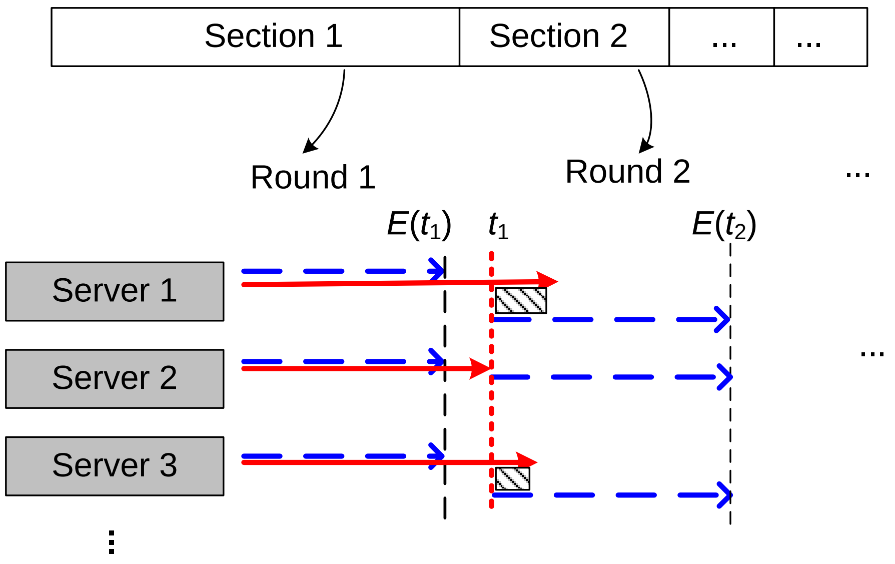


Figure 3-8: Co-Allocation of ARAM Scheme in optimistic status

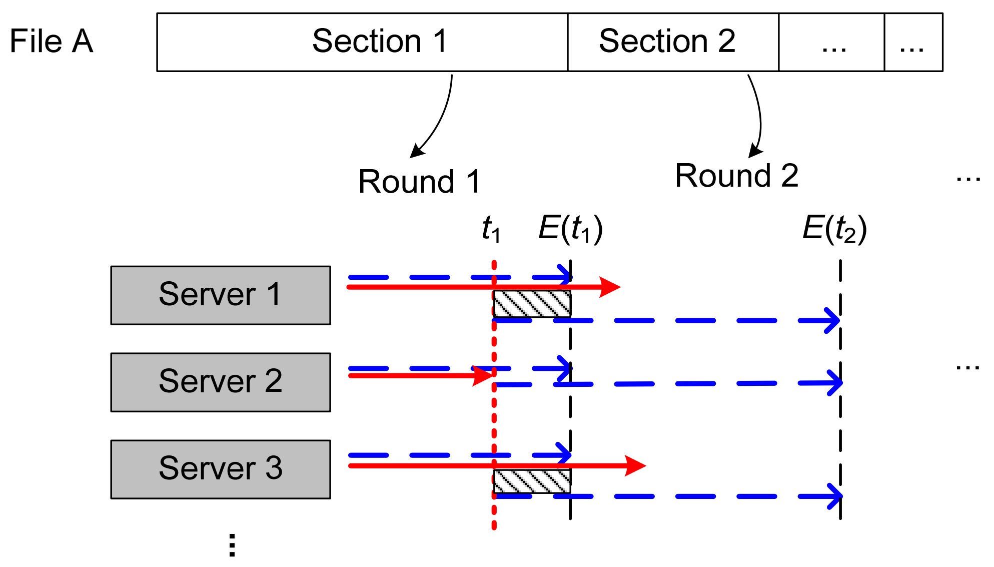


Figure 3-9: Co-Allocation of ARAM in pessimistic

3.10 ARAM + Co-Allocation Algorithm Scheme

The ARAM+ is abbreviated from Anticipative Recursively-Adjusting Mechanism Plus. This scheme is not merely inherited from ARAM; it has also been enhanced in two areas: one is TCP Bandwidth Estimation Model (TCPBEM), and another is Burst Mode (BM). ARAM+ continually adjusts the workloads on selected replica servers by measuring actual bandwidth performance via TCPBEM during data file transfers (Seeing Figure 3-10) 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. This scheme has combined these elements approach. The multithreading in the Burst Mode (BM) design came from DDoS attacks seeing Figure 3-10 to explain using 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, the Anticipative Recursively-Adjusting Mechanism (ARAM) 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.

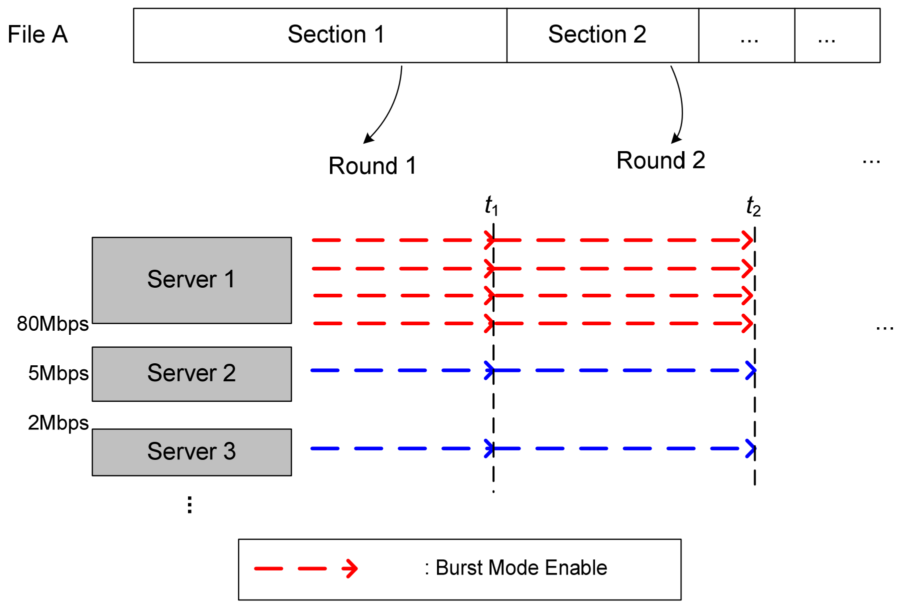


Figure 3-10: Co-Allocation of ARAM+

Chapter 4 Our Technology

4.1JSP and Servlet Technology

Servlets are the Java platform technology of choice for extending and enhancing Web servers. Servlets provide a component-based, platform-independent method for building Web-based applications. Servlets have access the entire family of Java APIs, such as the JDBC API to access enterprise databases. Also Servlets can access a library of HTTP-specific calls and receive all the benefits of the mature Java language, including portability, performance, reusability, and crash protection. JSP technology is an extension of the Servlet technology created to support authoring of HTML and XML pages. It makes it easier to combine fixed or static template data with dynamic content.

4.2MIFAS system component

In the previous file or download transfer and performance on the appraisal of the experimental study was designed to use the SUN Java programming experiment in the previous file download or transformer application we call Cyber Transformer, this experimental study to measure the units of each step throughout the Department to conduct the file transfer performance, so We chose to use JSP technology to complete our experiments, so we have a lot of rewriting the previous Cyber for the new WEB components, in order to achieve experimental purposes. In this research we use two respective experiment environments; one is running on cloud (Haddops) with HDFS over FTP, the other one is running on grid.

The illustrate (Seeing Figure 4-1) of MIFAS system, we used Hadoop Distributed File System (HDFS) as our distributed storage; we separate MIFAS as three parts as cloud architecture. The three parts of architecture respective as Infrastructure as a Service (IaaS); Platform as a Service (PaaS) and Software as a Service (PaaS), The HDFS as our replica storage, we use a JAVA API call as HDFS over FTP as our connector to communicate with Globus.

Globus as middle communicate ware, GridFTP as FTP base file transfer, J2EE as MIFAS system container and the top of illustrate is MIFAS Web System, it is user interface of web application.

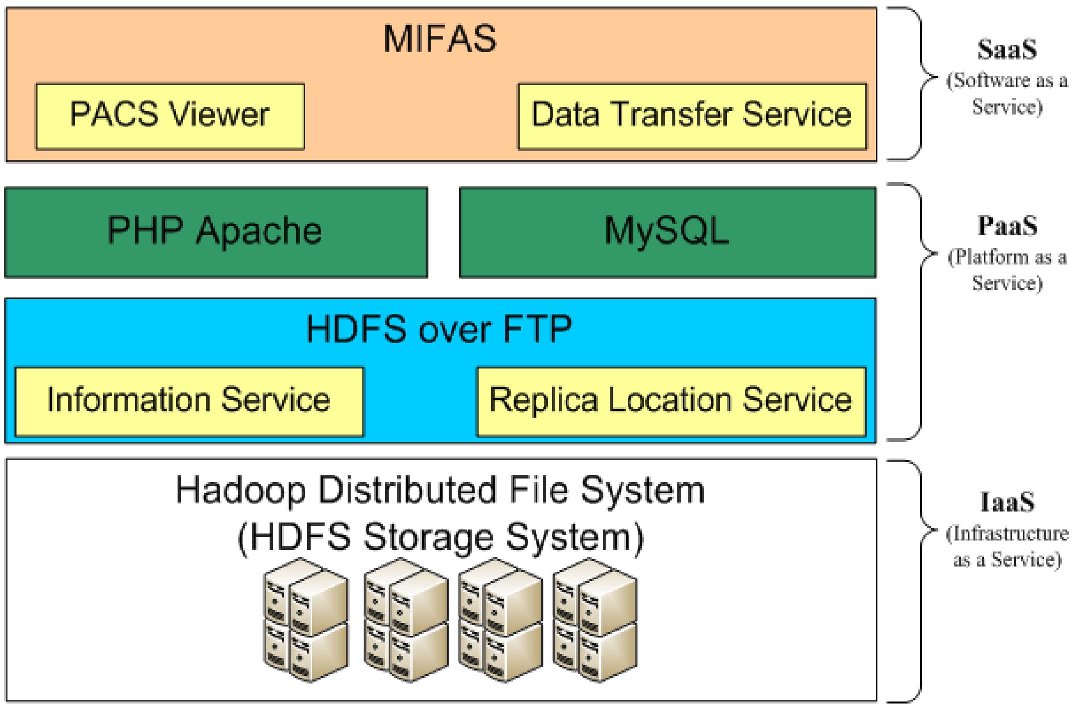


Figure 4-1: MIFAS working on HDFS over FTP

The illustrate (Seeing Figure 4-2) of MIFAS system, We used grid as our replica distributed file system storage, Globus as middle communicate ware, GridFTP as FTP base file transfer, J2EE as MIFAS system container and the top of illustrate is MIFAS Web System, it is user interface of web application.

