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

利用虛擬化容錯技術建置雲端醫療影像檔案存取系統

Implementation of a Medical Image File Accessing System with Virtualization Fault Tolerance on Cloud

研究生：郭政達

中華民國 一百年七月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 郭 政 達 所提之論文

利用虛擬化容錯技術建置雲端醫療影像檔案存取系統

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 許慶賢 簽章

委 員 李冠憬

朱正忠

賴冠州

指 導 教 授 楊朝棟 簽章

中華民國 100 年 6 月 28 日

摘要

近年來，雲端運算與虛擬化已經成為計算機科學中最熱門的領域。雲端運算透過現有網路技術、平行技術及分散式技術，將分散的電腦組成為一個能提供超強功能的叢集，為網際網路使用者提供運算、儲存及軟硬體等服務。本機電腦可以不用再像傳統電腦那樣需要空間足夠的硬碟、大功率的處理器和大容量的記憶體，只需要一些必要的硬體裝置如網路裝置和基本的輸入輸出等設備。使用者也不需要瞭解伺服器在哪裡，不用關心內部如何運作，透過網際網路就可以透明地使用各種資源。伺服器虛擬化可以提昇資源的利用率、提高部署的靈活性、協助精簡管理的人力及節能省碳等好處，透過虛擬化管理工具，就能更有彈性管理應用程式資源的配置。本篇論文專注於將雲端儲存叢集融入虛擬化的技術做到高可用性的服務。我們實作出一個醫療影像系統，並且將系統儲存服務架構在分散式檔案系統上。在名稱節點的部份以虛擬機器做建置，透過分散式副本區塊配置就能將指定的網絡設備做一整個磁碟的鏡像。配合心跳技術控管，可做到虛擬機器高可用性的即時同步與故障轉移備援切換。最後，透過實驗也證明了資料儲存叢集高可靠性及容錯的能力。

關鍵字：數位影像儲存通訊系統、醫療數位影像傳輸協定、雲端計算、分散式檔案系統、虛擬化、虛擬化管理、虛擬化容錯、高可用性。

Abstract

In recent years, Cloud Computing and Virtualization has become the most popular computer science field. Cloud Computing through the existing network, parallel and distributed technology, distributed computer composed of a cluster can provide powerful features for Internet users to provide computing, storage and software and hardware services. The local computer may not be as traditional as the computer needs enough hard disk space, high-power processors and high-capacity memory, only the necessary hardware devices such as network devices and basic input and output devices. Users will not need to know where the server, don't care how it works internally, through the Internet can be transparent use of resources. Server virtualization can improve resource utilization, improve deployment flexibility and help streamline the management of manpower and energy-saving and carbon benefits through virtualization management tools, management applications can be more flexible allocation of resources. This paper focuses on the cloud storage virtualization technology into the cluster to achieve high-availability services. We really make a medical imaging system and the system architecture in a distributed file system. Namenode using the virtual machine to do a build, through Distributed Replicated Block Device can be specified by the network equipment to do an entire disk image. Finally, through experiments also proved the high reliability data storage clustering and fault tolerance capabilities.

Keywords: Picture Archiving and Communications System (PACS), Digital Imaging and Communications in Medicine (DICOM), Cloud Computing, Distributed File System (DFS), Virtualization, Virtualization Management, Virtualization Fault Tolerence(VFT), High Availability (HA)

Acknowledgements

在資工所兩年的學習過程，隨著論文的付梓，即將劃上句點，這段時間以來的點點滴滴，有回憶，有不捨。不論是課業上的學習、參加校外比賽或論文的寫作及投稿等，的確都花了大家蠻多的心思，不過讓我們從中獲益匪淺。

本論文能順利完成，幸蒙楊朝棟博士的指導與教誨，對於研究的方向、觀念的啟迪、架構的匡正以及資料的提供，於此獻上最深的敬意與謝意。論文口試期間，承蒙口試委員朱正忠院長、許慶賢教授、李冠憬教授及賴冠州教授的鼓勵與疏漏處之指正，使得本論文更加完備，在此謹深致謝忱。

在來要感謝研究所合作的學長、同學們，謝謝龍騰、威利、翔耀、冠傑及本加，週末常常出來一起做實驗及撰寫論文，才得以順利完成。也要感謝智霖、柏翰在這段期間提供支持與協助，沒有你們無私的付出，我將不可能完成我的研究工作。

最後感謝我的家人，沒有你們的支持與體諒，我將無法順利完成論文，更遑論畢業取得碩士學位，對於所有幫助過我、關懷過我的人，致上由衷感謝。

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

More and more long-term costs control an onsite medical imaging archive. In 2005, the typical U.S. provider estimated that 50 MB of storage were required for an average study. According to the U.S. diagnostic imaging archive storage demand forecasts (Figure 1-1), it attempting to arrive at an estimate of the total storage volume required to store medical image data. The first assumption that one has to make is to consider only the primary copies of images, that is, to assume that there is no data duplication taking place for the sake of redundancy (e.g. RAID) or disaster recovery planning [1-4].

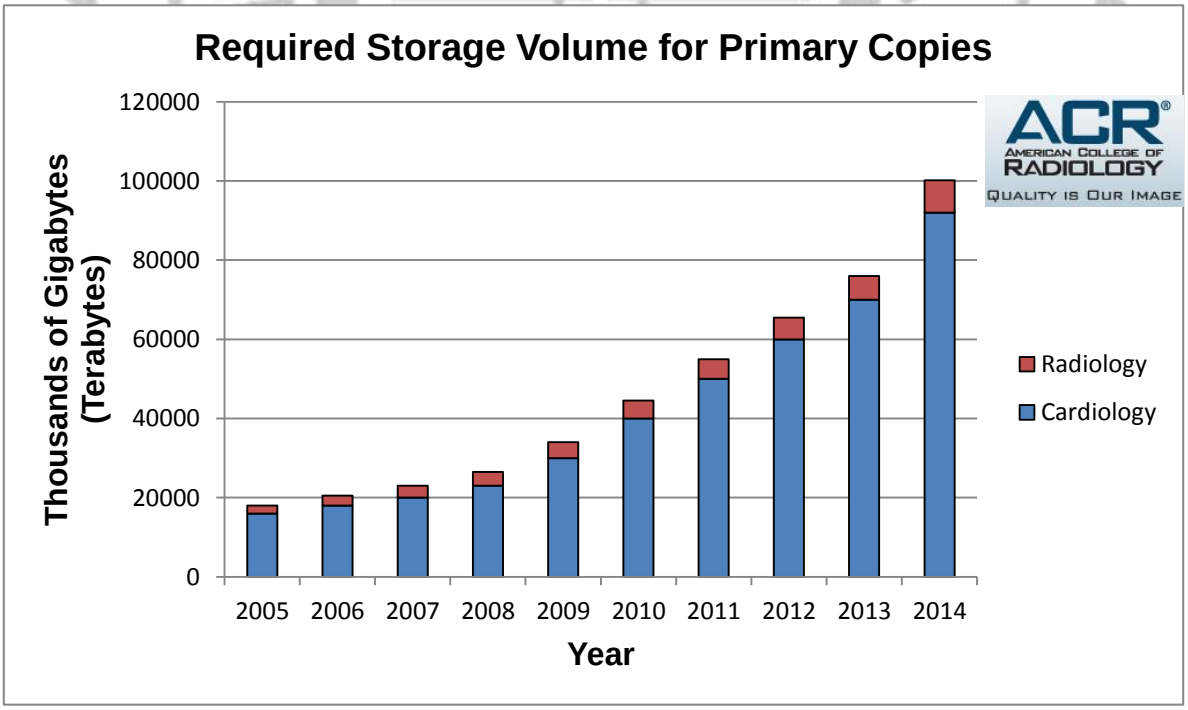


Figure 1-1. U.S. Diagnostic Imaging Archive Storage Requirement Forecast

It is worth noting that while storage cost per Terabyte is declining, the overall cost to manage storage is growing. The common misperception is that storage is inexpensive, but the reality is that storage volumes continue to grow faster than hardware price declines. Using of cloud computing promises to reduce costs, high scalability, availability and disaster recoverability, can we solve the long-term face the problem of medical image archive[5-9].

Build cloud computing in this large-scale parallel computing cluster is growing with thousands of processors [3-5, 9-14]. In such a large number of compute nodes, faults are becoming common place. Current virtualization fault tolerance response plan, focusing on recovery failure, usually relies on a checkpoint/restart mechanism. However, in today's systems, node failures can often be predicted by detecting the deterioration of health conditions [15-25].

For over a decade, the majority of all hospital and private radiology practices have transformed from film-based image management systems to a fully digital (filmless and paperless) environment but subtly dissimilar (in concept only) to convert from a paper medical chart to an HER. Film and film libraries have given ways to modern picture archiving and communication systems (PACS). And they offer highly redundant archives that tightly integrate with historical patient metadata derived from the radiology information system. These systems may be not only more efficient than film and paper but also more secure as they incorporate with safeguards to limit access and sophisticate auditing systems to track the scanned data. However, although radiologists are in favor of efficient access to the comprehensive imaging records of our patients within our facilities, we ostensibly have no reliable methods to discover or obtain access to similar records which might be stored elsewhere [6-8].