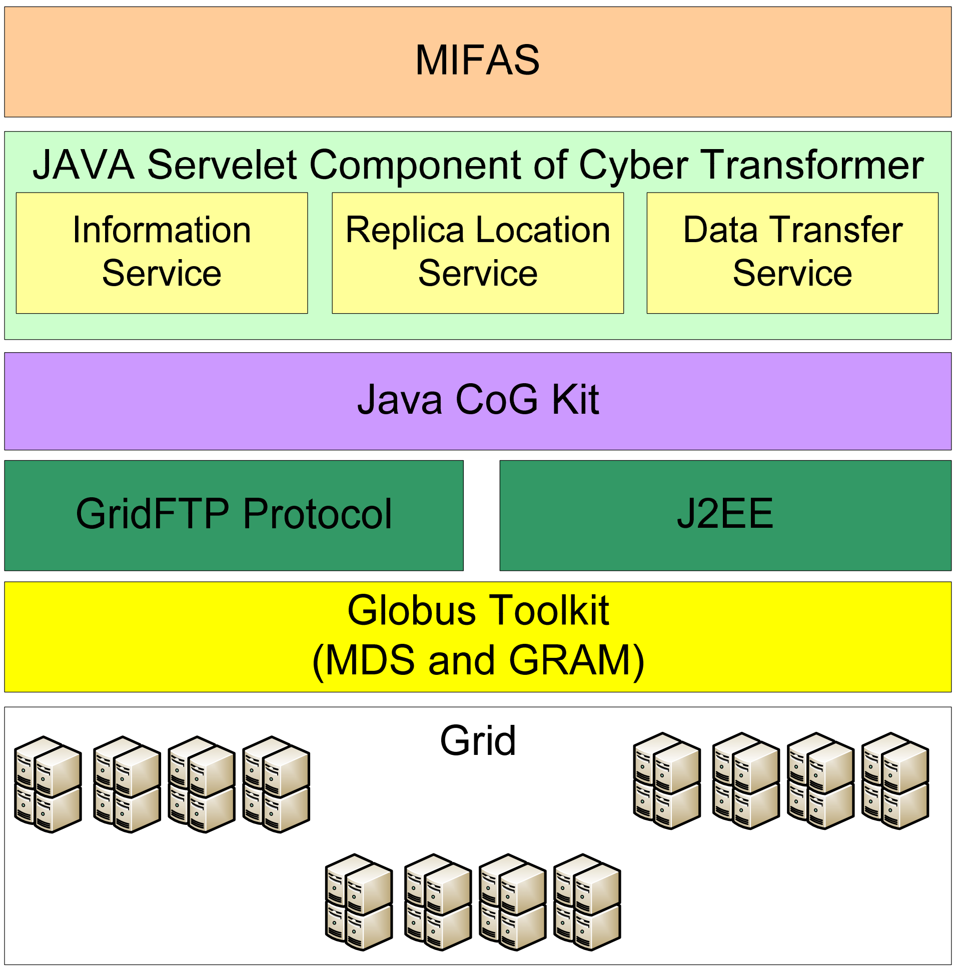


Figure 4-2: MIFAS working on grid

4.3 Hadoop Distributed File System

The HDFS (Hadoop Distributed File System) file system stores large files across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence does not require RAID storage on hosts. The HDFS file system is built from a cluster of data nodes, each of which serves up blocks of data over the network using a block protocol specific to HDFS. They also serve the data over HTTP, allowing access to all content from a web browser or other client. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high. A file system requires one unique server, the name node. This is a single point of failure for an HDFS installation. If the name node goes down, the file system is offline. When it comes back up, the name node must replay all outstanding operations. This replay process can take over half an hour for a big cluster.

4.4 MIFAS system transaction flow

The following of MIFAS Web system summary to illustrate (Seeing Figure 4-3) the process of MIFAS Web system transaction flow. First user key-in input their username and password to authenticate whether they can go to the MIFAS Web system or not, second the user can select left list or input search condition to query the patient a list of information and list of information support some icons to view or download some files like as image or some data, third when system list user information, user can direct to view patient’s PACS temp view image, forth user can use MIFAS configuration system to configured MIFAS co-allocation for example the user can configured the co-allocation scheme which they want or use also can configured server local ate …etc, fifth when user want to start download PACS system will query MIFAS configuration database to initial some work for download working, the last system will reference MIFAS configuration to select suit co-allocation algorithm to download PACS image.

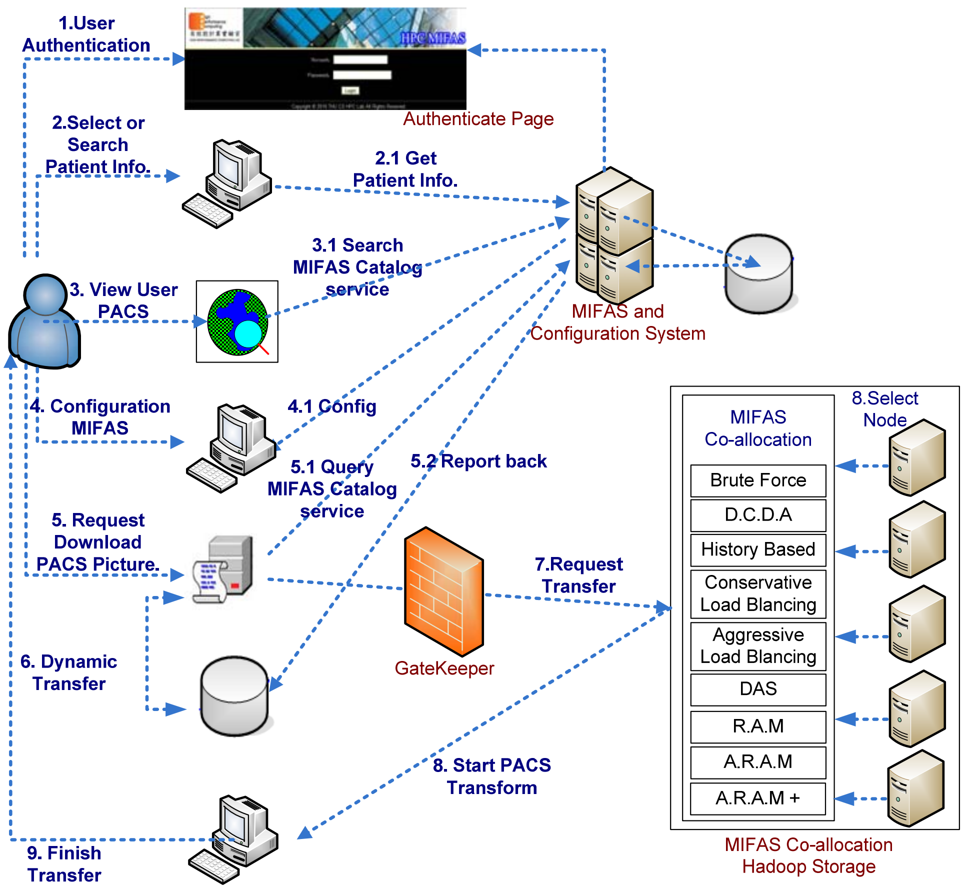


Figure 4-3: MIFAS transaction flow

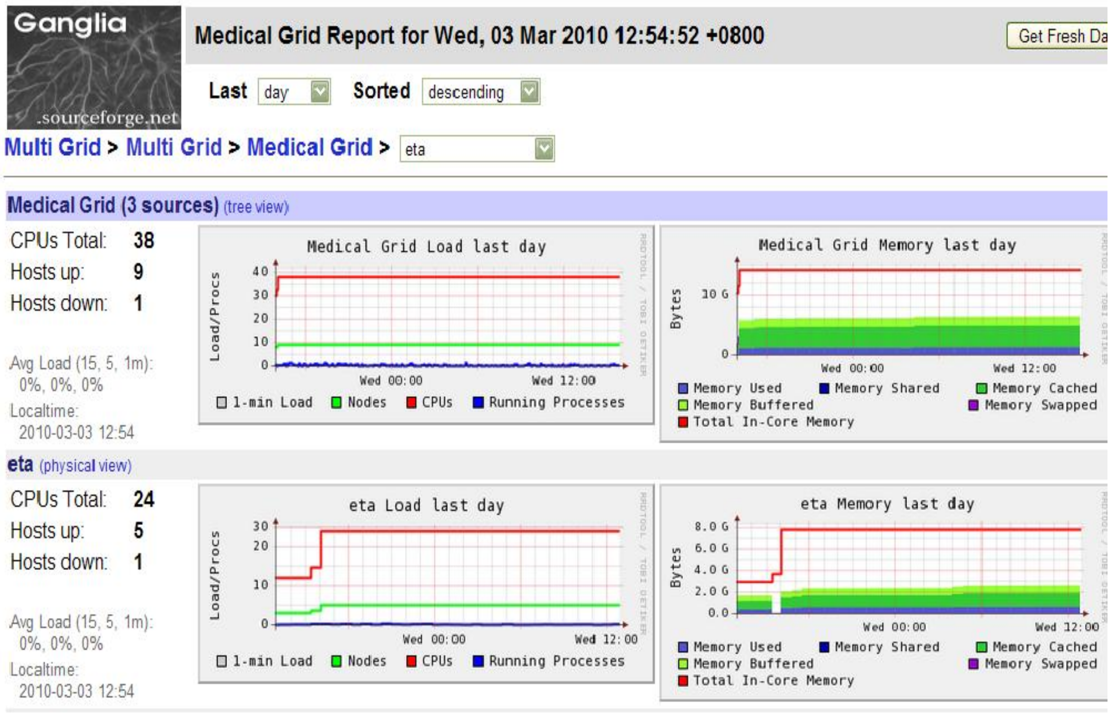
4.5 Our Test Tools

In this research we used some monitor performance and download tools for performance monitor tools we used Ganglia monitor’s tools for each node computing performance measure, for download tools we also develop and translate Cyber Transformer as our tools we call the new tool name as MIFAS System as our client web system to test file download, this tool not only be a file downloader also be a PACS viewer.

4.6 Ganglia Monitor

The Ganglia is a distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization.

It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. In this research we used the ganglia monitor system to help us to monitor storage, memory and CPU usage to measure each node performance. As below picture (Seeing Figure 4-4) is our experimental ganglia monitor web page.


Ganglia .sourceforge.net

Medical Grid Report for Wed, 03 Mar 2010 12:54:52 +0800

Last day Sorted descending

Multi Grid > Multi Grid > Medical Grid > eta

Medical Grid (3 sources) (tree view)

CPUs Total: 38
Hosts up: 9
Hosts down: 1

Avg Load (15, 5, 1m): 0%, 0%, 0%
Localtime: 2010-03-03 12:54

eta (physical view)

CPUs Total: 24
Hosts up: 5
Hosts down: 1

Avg Load (15, 5, 1m): 0%, 0%, 0%
Localtime: 2010-03-03 12:54

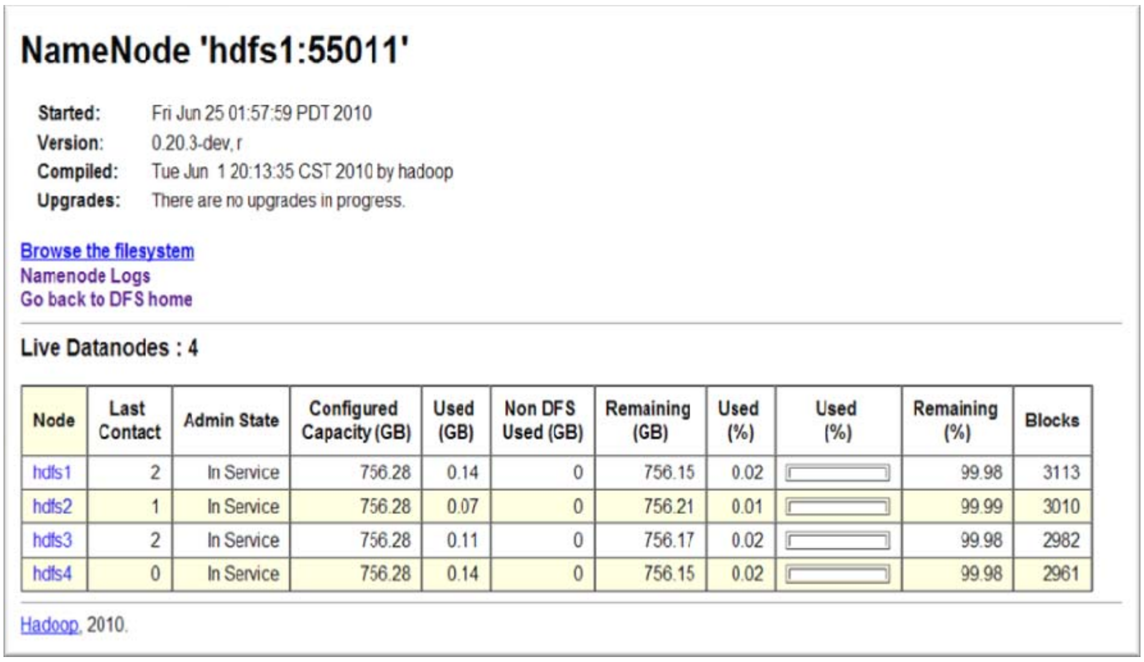
Figure 4-4: Ganglia Monitor

4.7 Log4J

To measure the performance of data transfer speed and transaction time in MIFAS we not only use the tools of ganglia but also used Apache Log4J. Logging equips the developer with detailed context for application failures. On the other hand, testing provides quality assurance and confidence in the web application. With log4j it is possible to enable logging at runtime without modifying the application binary. The log4j package is designed so that these statements can remain in shipped code without incurring a heavy performance cost. Logging behavior can be controlled by editing a configuration file, without touching the application binary.

4.8 Hadoop over FTP

In this research we conduct two kinds of back-end replica storage; the first one is MIFAS working on grid and the second is HDFS, as below illustrate Figure 4-5 and Figure 4-6 is our second experimental environment, this environment we used *hdfs1* and *oct1* (seeing Table 6-2: **Experimental Machine List on Cloud**) as all HDFS name node the *hdfs1*and *oct1* also is a FTP server so that all of replica file in other HDFS data node server will be collect in *hdfs1* or *oct1*.



NameNode 'hdfs1:55011'

Started:	Fri Jun 25 01:57:59 PDT 2010
Version:	0.20.3-dev, r
Compiled:	Tue Jun 1 20:13:35 CST 2010 by hadoop
Upgrades:	There are no upgrades in progress.

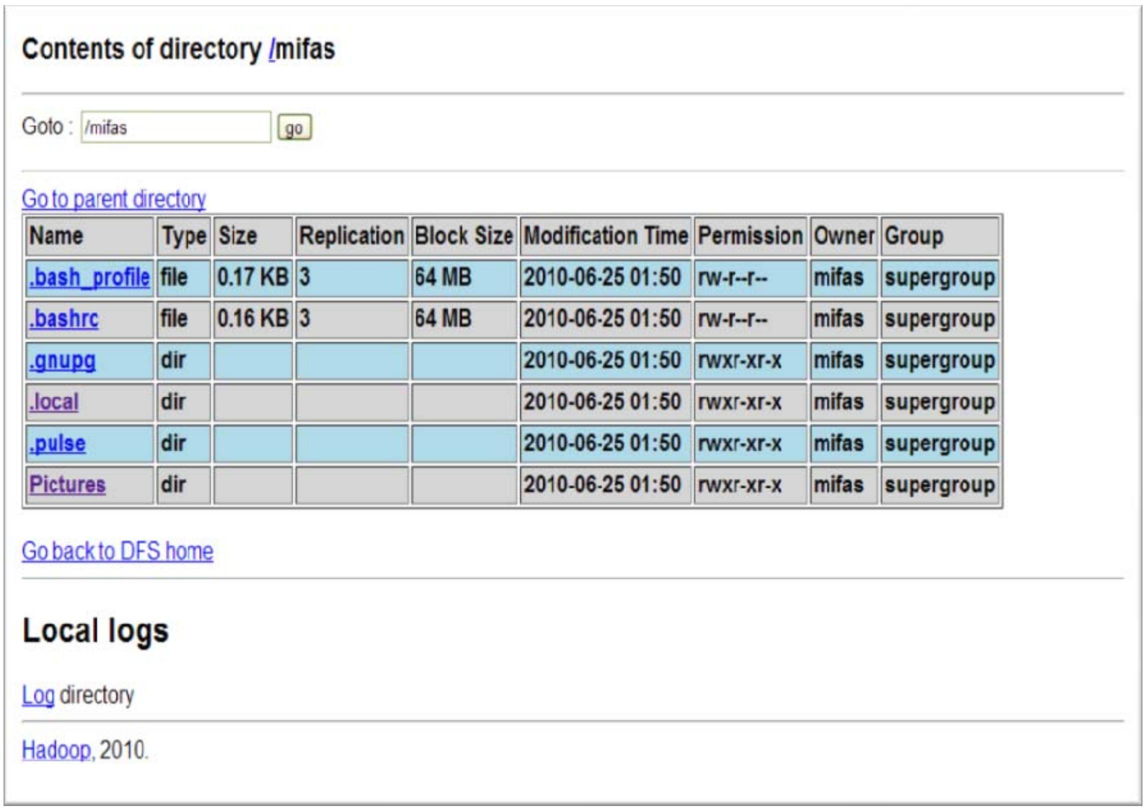
Browse the filesystem
Namenode Logs
Go back to DFS home

Live Datanodes : 4

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
hdfs1	2	In Service	756.28	0.14	0	756.15	0.02	▭	99.98	3113
hdfs2	1	In Service	756.28	0.07	0	756.21	0.01	▭	99.99	3010
hdfs3	2	In Service	756.28	0.11	0	756.17	0.02	▭	99.98	2982
hdfs4	0	In Service	756.28	0.14	0	756.15	0.02	▭	99.98	2961

Hadoop, 2010.