According to our research, there were few Medical Image implementations on cloud environment. However, a familiar research presented the benefits of Medical Images on cloud

were: Scalability, Cost effective and Replication [9]. In the same study, they also presented a HPACS system but lacked of management interface.

1.2 Contribution

We build a HDFS as a platform for Medical Image File Access System (MIFAS), and to do fault-tolerant for HDFS. Instead of a reactive scheme for Virtualization Fault Tolerance (VFT), we are promoting a one where processes automatically migrate from “unhealthy” nodes to healthy ones. Our approach relies on operating system virtualization techniques exemplified by Xen. It leverages virtualization techniques combined with health monitoring and load-based migration. We exploit Xen’s live migration mechanism for a guest operating system (OS) to migrate a Namenode from a health-deteriorating node to a healthy one without stopping the Namenode task during most of the migration.

Our FT daemon orchestrates the tasks of health monitoring, load determination and initiation of guest OS migration. The results showed that the actual cost of relocation hidden cost of live migration has been seamless transfer can be done. Furthermore, migration overhead is shown to be independent of the number of nodes in our experiments indicating the potential for scalability of our approach. Overall, our enhancements make FT a valuable asset for long running HDFS task, particularly as a complementary scheme to reactive FT using full checkpoint/restart schemes. In the context of OS virtualization, we believe that this is the first comprehensive study of fault tolerance where live migration is actually triggered by health monitoring.

1.3 Thesis Organization

This paper is organized as follows. First, in section 2 we introduce the background and related works. Section 3 describes the implementation details of system architecture. In section 4 we show a result of the experiment, and finally section 5 outlines the main conclusions and the future work.

Chapter 2
Background

2.1 Medical Technology

To be able to define why we need the cloud in medicine and health care? What are the benefits of using it? And how we can use it? We need first to define the key characteristics of all categories.

2.1.1 PACS

PACS is an acronym that stands for **P**icture **A**rchiving and **C**ommunication **S**ystem. It revolutionized the field of radiology, which now consists of all digital, computer-generated images as opposed to the analog film of yesteryear. Analog film took up space and time for filing and retrieval and storage, and was prone to being lost or misfiled. PACS therefore saves precious time and money, and reduces the liability caused by filing errors and lost films. A PACS consists of four major components: the imaging modalities such as CT and MRI, a secured network for the transmission of patient information, workstations for interpreting and reviewing images, and archives for the storage and retrieval of images and reports. Combined with available and emerging Web technology, PACS has the ability to deliver timely and efficient access to images, interpretations, and related data. PACS breaks down the physical and time barriers associated with traditional film-based image retrieval, distribution, and display. PACS is primarily responsible for the inception of virtual radiology, as images can now be viewed from across town, or even from around the world. Additionally, PACS acts as

a digital filing system to store patients' images in an organized way which enables records to be retrieved with ease as needed for future reference.

2.1.2 DICOM vs. Nanodicom

What is DICOM? Short for Digital Imaging and Communications in Medicine, a standard in the field of medical informatics for exchanging digital information between medical imaging equipment (such as radiological imaging) and other systems, ensuring interoperability. The standard specifies: a set of protocols for devices communicating over a network the syntax and semantics of commands and associated information that can be exchanged using these protocols a set of media storage services and devices claiming conformance to the standard, as well as a file format and a medical directory structure to facilitate access to the images and related information stored on media that share information. The standard was developed jointly by ACR (the American College of Radiology) and NEMA (the National Electrical Manufacturers Association) as an extension to an earlier standard for exchanging medical imaging data that did not include provisions for networking or offline media formats [26].

DICOM enables the integration of scanners, servers, workstations, printers, and network hardware from multiple manufacturers into a picture archiving and communication system (PACS). The different devices come with DICOM conformance statements which clearly state the DICOM classes they support. DICOM has been widely adopted by hospitals and is making inroads in smaller applications like dentists' and doctors' offices.

DICOM differs from some, but not all, data formats in that it groups information into data sets. That means that a file of a chest X-Ray image, for example, actually contains the patient ID within the file, so that the image can never be separated from this information by mistake. This is similar to the way that image formats such as JPEG can also have embedded

tags to identify and otherwise describe the image.

A DICOM data object consists of a number of attributes, including items such as name, ID, etc., and also one special attribute containing the image pixel data (i.e. logically, the main object has no "header" as such - merely a list of attributes, including the pixel data). A single DICOM object can only contain one attribute containing pixel data. For many modalities, this corresponds to a single image. But note that the attribute may contain multiple "frames", allowing storage of cine loops or other multi-frame data. Another example is NM data, where an NM image by definition is a multi-dimensional multi-frame image. In these cases three-dimensional or four-dimensional data can be encapsulated in a single DICOM object. Pixel data can be compressed using a variety of standards, including JPEG, JPEG Lossless, JPEG 2000, and Run-length encoding (RLE). LZW (zip) compression can be used for the whole data set (not just the pixel data) but this is rarely implemented.

The same basic format is used for all applications, including network and file usage, but when written to a file, usually a true "header" (containing copies of a few key attributes and details of the application which wrote it) is added.

Nanodicom is a DICOM file parser. So far, DICOM dictionary has been updated to 2009. Except for a few cases (multiple and conditional VR), nearly 99% of the label can be resolved. Image resolution can also be part of the identification of a number of formats supported.

2.2 Virtualization

2.2.1 Virtualization Technology

One of great job “Xen” was developed by University of Cambridge Computer Laboratory; as

of 2010 the Xen community develops and maintains Xen as free software, licensed under the GNU General Public License (GPLv2). Virtualization lets you run multiple virtual machines on a single physical machine, with each virtual machine sharing the resources of that one physical computer across multiple environments. Virtualization is simply the logical separation of the request for some service from the physical resources that actually provide that service. In practical terms, virtualization provides the ability to run applications, operating systems, or system services in a logically distinct system environment that is independent of a specific physical computer system. Obviously, all of these have to be running on a certain computer system at any given time, but virtualization provides a level of logical abstraction that liberates applications, system services, and even the operating system that supports them from being tied to a specific piece of hardware. Virtualization, focusing on logical operating environments rather than physical ones, makes applications, services, and instances of an operating system portable across different physical computer systems. Virtualization can execute applications under many operating systems, manage IT more efficiently, and allot resources of computing with other computers [15].

It's not a new technique, IBM had implemented on 360/67 and 370 on 60, 70 eras. Virtualization gets hardware to imitate much hardware via Virtual Machine Monitor, and each one of virtual machines can be seemed as a complete individual unit. For a virtual machine, there are memories, CPUs, unique complete hardware equipment, etc... It can run any operating systems, called Guest OS, and do not affect other virtual machines.

In general, most virtualization strategies fall into one of four major categories [16]:

Full Virtualization: Also sometimes called hardware emulation. In this case an unmodified operating system is run using a hypervisor to trap and safely translate/execute privileged instructions on-the-fly. Because trapping the privileged instructions can lead to significant performance penalties, novel strategies are used to aggregate multiple instructions and

translate them together. Other enhancements, such as binary translation, can further improve performance by reducing the need to translate these instructions in the future.

Para-virtualization: Like full virtualization, para-virtualization also uses a hypervisor, and also uses the term virtual machine to refer to its virtualized operating systems. However, unlike full virtualization, para-virtualization requires changes to the virtualized operating system. This allows the VM to coordinate with the hypervisor, reducing the use of the privileged instructions that are typically responsible for the major performance penalties in full virtualization. The advantage is that para-virtualized virtual machines typically outperform fully virtualized virtual machines. The disadvantage, however, is the need to modify the para-virtualized virtual machine/operating system to be hypervisor-aware. This has implications for operating systems without available source code.

Operating System-level Virtualization: The most intrusive form of virtualization is operating system level virtualization. Unlike both para-virtualization and full virtualization, operating system-level virtualization does not rely on a hypervisor. Instead, the operating system is modified to securely isolate multiple instances of an operating system within a single host machine. The guest operating system instances are often referred to as virtual private servers (VPS). The advantage to operating system-level virtualization lies mainly in performance. No hypervisor/instruction trapping is necessary. This typically results in system performance of near-native speeds. The primary disadvantage is that all VPS instances share a single kernel. Thus, if the kernel crashes or is compromised, all VPS instances are compromised. However, the advantage to having a single kernel instance is that fewer resources are consumed due to the operating system overhead of multiple kernels.

Native Virtualization: Native virtualization leverages hardware support for virtualization within a processor itself to aid in the virtualization effort. It allows multiple unmodified operating systems to run alongside one another, provided that all operating systems are

capable of running on the host processor directly. That is, native virtualization does not emulate a processor. This is unlike the full virtualization technique where it is possible to run an operating system on a fictional processor, though typically with poor performance. In x86 64 series processors, both Intel and AMD support virtualization through the Intel-VT and AMD-V virtualization extensions. X86 64 Processors with virtualization support are relatively recent, but are fast becoming widespread.