Figure 4-5: Hadoop HDFS status page


Contents of directory /mifas

Goto : /mifas go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
.bash_profile	file	0.17 KB	3	64 MB	2010-06-25 01:50	rw-r--r--	mifas	supergroup
.bashrc	file	0.16 KB	3	64 MB	2010-06-25 01:50	rw-r--r--	mifas	supergroup
.gnupg	dir				2010-06-25 01:50	rwxr-xr-x	mifas	supergroup
.local	dir				2010-06-25 01:50	rwxr-xr-x	mifas	supergroup
.pulse	dir				2010-06-25 01:50	rwxr-xr-x	mifas	supergroup
Pictures	dir				2010-06-25 01:50	rwxr-xr-x	mifas	supergroup

Go back to DFS home

Local logs

Log directory

Hadoop, 2010.

Figure 4-6: File status in one of HDFS

Chapter 5 MIFAS System

5.1 MIFAS User Client System

In this research, we developed a system call: MIFAS [32]. In usually we divided MIFAS system as two parts, the first part is MIFAS main system and the other part is MIFAS system, Figure 5-1 and Figure 5-2 show the MIFAS configuration system main page.

MIFAS system not only a downloader it also a mini PACS system, in Figure 5-1 show the left area the user can select any or all type examine type to list search condition in examine list, the user also can key in some search condition to search some result.

MIFAS systems not only above description function, there are four powerful functions in MIFAS were file status, image preview, comment and download.

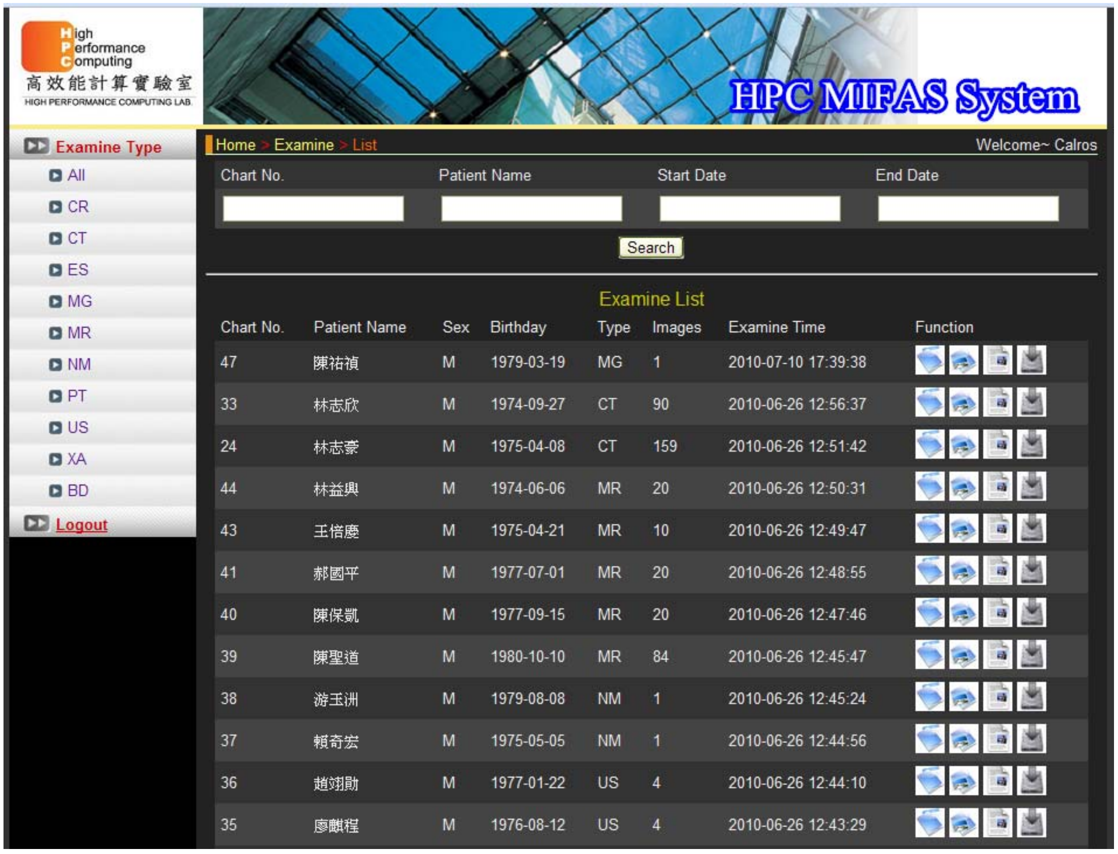
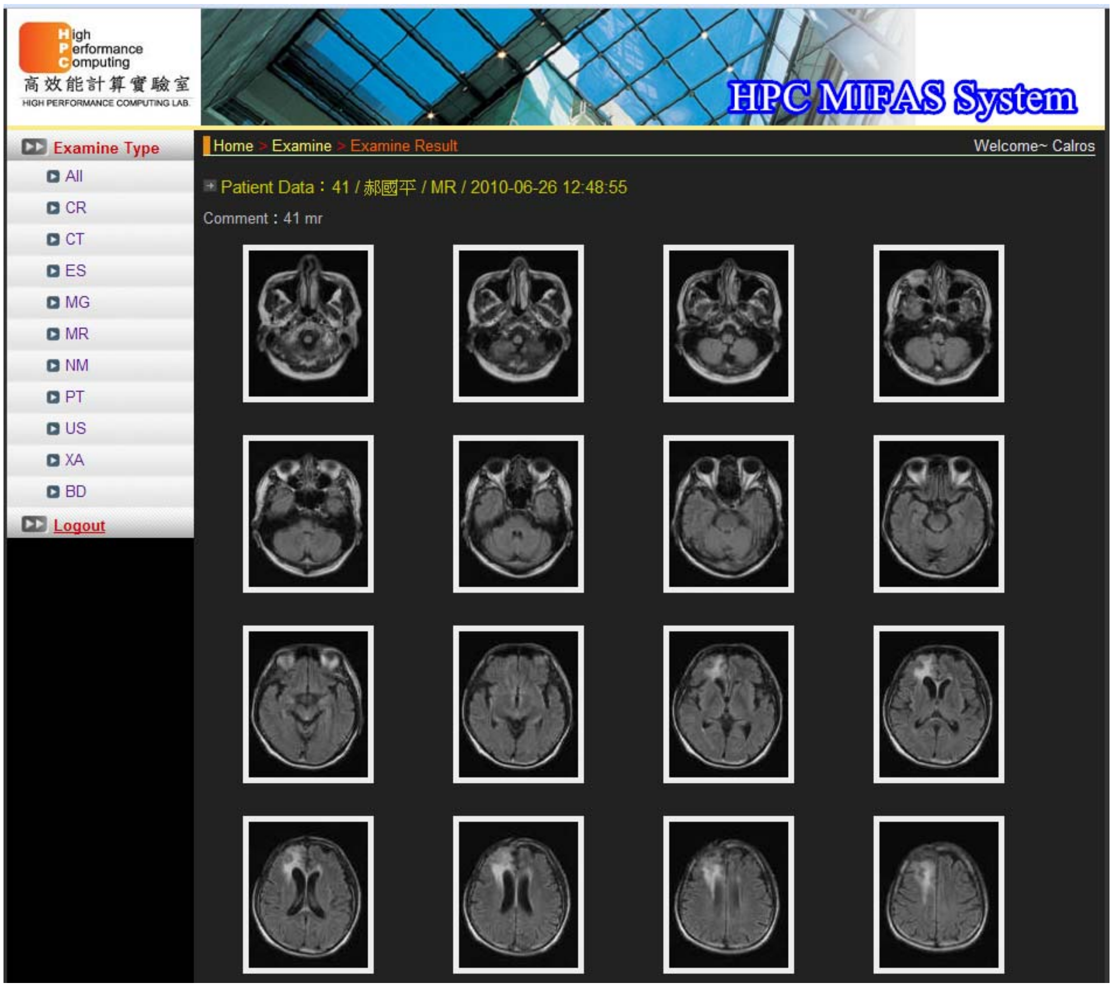


Chart No.	Patient Name	Start Date	End Date

Examine List

Chart No.	Patient Name	Sex	Birthday	Type	Images	Examine Time	Function
47	陳祐禎	M	1979-03-19	MG	1	2010-07-10 17:39:38	
33	林志欣	M	1974-09-27	CT	90	2010-06-26 12:56:37	
24	林志豪	M	1975-04-08	CT	159	2010-06-26 12:51:42	
44	林益興	M	1974-06-06	MR	20	2010-06-26 12:50:31	
43	王信慶	M	1975-04-21	MR	10	2010-06-26 12:49:47	
41	郝國平	M	1977-07-01	MR	20	2010-06-26 12:48:55	
40	陳保凱	M	1977-09-15	MR	20	2010-06-26 12:47:46	
39	陳聖道	M	1980-10-10	MR	84	2010-06-26 12:45:47	
38	游玉洲	M	1979-08-08	NM	1	2010-06-26 12:45:24	
37	賴奇宏	M	1975-05-05	NM	1	2010-06-26 12:44:56	
36	趙翊勛	M	1977-01-22	US	4	2010-06-26 12:44:10	
35	廖麒程	M	1976-08-12	US	4	2010-06-26 12:43:29	