For the remainder of this paper we choose “Xen Hypervisors” to be our virtualization technology platform. The reason of this choose which is we have to combine with the virtualization management tool, and its limitation is must using “Xen” to be the platform.

As a result of the widespread of virtual machine software in recently years, two best x86 CPU manufacturers Intel/AMD, with efficiency of x86 computers and increasing of compute core of CPU, both have published the new integrated virtualization on CPU, one for Intel Vander pool and another for AMD Pacifica. These technologies also support Xen, and make efficiency step up more than initial stages [27].

2.2.2 Virtualization Fault Tolerance

Virtualization Fault Tolerance (VFT) is a pioneering component, which provides continuous availability to applications, preventing downtime and data loss in the event of server failures. It provides operational continuity and high levels of uptime in hypervisor environments, with simplicity and at a low cost [17-19, 28, 29].

Fault tolerant virtual machines and auxiliary copy is not allowed to run on the same host. The two virtual machines constantly heartbeat against each other and if either virtual machine instance loses the heartbeat, the other takes over immediately. The heartbeats are very frequent, with millisecond intervals, making the failover instantaneous with no loss of data or

state. Fault tolerance using anti-affinity rules to ensure that two instances of fault-tolerant virtual machines will never be the same host. This ensures that the host failure cannot result in loss of two virtual machines.

Tolerance to avoid "split brain" situation occurs, this situation may lead to the virtual machine to recover from failure after the two active copies. Shared memory locks on the atomic coordinate file for failover, so that only one end can be used as the primary virtual machine continues to run by the system automatically re-generate a new auxiliary virtual machine [20].

2.2.3 Virtual Machine Management

A key component in this scenario is the virtual machine (VM) management system. A VM manager provides a centralized platform for efficient and automatic deployment, control, and monitoring of VMs on a distributed pool of physical resources. Usually, these VM managers also offer high availability capabilities and scheduling policies. The OpenNebula is a virtual infrastructure engine that enables the dynamic deployment and re-allocation of virtual machines in a pool of physical resources. The OpenNebula system extends the benefits of virtualization platforms from a single physical resource to a pool of resources, decoupling the server, not only from the physical infrastructure but also from the physical location [30].

The OpenNebula contains one frontend and multiple backend. The front-end provides users with access interfaces and management functions. The back-ends are installed on Xen servers, where Xen hypervisors are started and virtual machines could be backed. Communications between frontend and backend employ SSH. The OpenNebula gives users a single access point to deploy virtual machines on a locally distributed infrastructure.

OpenNebula orchestrates storage, network, virtualization, monitoring, and security

technologies to enable the dynamic placement of multi-tier services (groups of interconnected virtual machines) on distributed infrastructures, combining both data center resources and remote cloud resources, according to allocation policies [30]. In Figure 2-1, the OpenNebula internal architecture can be divided into three layers.

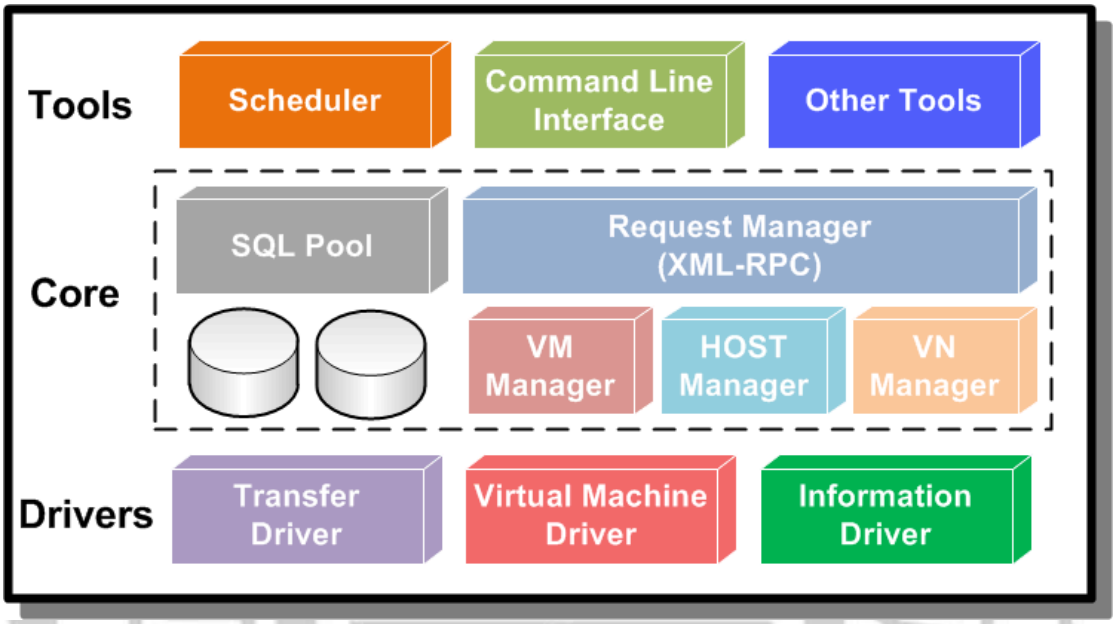


Figure 2-1. OpenNebula Internal Architecture

Tool: management tools developed using the interfaces provided by the OpenNebula Core.

Core: the main virtual machine, storage, virtual network and host management components.

Drivers: it is to plug-in different virtualization, storage and monitoring technologies and Cloud services into the core.

However the OpenNebula lack a GUI management tool. In pervious works we build virtual machines on OpenNebula and implemented Web-based management tool. Thus, the system administrator can be easy to monitor and manage the entire OpenNebula System on our project. OpenNebula is composed of three main components: (1) the OpenNebula Core is a centralized component that manages the life cycle of a VM by performing basic VM operations, and also provides a basic management and monitoring interface for the physical hosts (2) the Capacity Manager governs the functionality provided by the OpenNebula core.

The capacity manager adjusts the placement of VMs based on a set of pre-defined policies (3) Virtualize Access Drivers. In order to provide an abstraction of the underlying virtualization layer, OpenNebula uses pluggable drivers that expose the basic functionality of the hypervisor [21].

2.3 Cloud

Cloud is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics (On-demand self-service, Broad network access, Resource pooling, Rapid elasticity, Measured Service); three service models (Cloud Software as a Service (SaaS), Cloud Platform as a Service (PaaS), Cloud Infrastructure as a Service (IaaS)); and, four deployment models (Private cloud, Community cloud, Public cloud, Hybrid cloud). Key enabling technologies include: (1) fast wide-area networks, (2) powerful, inexpensive server computers, and (3) high-performance virtualization for commodity hardware [10].

2.3.1 Hadoop and HDFS

Hadoop is one of the most salient pieces of the data mining renaissance which offers the ability to tackle large data sets in ways that weren't previously possible due to time and cost constraints. It is a part of the apache software foundation and its being built by the community of contributor in all over the world. The Hadoop project promotes the development of open source software and supplies a framework for the development of highly scalable distributed

computing applications [11].

Hadoop is a top-level Apache project being built and used by a global community of contributors, using the Java programming language [31]. It provides a framework for developing highly scalable distributed applications. The developer just focuses on applying logic instead of processing detail of data sets. The HDFS 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 Datanodes, each of which serves up blocks of data over the network using a block protocol. They also serve the data over HTTP, allowing access to all content from a web browser or other client. Datanodes can connect 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 Namenode. This is a single point of failure for an HDFS installation. If the Namenode goes down, the file system will be off-lined. When it comes back up, the Namenode must replay all outstanding operations. This replay process can take over half an hour for a big cluster [12].

2.4 Distributed Replicated Block Device

Distributed Replicated Block Device (DRBD) refers to block devices designed as a building block to form high availability clusters. It is done by mirroring a whole block device via an assigned network. DRBD can be understood as network based raid-1. [32].

In Figure 2-2 is the DRBD of the workflow, the two block represent two servers that form an HA cluster. The boxes contain the usual components of a Linux kernel: file system, buffer cache, disk scheduler, disk drivers, TCP/IP stack and network interface card (NIC) driver. The black arrows illustrate the flow of data between these components. The red arrows show the flow of data, as DRBD mirrors the data of a highly available service from the active

node of the HA cluster to the standby node of the HA cluster.

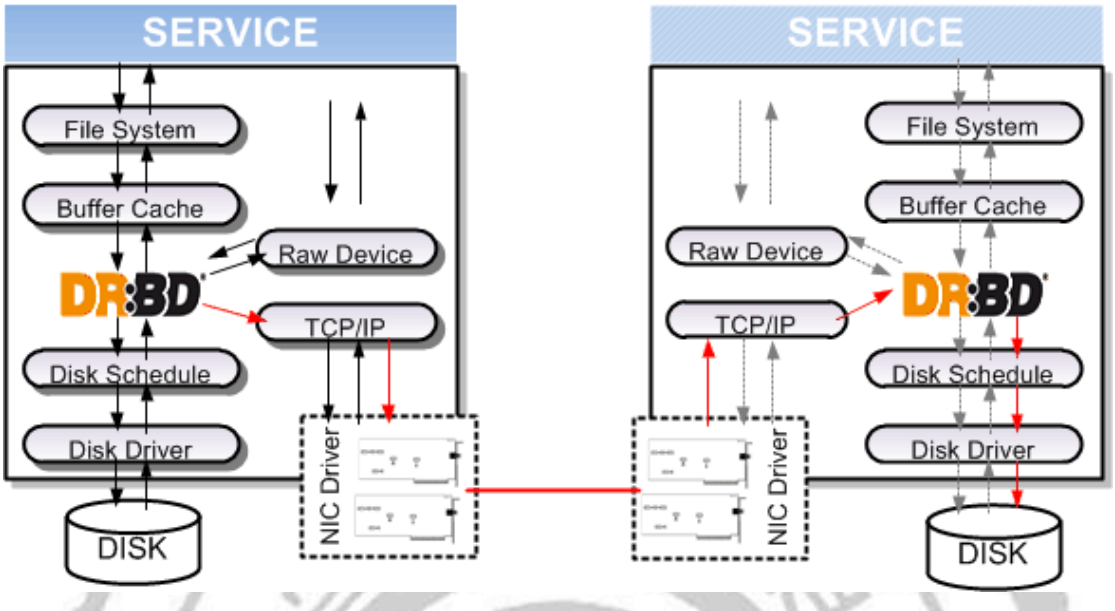


Figure 2-2. DRBD System Workflow

In Figure 2-3 shows a cluster where the left node is currently active, i.e., the service's IP address that the client machines are talking to is currently on the left node. The service including its IP address, which can be migrated to the other node at any time, either due to a failure of the active node or as an administrative action. The lower part of the illustration shows a degraded cluster. In HA speak the migration of a service is called failover, the reverse process is called failback and when the migration is triggered by an administrator it is called switchover [33].

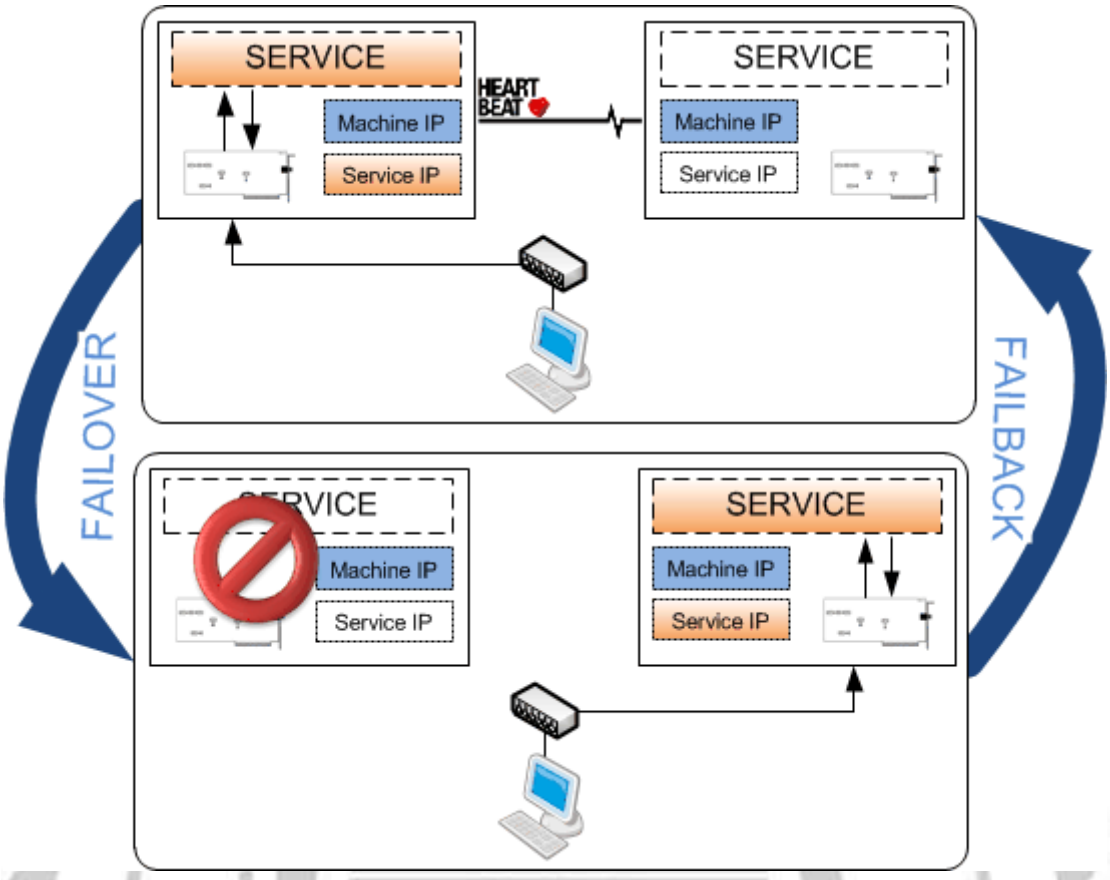


Figure 2-3. DRBD with Heartbeat

2.5 Related Works

HDFS servers (i.e., Datanodes) and traditional streaming media servers are both used to support client applications that have access patterns characterized by long sequential reads and writes. As such, both systems are architected to favor high storage bandwidth over low access latency [34].

Recently “Cloud” became a hot word in this field. S. Sagayaraj [9] proposes that Apache Hadoop is a framework for running applications on large clusters built of commodity hardware. The Hadoop framework transparently provides both reliability and data motion. Hadoop implements a computational paradigm named Map/Reduce, where the application is divided into many small fragments of work. Each fragment of work may be executed or

re-executed on any node in the cluster. So, by just replacing the PACS Server with Hadoop Framework can lead to good, scalable and cost effective tool for the Imaging solution for Health Care System. In the same study, they also presented a HPACS system but lacked of management interface.

It is complex for kind of performance issues, but J. Shafer, et al. [13] proposed "The Hadoop distributed file system: Balancing portability and performance" had a good view in this field. The poor performance of HDFS can be attributed to challenges in maintaining portability, including disk scheduling under concurrent workloads, file system allocation, and file system page cache overhead. HDFS performance under concurrent workloads can be significantly improved through the use of application-level I/O scheduling while preserving portability. Further improvements by reducing fragmentation and cache overhead are also possible, at the expense of reducing portability. However, maintaining Hadoop portability whenever possible will simplify development and benefit users by reducing installation complexity, thus encouraging the spread of this parallel computing paradigm.

In our previous reproaches [35-40] use co-allocation to solve the data transfer problem in grid environment. It is the foundation of this paper. But it is for grid not implement on cloud, so in this paper we have a significant change from previous works, because we implement it on the cloud environment and use virtualization fault tolerance techniques.

Chapter 3

System Design and Implementation

In this section, describes a big significant improvement than previous MIFAS works. We use virtualization technology to build our system. And also enhance HA, for more detail please see below explanation.

3.1 Virtualization Fault Tolerance Design

Hadoop infrastructure has become a critical part of day-to-day business operations. As such, it was important for us to find a way to resolve the single-point-of-failure issue that surrounds the master node processes, namely the Namenode and JobTracker. While it was easy for us to follow the best practice of offloading the secondary Namenode data to an NFS mount to protect metadata, ensuring that the processes were constantly available for job execution and data retrieval were of greater importance. We've leveraged some existing, well tested components that are available and commonly used in Linux systems today. Our solution primarily makes use of DRBD from LINBIT and Heartbeat from the Linux High Availability (HA) project which we called Virtualization Fault Tolerance (VFT). The natural combination of these two projects provides us with a reliable and highly available solution, which addresses limitations that currently exist. In this context, virtualization is being used as a solution not only to provide service flexibility, but also to consolidate server workloads and improve server utilization. A virtualized based system can be dynamically adapted to the client demands by deploying new virtual nodes when the demand increases, and powering off and consolidating virtual nodes during periods of low demand.

The Virtualization Fault Tolerance (VFT) has three main phases: virtual machines

migration policy, information gathering, and keep services always Available. The work flow can be described as follow illustration (Figure 3-1). But, there is a constraint of this methodology which is the physical host number must bigger than three, it is the base requirement to achieve VFT methodology. The coming section will explain this reason.

Virtual Machines Migration Policy: it stands for enable Dynamic Resource Allocation (DRA) to ensure entire distribution virtualization cluster under a best performance.

Information Gathering: this phase is presented that we have a detection mechanism to retrieve all Hosts and check Hosts is alive or not. In our paper we use Internet Control Message Protocol (ICMP) to collect information every 5 minutes.

Keep Service Always Available: assume VM m is under DRBD with Heartbeat mechanism and Host n is unavailable physical machine. Once the Host n is shutting down, if VM m is secondary node, then it will move to on-line Host and boot automatically. If VM m is primary node then secondary node will replace the VM m to primary node immediately. Next pre-primary node will boot on on-line host and become secondary.