Figure 5-1: MIFAS User Client System


High Performance Computing
高效能計算實驗室
HIGH PERFORMANCE COMPUTING LAB.
HPC MIFAS System
Examine Type
All
CR
CT
ES
MG
MR
NM
PT
US
XA
BD
Logout
Home > Examine > Examine Result
Welcome~ Calros

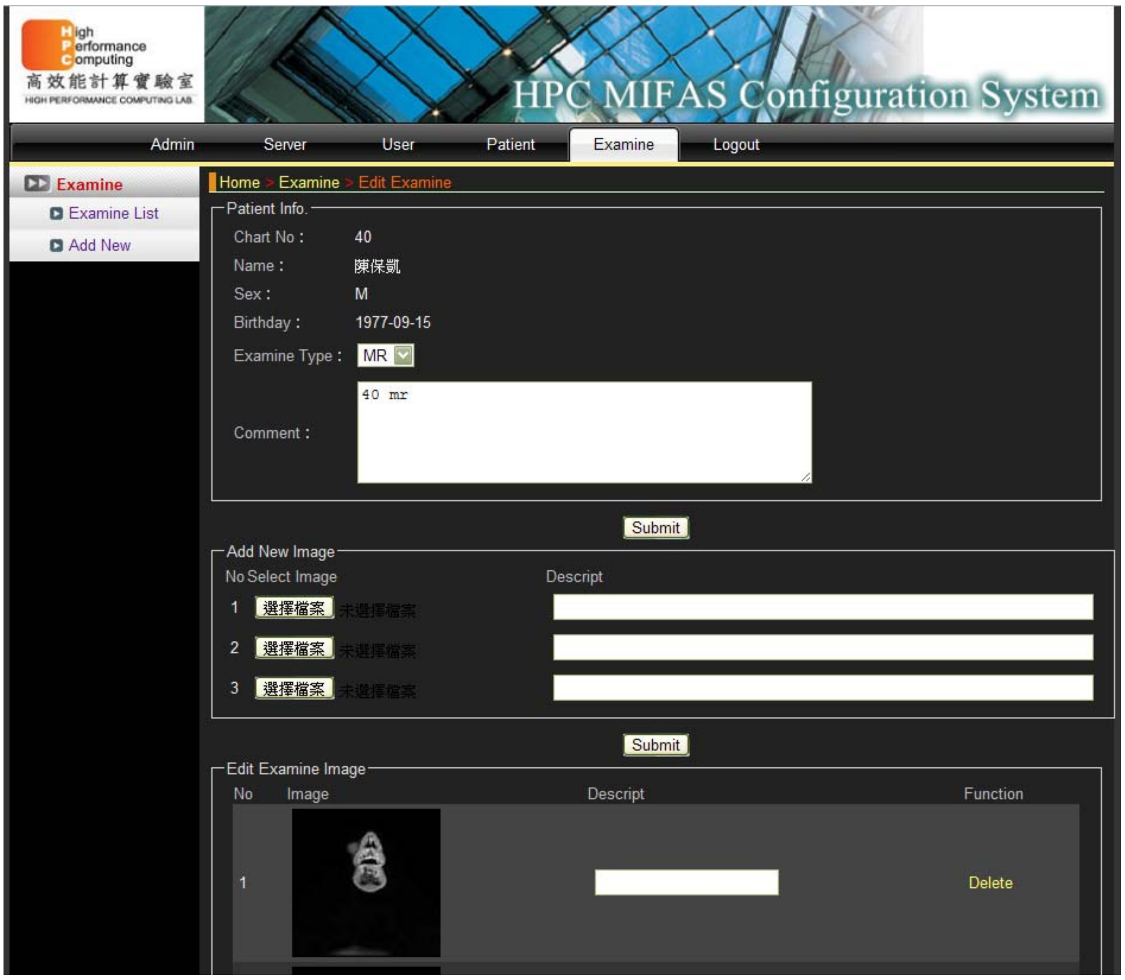
Patient Data：41 / 郝國平 / MR / 2010-06-26 12:48:55

Comment：41 mr

Figure 5-2: Mini PACS in MIFAS

5.2 MIFAS Configuration System

The MIFAS Configuration System is to configure MIFAS[32][33], It consist of five functions: (1) Admin: this function is configure the configurations system who can login to modify, (2) Server: this function is configure the back-end storage, in this research we used HDFS as our back-end replica distribution storage access file, (3) User: this function is configure who can login MIFAS main system but not include MIFAS configuration system, (4) Patient: this function is configure patient information, (6) Examine: this function is configure patient image data, it also support image batch upload and data modify include: modify, upload image and delete image.



High Performance Computing
高效能計算實驗室
HIGH PERFORMANCE COMPUTING LAB
HPC MIFAS Configuration System
Admin
Server
User
Patient
Examine
Logout
Examine
Examine List
Add New
Home > Examine > Edit Examine
Submit
Submit
選擇檔案
Delete

Patient Info.

Chart No :	40
Name :	陳保凱
Sex :	M
Birthday :	1977-09-15
Examine Type :	MR
Comment :	40 mr

Add New Image

No	Select Image	Descript
1	選擇檔案 未選擇檔案	
2	選擇檔案 未選擇檔案	
3	選擇檔案 未選擇檔案	

Edit Examine Image

No	Image	Descript	Function
1			Delete

Figure 5-3: MIFAS Configuration system

Chapter 6 Experimental Results

6.1 Experimental Environments

The experiments in this work were conducted and evaluated two environments, the first environment is MIFAS on grid environment (seeing Figure 6-1 and Table 6-1) and other one is MIFAS on the Hadoop environment (seeing Figure 6-2 and Table 6-2), the first environment which consists of more than 40 processors distributed over 10 clusters located at 2 educational institutions and 1 hospital institutions (Tunghai University—THU, Chung Shan Medical University—CSMU, and Chung Shan Hospital Hospital—CSH.). Figure 6-1 shows statuses for all machines used in the grid test bed on one Ganglia monitor web page (Seeing Figure 4-4). All of institution network are interconnected by the 1 Gbps Taiwan Academic Network (TANET). The Grid platform is built around 14 computing nodes, more than 40 CPUs with differing clock rate and storage (Seeing Table 6-1), and total storage of more than 30 GB. All the institutions are in Taiwan, Two of institutions CSH and CSMU are nearby each other, but CSH and CSMU's distance from THU at least 10 kilometer. All nodes on Grid have installed Globus 4.2.1 or newest version. We performed wide-area data transfer experiments using MIFAS Hadoop Parallel Download System, our MIFAS system tool, on our co-allocation test bed at Tunghai University (THU), Chung Shan Medical University (CSMU), and Chung Shan Hospital Hospital (CSH). For measure fetch data speed and performance we let each client to fetch each replica data server.

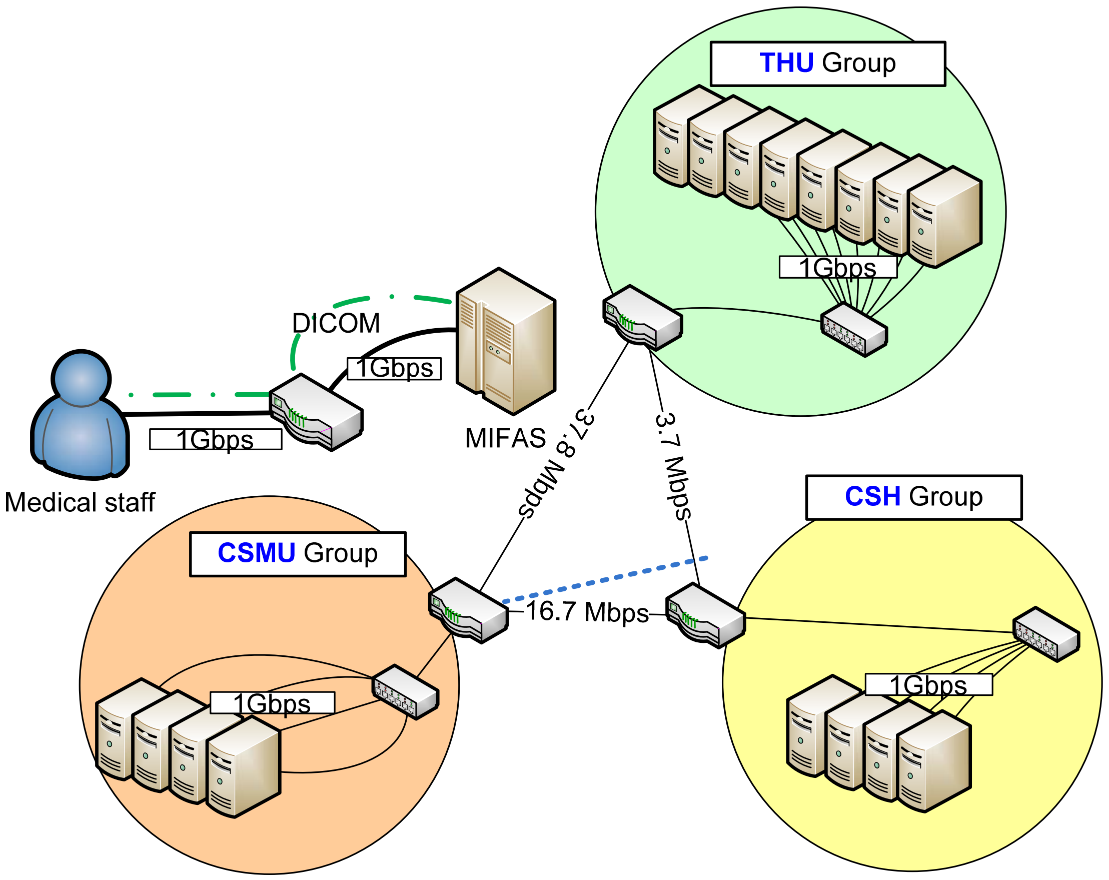


Figure 6-1: Experimental Environment on grid

The second experimental environment which consists about 10 processors distributed over 10 clusters located at THU (Tunghai University—THU.) Figure 6-2 shows statuses for all machines used in the cloud test bed.

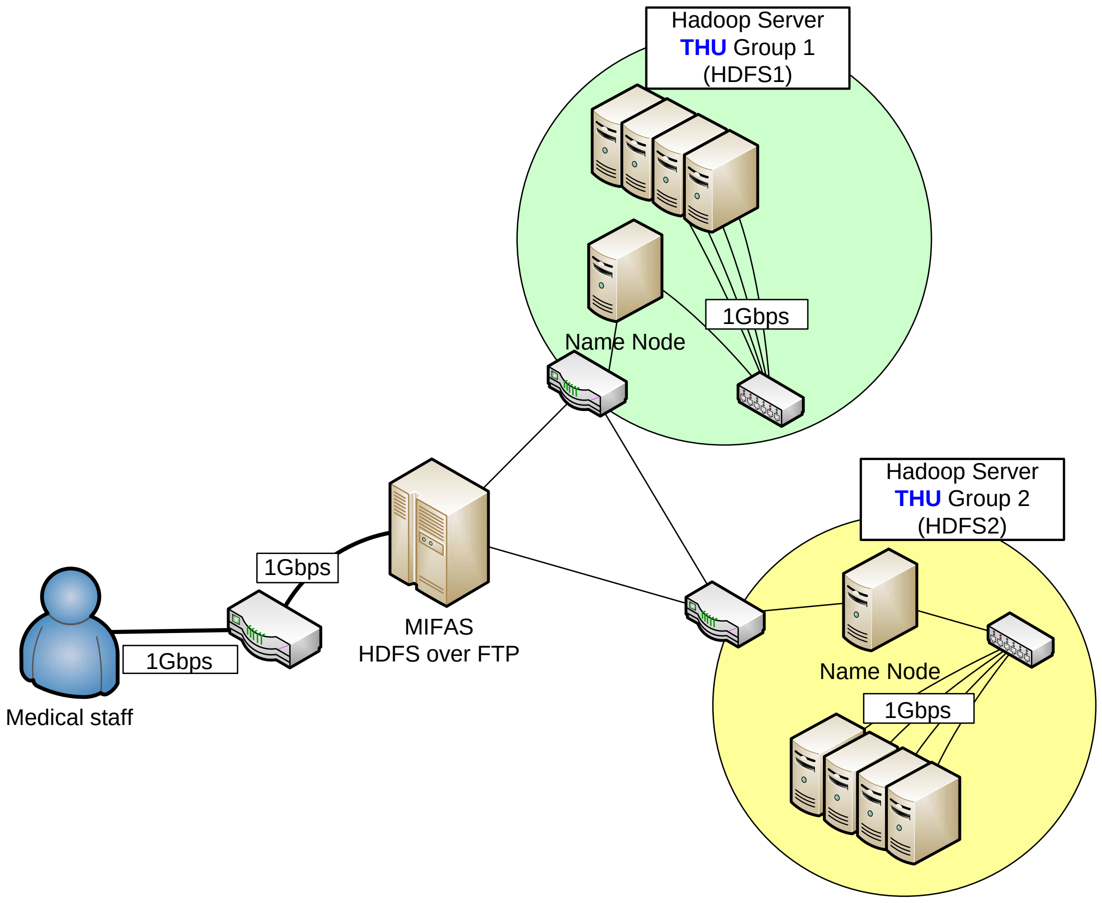


Figure 6-2: Experimental Environment on cloud

Table 6-1: Experimental Machine List on Grid

Server Locate	Server Name	CPU	RAM	HDD	Network
CSMU	C1	Intel CD2 E7500	2GB	500GB	10/100/1G
	C2	Intel CD2 E7500	2GB	500GB	10/100/1G
	C3	Intel CD2 E7500	2GB	500GB	10/100/1G
	C4	Intel CD2 E7500	2GB	500GB	10/100/1G
CSH	G1	Intel CD2 E7500	2GB	500GB	10/100/1G
	G2	Intel CD2 E7500	2GB	500GB	10/100/1G
	G3	Intel CD2 E7500	2GB	500GB	10/100/1G
THU	ETA1	Intel Xeon DC	1GB	60GB	10/100/1G
	ETA2	Intel Xeon DC	1GB	60GB	10/100/1G
	ETA3	Intel Xeon DC	1GB	60GB	10/100/1G
	ETA4	Intel Xeon DC	1GB	60GB	10/100/1G
	ETA5	Intel Xeon DC	1GB	60GB	10/100/1G
	ETA6	Intel Xeon DC	1GB	60GB	10/100/1G
	ETA7	Intel Xeon DC	1GB	60GB	10/100/1G
	ETA8	Intel Xeon DC	1GB	60GB	10/100/1G
	ETA9	Intel Xeon DC	1GB	60GB	10/100/1G

*Intel CD2 E7500 = Intel Core Duo2 E7500 2.93GHz.

*Intel Xeon DC = Intel Xeon Dual Core 2.0 ~ 3.2GHz.

*Network 10/100/1G = 10/100/1000Mbps.

Table 6-2: Experimental Machine List on Cloud

Server Locate	**Server Name**	**CPU**	**RAM**	**HDD**	**Network**
THU Group1	hdfs1	Intel Core i7	1GB	756GB	10/100/1G
	hdfs2	Intel Core i7	1GB	756GB	10/100/1G
	hdfs3	Intel Core i7	1GB	756GB	10/100/1G
	hdfs4	Intel Core i7	1GB	756GB	10/100/1G
THU Group2	oct1	Intel CD2 E7500	2GB	340GB	10/100/1G
	oct3	Intel CD2 E7500	2GB	340GB	10/100/1G
	oct4	Intel CD2 E7500	2GB	565GB	10/100/1G
	oct5	Intel CD2 E7500	2GB	5390GB	10/100/1G
*HDFS1 = THU GROUP1 Name Node *OCT5 = THU GROUP2 Name Node *Network 10/100/1G = 10/100/1000Mbps.					

6.2 Experimental Results

Before start some test cases we getting simulate some measure value of network fluctuate like as normal network fluctuate bandwidth and line broken status, in this research we measure the three institutes' (THU, CSH and CSMU) network bandwidth. The following illustrate chart for us in each institutes site, the average transfer rate, which is very clear that we can see that THU and CSMU is carried out through academic network file transfers, the rate even faster than any other networks bandwidth.