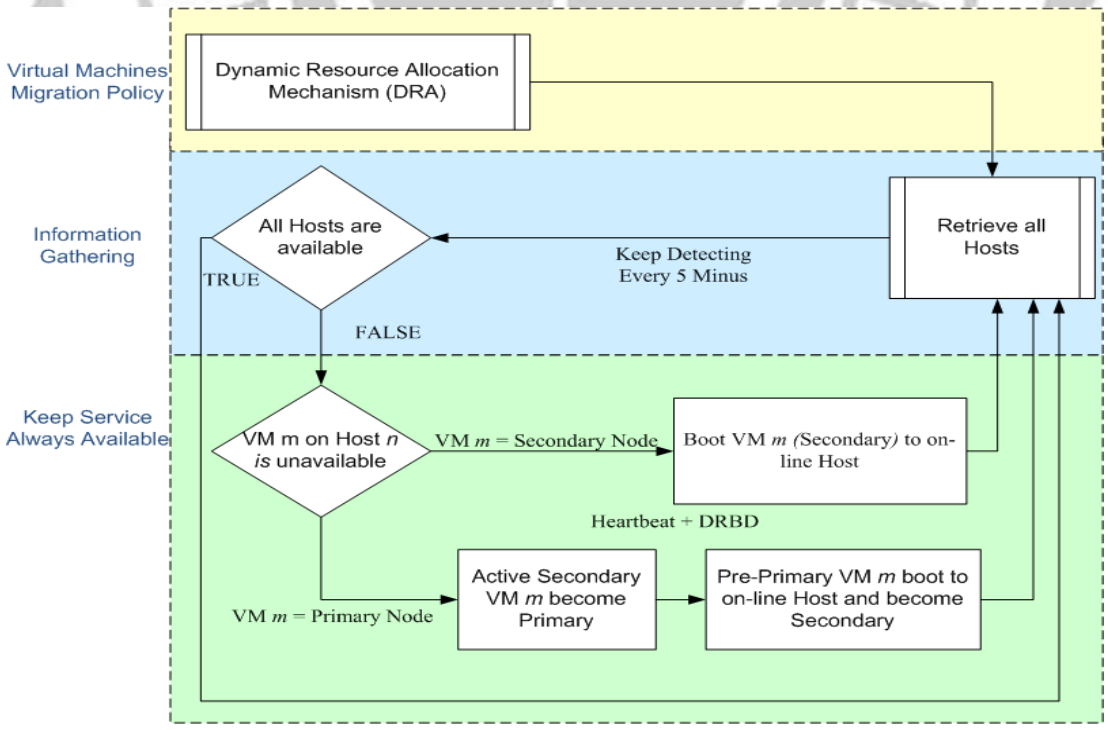


Figure 3-1. Virtualization Fault Tolerance Flow

In our paper, the HA components can be shown as Figure 3-2. Follow bottom to top, the infrastructure, Hosts, means physical machines. And Xen Hypervisor is one of virtualization technology suit for Linux series OSs. And follow up is 2 VMs, VM2 is primary node, and VM1 is secondary node. Assume Hadoop Namenode is built on VM2 and VM1, and then VM1 is the backup of VM2. Under a DRBD with Heartbeat mechanism, you can see we use 5 IP to deploy this system, one pair is for Cross Over and the other pair is for identifying the primary and secondary, the last one is for the service usage. Finally, a key component OpenNebula is on the top layer, it is the key of entire scenario, this component provides a centralized platform for efficient and automatic deployment, control, and monitoring of VMs on a distributed pool of physical hosts. And we also compose DRA and web interface management tool components to combine with OpenNebula component.

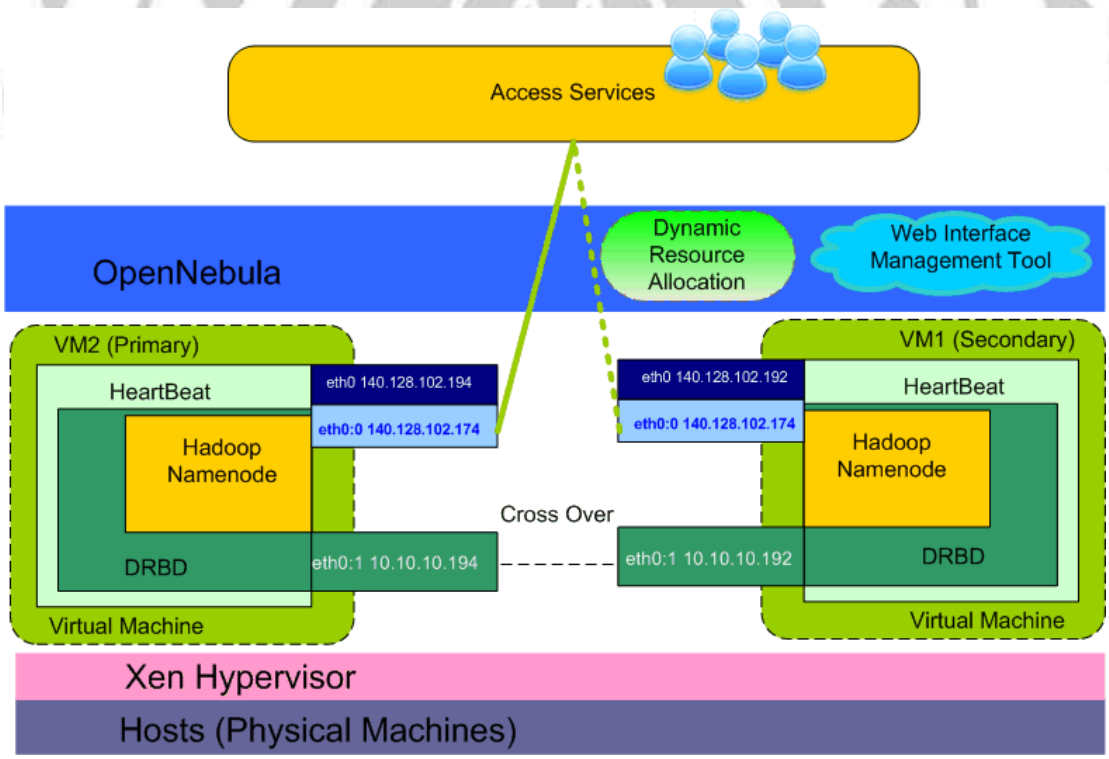


Figure 3-2. HA Components

Our approach in HA speak can be described as Figure 3-3 First one Host A was shutting down by unexpected matter meanwhile it also trigger VFT, next the secondary node VM 2

became primary and handover all services from pre-primary, that we called FAILOVER. Finally, VM 1 booted on Host C automatically and became secondary node, that we called FAILBACK.

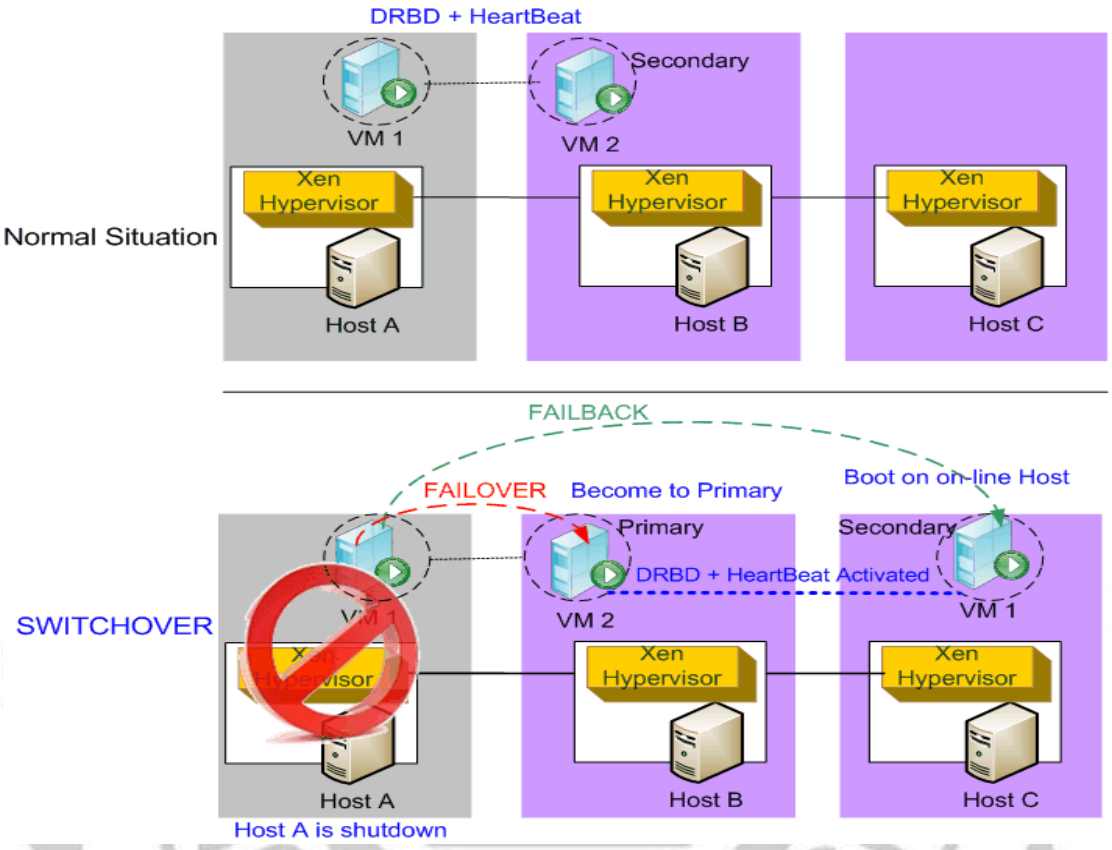


Figure 3-3. How to Trigger VFT

3.2 System Architecture

MIFAS was developed on cloud environment, Detailed System Components, such as Figure 3-4. The distribution file system was built on HDFS of Hadoop environment. This Hadoop platform could be described as PaaS (Platform as a Service). We extended a SaaS (Software as Service) based on PaaS. As the shown illustration, the top level of MIFAS was web-based interface. And now we do more things between Pass and IaaS, in order to achieve the Hadoop HA.