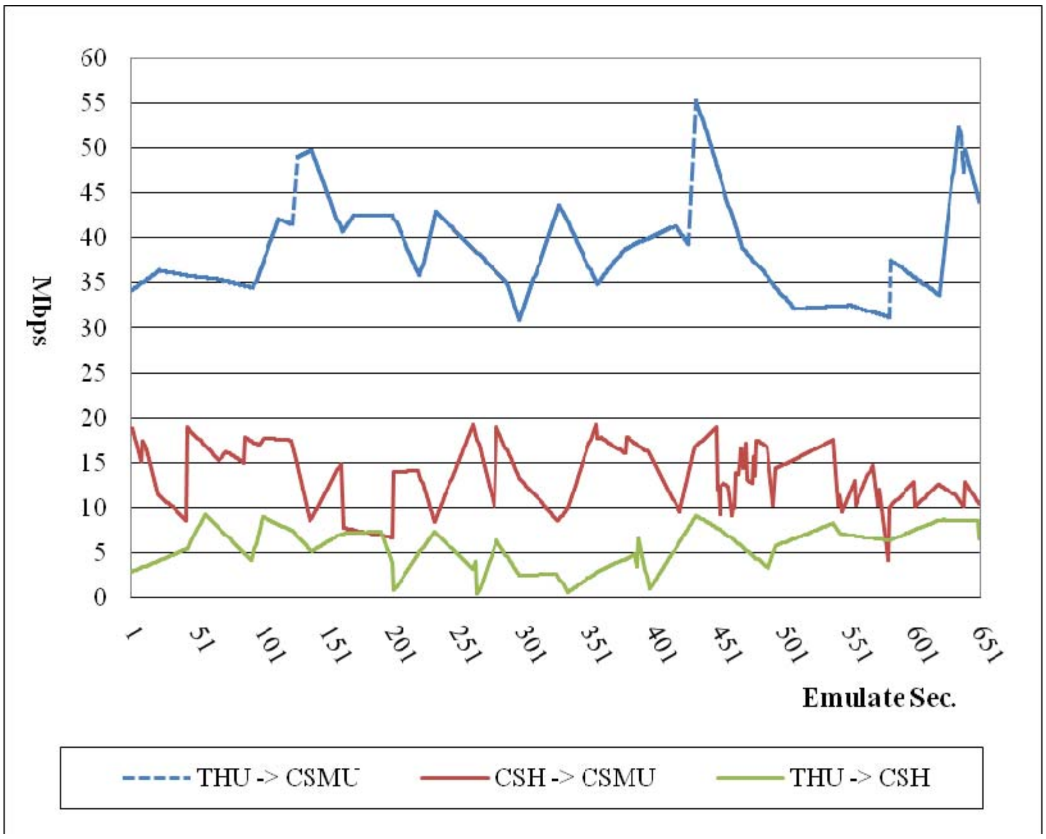


Figure 6-3: Normal Fluctuate Network Bandwidth

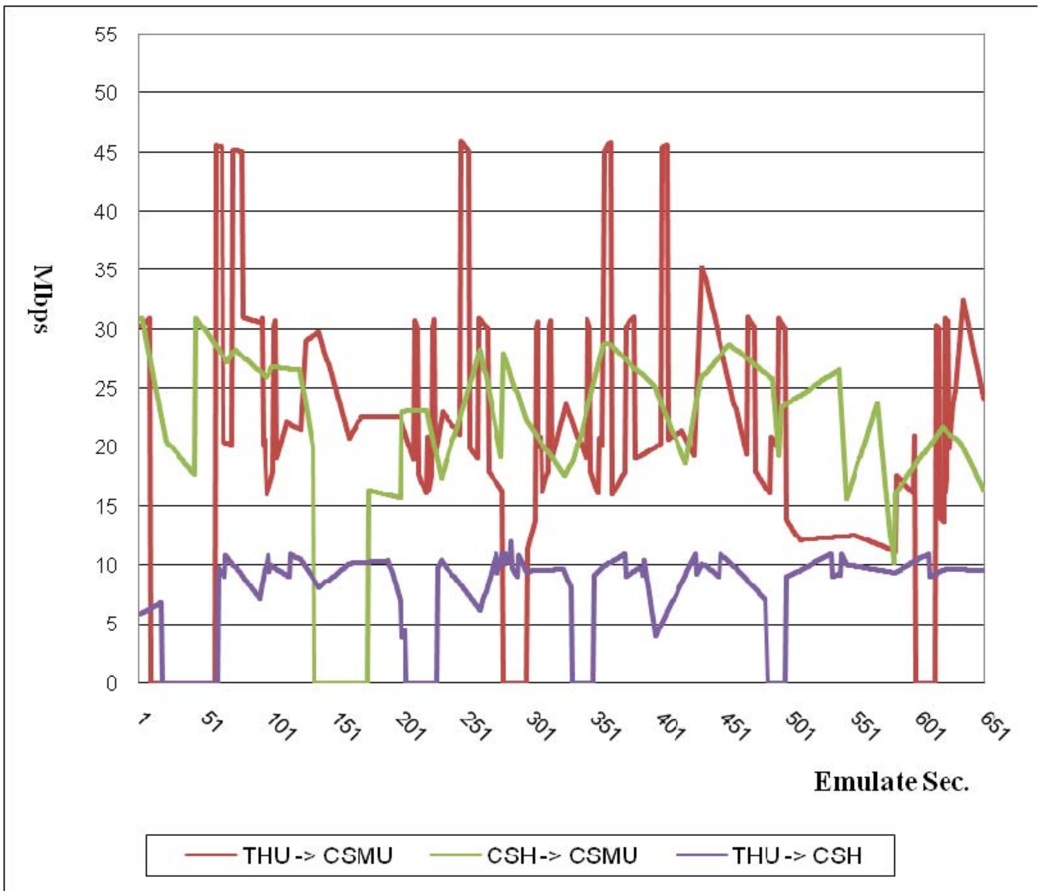


Figure 6-4: Emulate Unstable Fluctuate Network Bandwidth

In this research we will prepared three test cases for experimental to measure MIFAS Hadoop system download speed (time) and performance, in first test we will measure all co-allocation scheme with difference file sizes (10mega bytes to 1500mega bytes) in all site at normal fluctuate network bandwidth (seeing Figure 6-3), second test case we will measure all co-allocation scheme with difference file size (10 mega bytes to 1500 mega bytes) in unstable (broken line) network bandwidth (seeing Figure 6-4), finally we will measure all MIFAS Hadoop co-allocation scheme with difference file size (10mega bytes to 1000mega bytes) average download speed and test performance in each co-allocation scheme.

In this experiment, we performed several tests by downloading difference file sizes (10 mega bytes to 1500 mega bytes) using night algorithms of MIFAS Hadoop Co-allocation. The test results show the best algorithms of the night transmissions changes depending on the file size downloaded. Figure 6-5 shows the data for different download sizes (10 mega bytes to 1500 mega bytes) in night file transfer modes. In Figure 6-5, we can clearly see that there is no difference in the night algorithms for downloading 10 mega bytes file. In this case, using different ways for query or file retrieval, the MIFAS Hadoop Co-allocation algorithms would be completed around the same time. From the figure, we determine the best transmission method for file size over 50 mega bytes to 1500 mega bytes, is ARAM+ algorithm. From this experiment, we find the best transmission method and then use this to perform other experiments.

The first test case is to measure the completion time of download speed of our proposed technique in MIFAS Hadoop system in normal network bandwidth, in this test case we implemented MIFAS system in all institute and configured MIFAS configuration in following nine kinds of co-allocation schemes: Brute-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+). We use some test model (download files for 10 mega bytes ~ 1500 mega bytes size) for nine kinds of co-allocation scheme to average count the completion time in download speed. As below illustrate can easily to analyzed those nine kinds of co-allocation scheme performance by comparing transfer finish times and overall performance, as shown Figure 6-5.

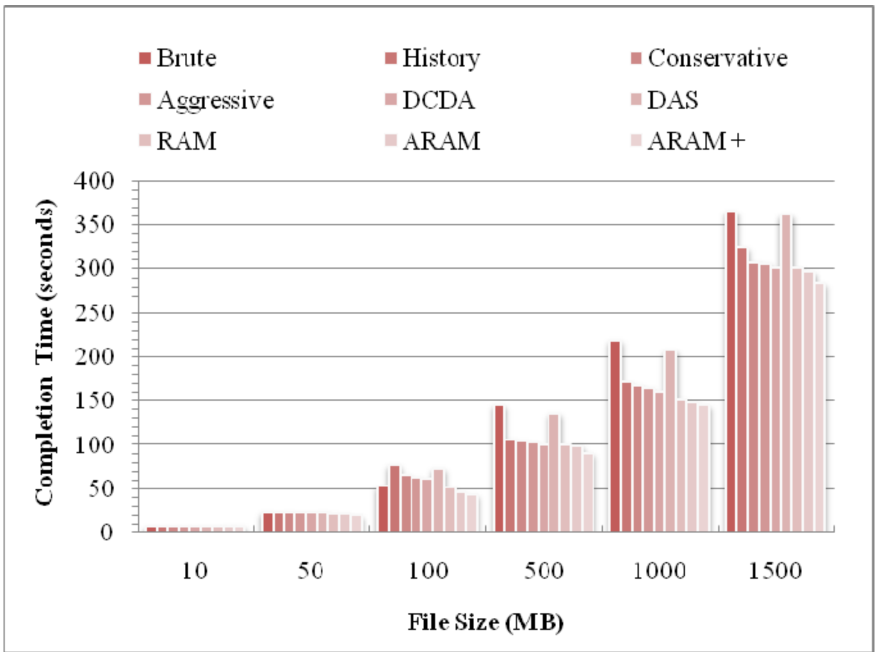


Figure 6-5: 9 kinds of Co-Allocation Compare in normal network

The second test case is to measure the completion time of download speed of our proposed technique in MIFAS Hadoop system in an unstable network bandwidth, in this test case we implement MIFAS system in each institute and also configured MIFAS configuration in nine kinds of co-allocation scheme, also we used difference file size (10mega bytes to 1500mega bytes) for file transfer or download. To take into consider this test case the each institute network connect with Taiwan Academic Network (TANET) As below illustrate can easily to analyzed those nine kinds of co-allocation scheme performance by comparing transfer finish times and overall performance, as shown Figure 6-9.

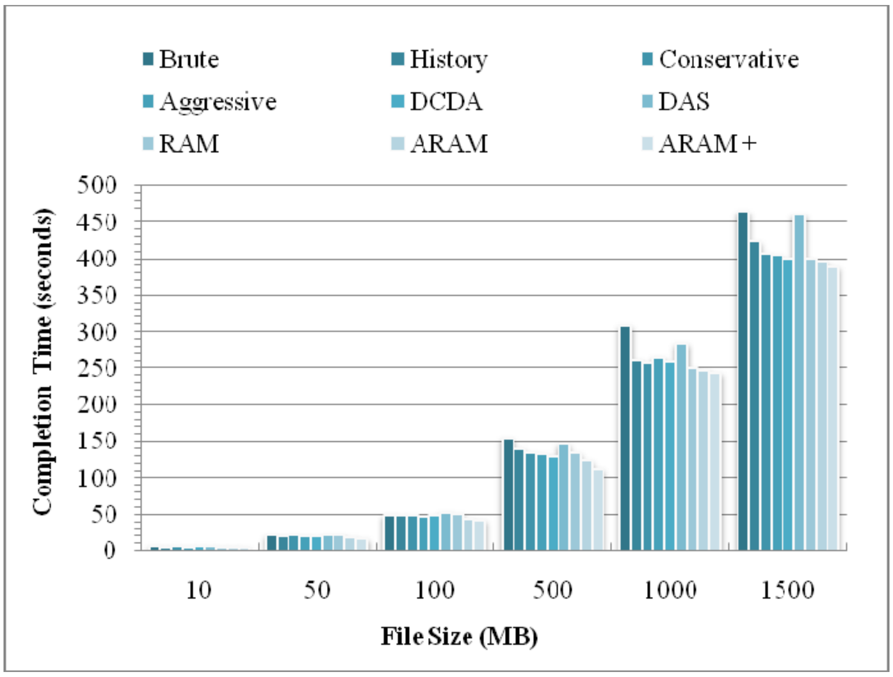


Figure 6-6: 9 kinds of Co-Allocation in emulate unstable network

The final test case is to measure the all co-allocation scheme average download time and performance, in this case we use 100 mega bytes to 1000 mega bytes size file to average the download speed and measure its performance.

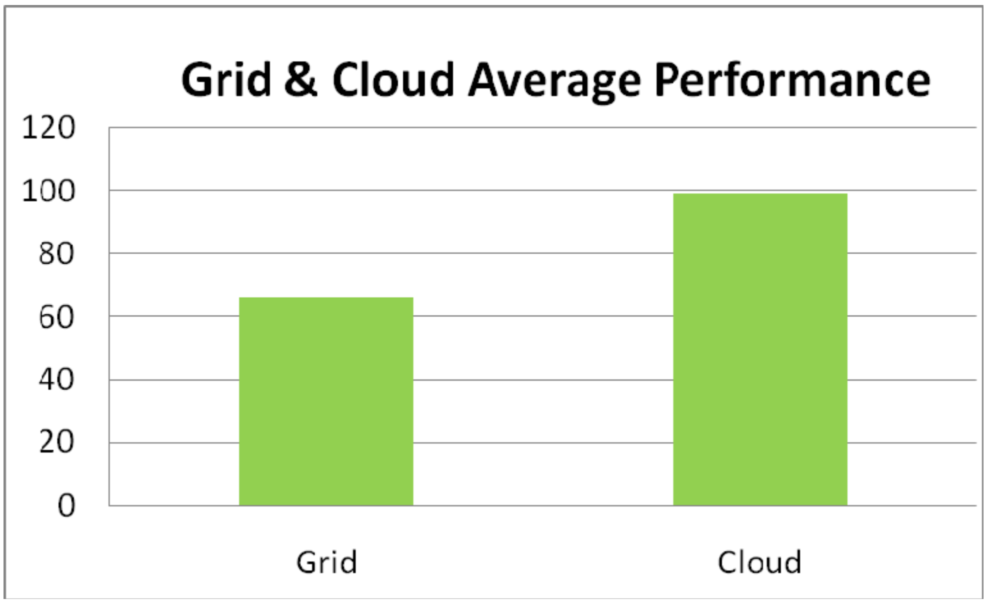


Figure 6-7: Average Performance of Grid and Cloud

Chapter 7 Conclusion

7.1Conclusion and Future work

In this research we implemented a web-system call as MIFAS system as our file transfer and download platform this system not only be a transfer and downloader it can use to manage file and PACS viewer. In this research we also developed another system call as MIFAS configuration system it not only can help MIFAS system to configuration that also can manage file server node configuration, upload batch file, dispatch file, manage all file in some of node server and configuration the co-allocation scheme which want to used. In this research we used Hadoop Distributed File System (HDFS) as our back-end cloud file storage manager, we used nine kind of file co-allocation scheme to test file transfer and download, there are three institute (THU, CSH and CSMU) to support us this research measure the rate of download speed and performance. Finally we used MIFAS system to deploy in each institute to test our co-allocation scheme, we can easily in normal network bandwidth maybe effect the file transfer and download speed, we can use some co-allocation scheme to overcome some obstacle like as: network bandwidth, CPU idle time usage , memory usage and etc. In this research future we can develop a smart co-allocation like as smart broker; this can help them to manage which status of environment is the best to use some co-allocation or before file transfer start the broker will pre-initial and detect some network environment to select the best download speed and performance.

Table 7-1: Compare Grid and MIFAS Cloud

	Grid	**MIFAS Cloud**
Platform	J2SE	Apache PHP
Develop Platform	Java	PHP…etc
DB	MySQL	MySQL
Infrastructure Construct	hard	easy
SaaS	limit	non-limit

7.2Future work

MIFAS is a web-base parallel transferring system with HDFS cloud storage system by HDFS over FTP, this co-allocation parallel transferring architecture is working on random, at future work we can enhance MIFAS parallel transferring by grid co-allocation algorithm.

Bibliography

[1] 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.

[2] 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.

[3] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid information services for distributed resource sharing, in: Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing, HPDC-10’01, August 2001, pp. 181–194.

[4] Global Grid Forum, http://www.ggf.org/.

[5] IBM Red Books, Introduction to Grid Computing with Globus, IBM Press. www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Chao-Tung Yang, Chiu-Hsiung Chen, and Ming-Feng Yang, “Implementation of a Medical Image File Accessing System in Co-allocation Data Grids”, Future Generation Computer Systems, July 2010.

[13] Chao-Tung Yang, Chiu-Hsiung Chen, Ming-Feng Yang, and Wen-Chung Chiang, “MIFAS: Medical Image File Accessing System in Co-allocation Data Grids”, Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference, Ilan, Taiwan, December 2008.

[14] Chao-Tung Yang, I-Hsien Yang, and Chun-Hsiang Chen, “RACAM: Design and Implementation of a Recursively-Adjusting Co-Allocation Method with Efficient Replica Selection in Data Grids”, Concurrency and Computation: Practice and Experience, May 2010.

[15] Chao-Tung Yang, Yao-Chun Chi, Ming-Feng Yang, and Ching-Hsieh Hsu, “An Anticipative Recursively-Adjusting Mechanism for Parallel File Transfer in Data Grids”, Concurrency and Computation: Practice and Experience, May 2010.

[16] Chao-Tung Yang, Ming-Feng Yang and Wen-Chung Chiang, “Enhancement of Anticipative Recursively-Adjusting Mechanism for Parallel File Transfer in Data Grids”, Journal of Network and Computer Applications, Volume 32, Issue 4, July 2009, Pages 834-845.

[17] Chao-Tung Yang, I-Hsien Yang, Shih-Yu Wang, Ching-Hsien Hsu and Kuan-Ching Li, “A Recursively-Adjusting Co-Allocation Scheme with a Cyber-Transformer in Data Grids”, Future Generation Computer Systems, Elsevier B.V., Volume 25, Issue 7, July 2009, Pages 695-703.

[18] Chao-Tung Yang, Ming-Feng Yang, Lung-Hsing Cheng, and Wen-Chung Chiang, “Enhancement of Anticipative Recursively-Adjusting Mechanism for Redundant Parallel File Transfer in Data Grids”, Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems, Australia December 2008.

[19] Chao-Tung Yang, Ming-Feng Yang and Wen-Chung Chiang, “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, 24-26 October 2008 in Shenzhen, Guangdong, China.

[20] Chao-Tung Yang, Yao-Chun Chi, Ming-Feng Yang and Ching-Hsien Hsu, "An Anticipative Recursively-Adjusting Mechanism for Redundant Parallel File Transfer in Data Grids", Proceedings of the Thirteenth IEEE Asia-Pacific Computer Systems Architecture Conference (ACSAC 2008), Aug. 4-6, 2008 in Hsinchu, Taiwan.

[21] Chao-Tung Yang, I-Hsien Yang, Kuan-Ching Li and Shih-Yu Wang, "Improvements on Dynamic Adjustment Mechanism in Co-Allocation Data Grid Environments", Journal of Supercomputing, Springer Netherlands, vol. 40, no. 3, pp. 269-280, June 2007.

[22] Chao-Tung Yang, Yao-Chun Chi and Chun-Pin Fu, "Redundant Parallel File Transfer with Anticipative Adjustment Mechanism in Data Grids",Journal of Information Technology and Applications, vol. 1, no. 4, pp. 305-313, March 2007.

[23] Java Cog Kit, http://www.globus.org/cog/java/

[24] Java Cog Kit WiKi, http://wiki.cogkit.org/wiki/Main_Page

[25] Global Grid Forum, http://www.ogf.org/

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

[27] Globus Toolkit 4.2.1 GridFTP User's Guide, http://www.globus.org/toolkit/docs/latest-stable/data/gridftp/user/#gridftp-user-quickstart, June 25, 2009.

[28] Hadoop – Wikipedia, http://en.wikipedia.org/wiki/Hadoop

[29] Java 2 Platform, Enterprise Edition (J2EE) Overview, http://java.sun.com/j2ee/overview.html

[30] Ganglia Monitor System, http://ganglia.sourceforge.net/

[31] Picture Archive and Communication System (PACS) -- Wikipedia, http://en.wikipedia.org/wiki/Picture_archiving_and_communication_system

[32] HPC MIFAS System, http://140.128.102.175/

[33] HPC MIFAS Configuration System, http://140.128.102.175/admin.php

[34] Log4J, http://logging.apache.org/log4j/1.2/