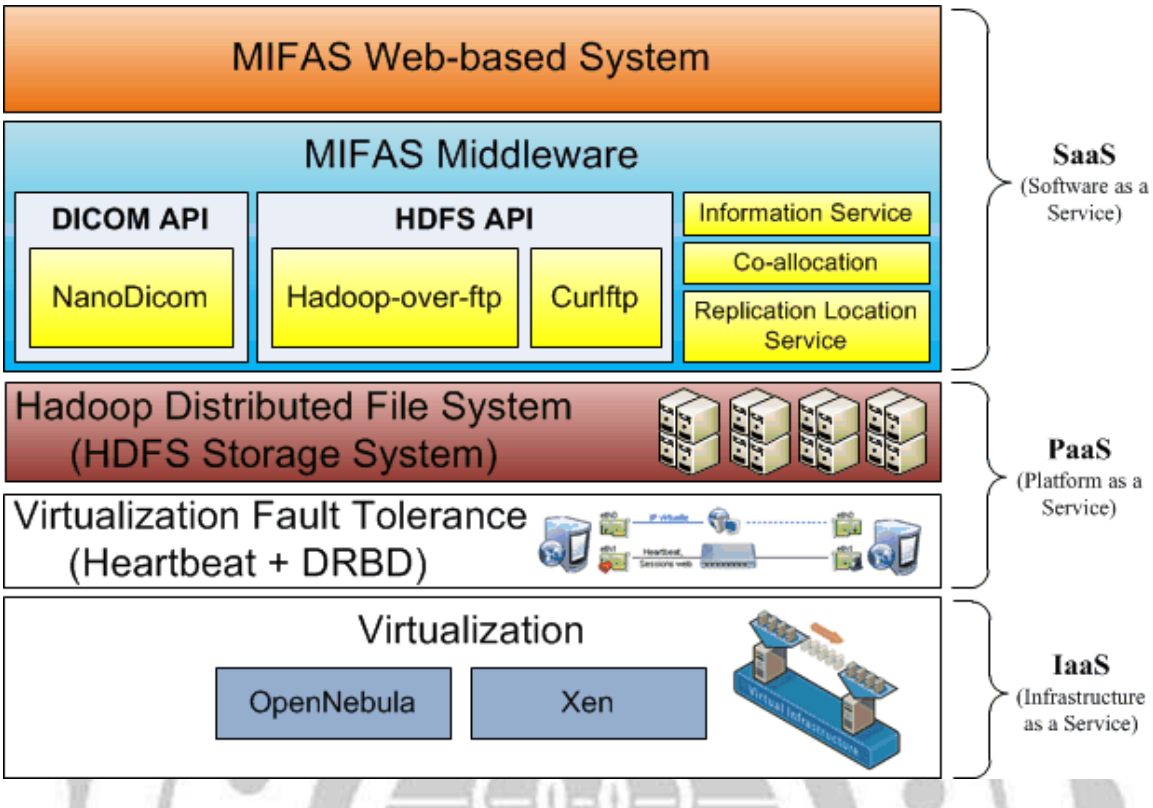


Figure 3-4. System Components

IaaS: in our previous work, we used OpenNebula to manage our VMs [22]. In order to achieve our goal, we migrated our servers into OpenNebula environment. And it is also a key feature to develop the VFT approach on virtualization.

PaaS: HDFS was a well-known could platform in this field, so we will save the introduction of HDFS in this section. But, as we mentioned the VFT approach in section 3.1, the Hadoop Namenode was also under HA control by our VFT approach. Figure 3-5 can be described as our configuration.

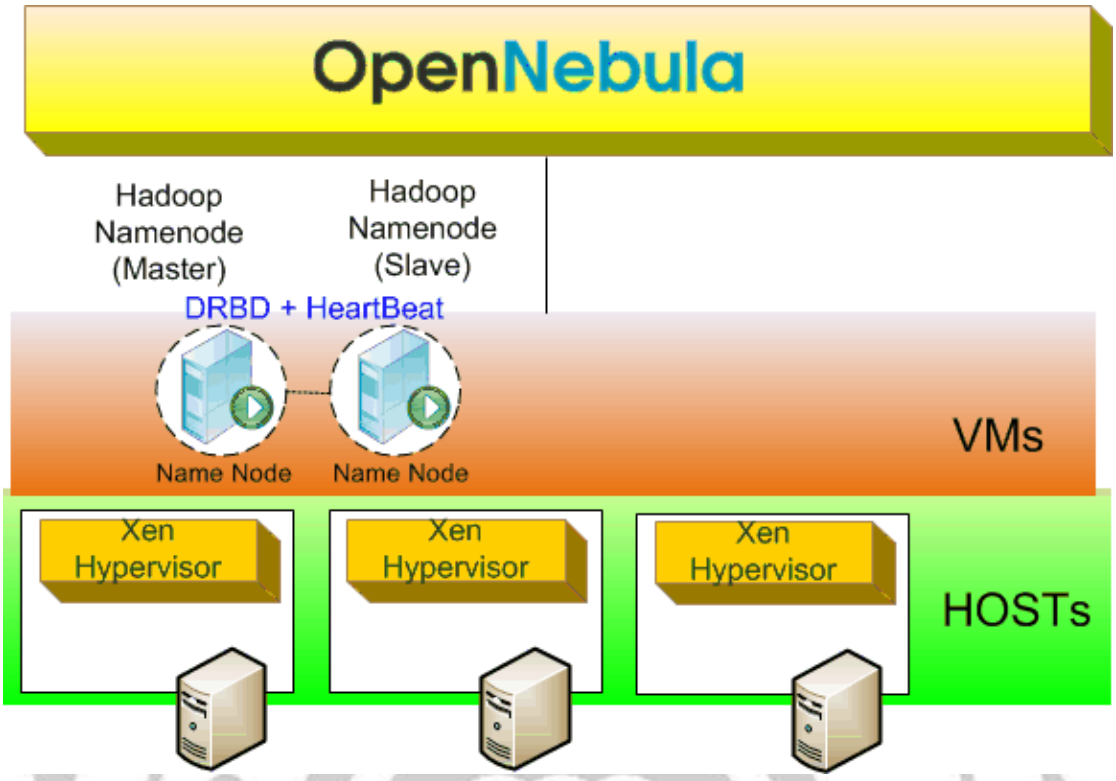


Figure 3-5. Deploy Namenode HA on Virtualization

SaaS: in this layer we called **Middleware.** It is the core of MIFAS. From the top level a web-based system was provided a GUI interface that users or administrators could manage patient's data on it also including the quick view of medical images. **Nanodicom** [41] was a component that made by PHP, it could convert DICOM file into JPEG without complex process. We applied it while uploading DICOM files, therefore the system would convert it into JPEG format automatically (Figure 3-6). In another hand, consider to the gap between Hadoop and general platform, we provided a good solution for it which is **Hadoop-over-ftp** [42] and **Curlftp** [43]. Both of these components are open source based software. The Hadoop-over-ftp could convert HDFS as FTP service, and then Curlftp could mount the FTP service as a local level storage. Thus, we could provide kind of solution for the enterprise or those people whom are not familiar with Hadoop as a remote storage. In generally, we split entire system into two different levels, one is the DICOM viewer and patient management, and the other is storage level.

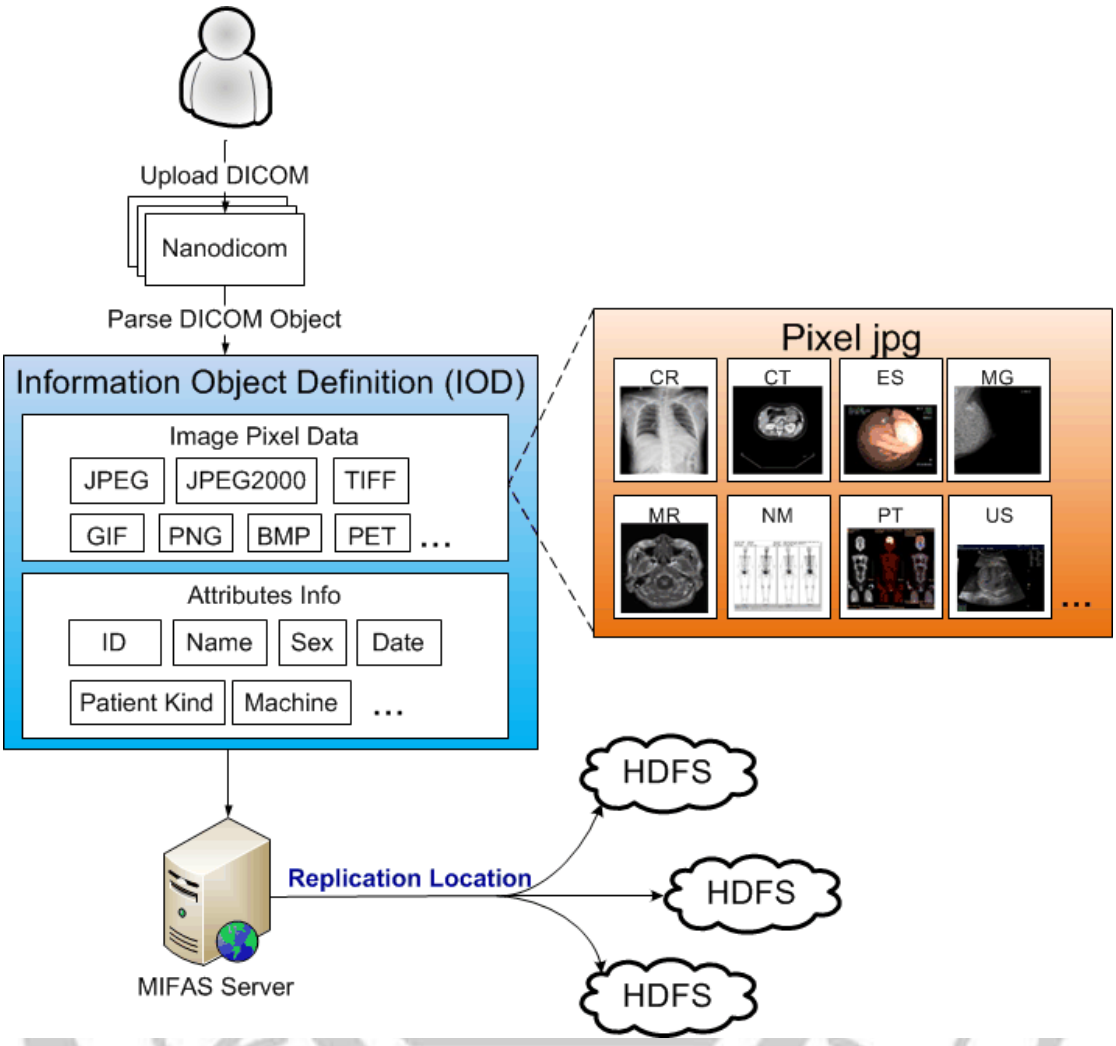


Figure 3-6. Compress DICOM to JPEG

This Middleware also collected necessary information such as bandwidth between server and server, the server utilization rate, and network efficiency. The information provided entirety MIFAS Co-allocation Distribution Files System to determine the best solution of downloading allocation jobs.

Information Service: To obtained analysis in the status of host. The Middleware of MIFAS had a mechanism to fetch the information of hosts called Information Service. In this research, we installed the **Ganglia** [44] in each member of Hadoop node to get the real-time state from all members. Therefore, we could get the best strategy of transmission data from Information Service which is one of the components of MIFAS Middleware.

Co-allocation: As our researched into section Ⅱ.C, Co-allocation mechanism could conquest the parallel downloading from Datanodes. Besides, it also sped up downloading and solved network faults problems. Due to user using MIFAS to access Medical Images, the co-allocation will be enabled automatically. In order to reached parallel downloading approaches, the system will split those file in to different parts and obtain data from different Cloud depend on Cloud health status. Therefore, we can get the best downloading strategy. In our earlier research [35-37] was also provided our co-allocation mechanism.

Replication Location Service: In this research, we built three groups of HDFS in different locations, and each HDFS owned an amount of Datanodes. The Replication Location Service means that the Service would automatically make duplication from private cloud to one another when medical images uploaded to MIFAS.

3.3 System Workflow

In Figure 3-7, it shows our efforts on MIFAS. In this research, we also made a real system to achieve our paper. The system's workflow shows in the shown illustration. Firstly, users input username and password to authenticate. Secondly, users could input search condition to query patients' information. Thirdly, users could also view patients' Medical Images. Fourthly, users can configure in MIFAS. Fifthly, if users can present MIFAS downing mechanism, it means the Middleware is workable in MIFAS.

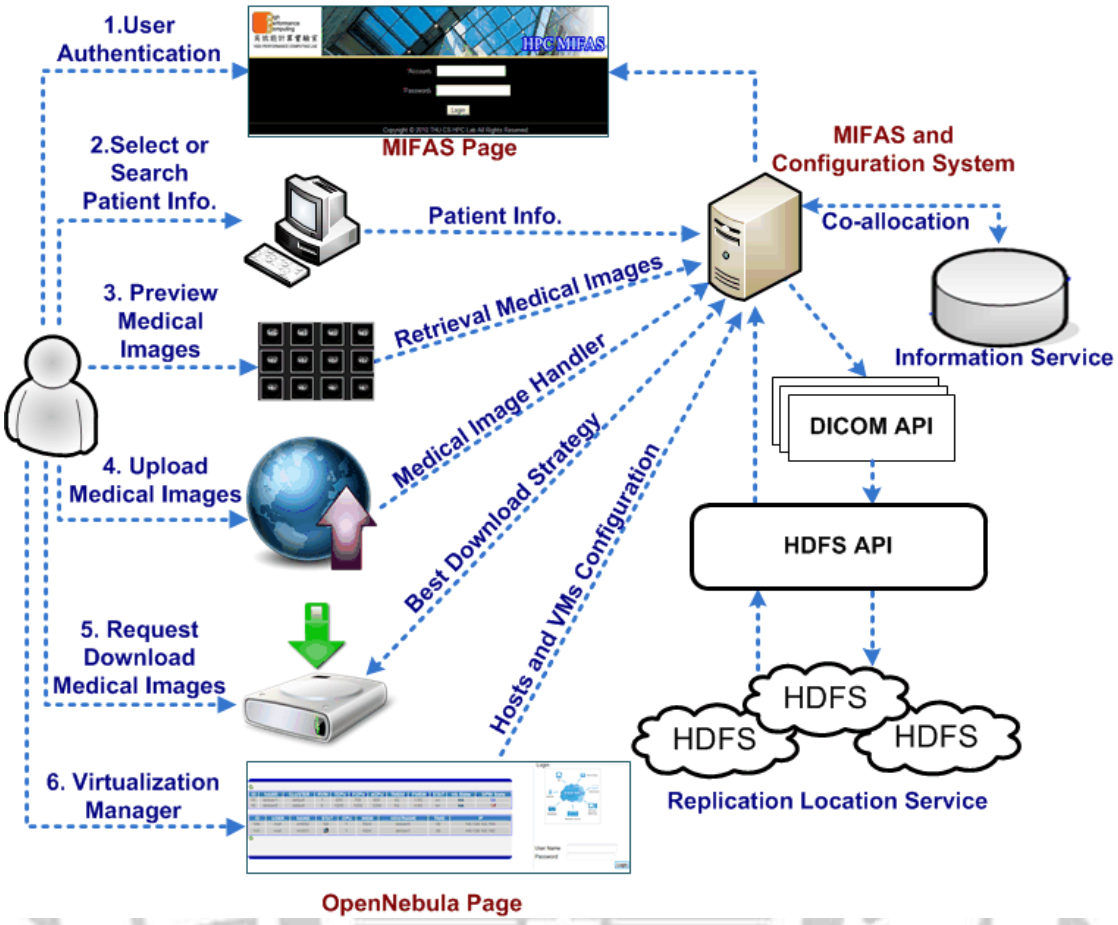


Figure 3-7. MIFAS System Workflow

3.4 MIFAS System Interface

MIFAS offer authentication interface as shown in Figure 3-8, the users have to pass the validation before login to the MIFAS. After passing the validation, you will see the Portal of MIFAS (In Figure 3-9). In this page, there are 3 main functional block, each one has its functional purpose.

Examine Type: it is the catalog of Medical Examination Type. Medical Images could from various medical imaging instruments, including ultrasound (US), magnetic resonance (MR), positron emission tomography (PET), computed tomography (CT), endoscopy (ENDO),

mammograms (MG), direct radiography (DR), computed radiography (CR) etc. The examine type is cataloged depend on the definition of DICOM.

Filter: this block provided search function; users can get patient information through inputted keyword. In web-based interface system, it is easy to reach this goal. User can capture any information that he/she wanted by filter. Thus, like other system on the internet, MIFAS provide multidimensional information for the users. There are four main options in the filter function which are “Chart NO” (Examination No), “Patient Name”, “Start of Examination Date”, and “End of Examination Date”.

Patient Information List: in this block, you will see the detail information according to the search condition of block B and block A. And it is also including several important functions.

Figure 3-8. Authorization Interface

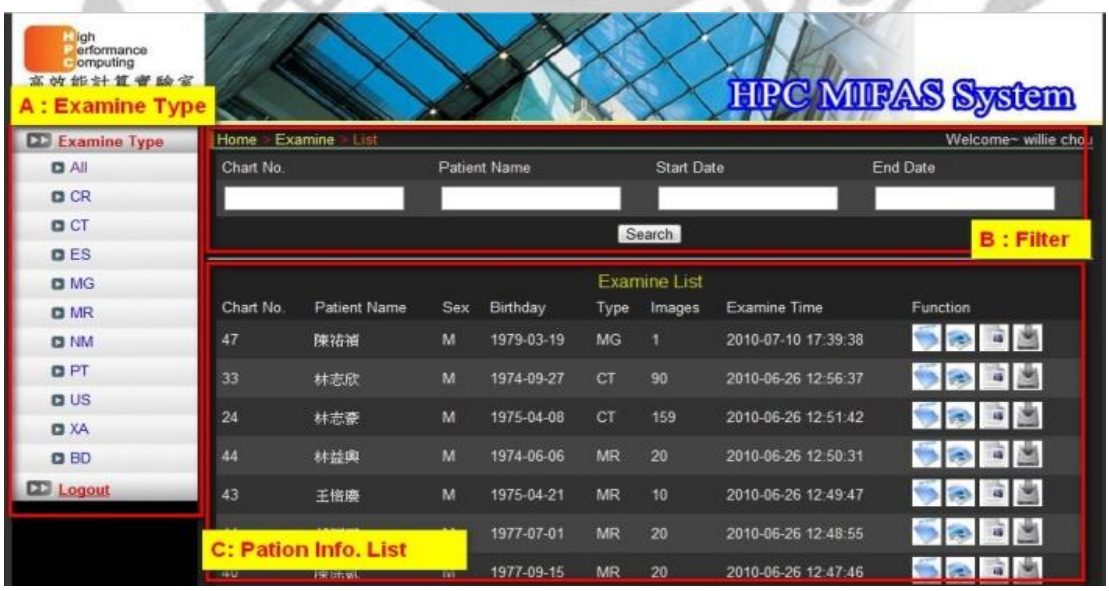


Figure 3-9. Portal of MIFAS System

Function items as shown Figure 3-10, which are the 1st Image Status, 2nd Thumbnail viewer, 3rd PACS Reporting, 4th Download File. Display the information of file distribution status as Figure 3-11 , including photographic description and photographic catalog. Thumbnail viewer function is in Figure 3-12, this function show the thumbnail of Medical Image and examination report. PACS Reporting is as shown Figure 3-13, it is the detail report of patient medical record. For more detail of Medical Images, we can through Download File function then utilize other professional DICOM viewer. Regarding to how to upload Medical Images to MIFAS please see Figure 3-14. According to our paper, the Replication Location Service will duplicate images to each cloud.

Finally the 4th icon in Figure 3-10 could download DICOM format Medical Images from MIFAS. MIFAS will enable co-allocation mechanism to allocate file through a best strategy.

Figure 3-10. Functions

File Status

File Name	Descript	A Server	B Server	C Server
photo_9db96151.jpg	Brain-1	V	V	
photo_8c0c6a52.jpg	Brain-2	V	V	
photo_8c0c6a53.jpg	Brain-3	V	V	

» Top

Figure 3-11. File Status

Figure 3-12. Medical Image Preview

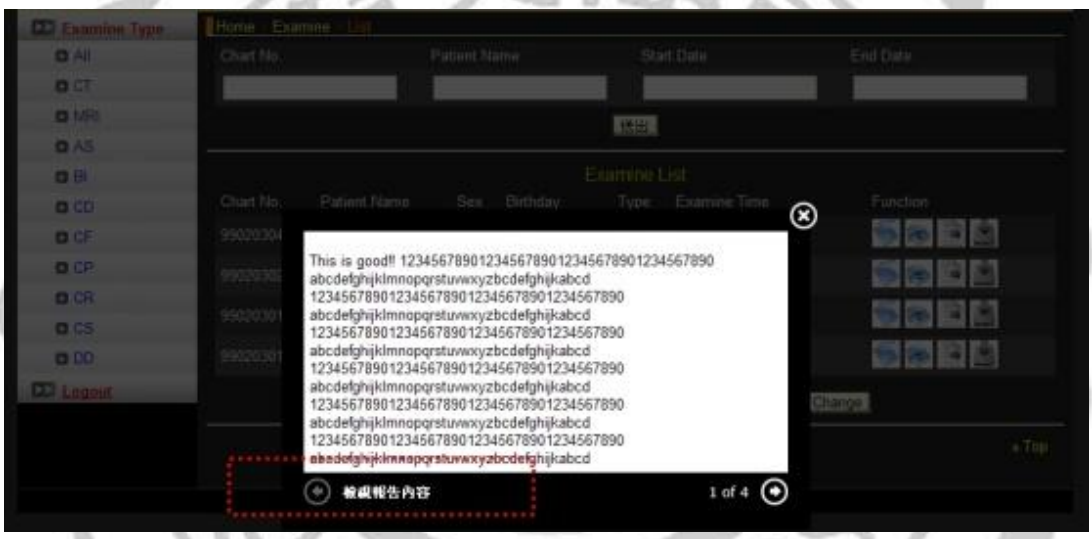


Figure 3-13. Patient Records

Figure 3-14. Upload Medical Images

3.5 Virtualization Manager Interface

We design a useful web interface for end users fastest and friendly to Implementation virtualization environment. In Figure 3-15, it shows the authorization mechanism, through the core of the web-based management tool, it can control and manage physical machine and VM life-cycle.

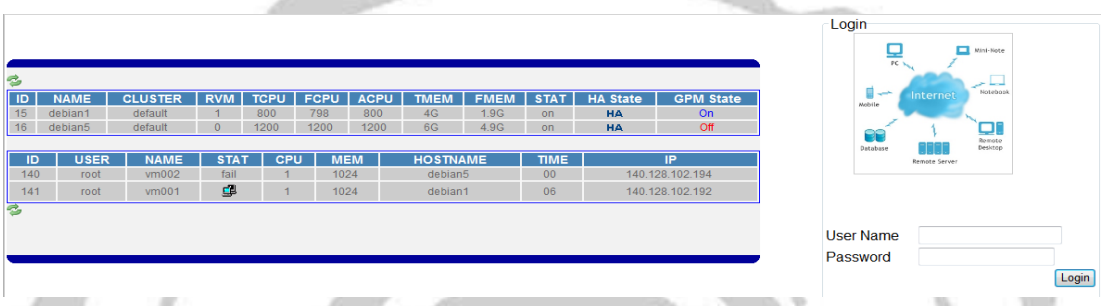


Figure 3-15. Web-Based Interface

The entire web-based management tool including physical machine management, virtual machine management and performance monitor. In Figure 3-16 it can set the VM attributes such as memory size, IP address, root password and VM name etc…, it also including the life migrating function. Life migration means VM can move to any working physical machine without suspend in-service programs. Life Migration is one of the advantages of OpenNebula. Therefore we could migrate any VM what we want under any situation, thus, we have a DRA mechanism to make the migration function more meaningful.

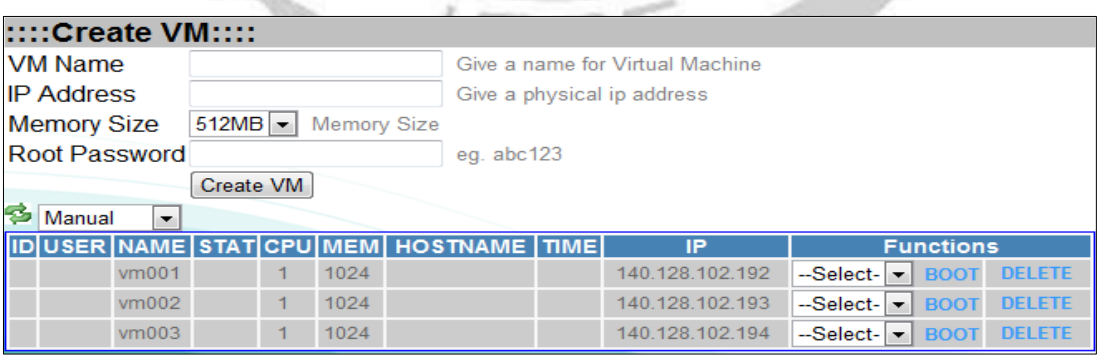


Figure 3-16. Virtual Machines Manager

Chapter 4

Experimental Results

In this section which, we made the environment for the entire fault-tolerant ability to do that, including a network failure, Datanodes fault, Namenode made when the occasion so tolerant.

4.1 Experimental Environments

In MIFAS environment, HDFS build 3 nodes (THU1, THU2 and CSMU). For each Namenode are done in two VMs configuration, and use the DRBD with heartbeat sync to do this part of the configuration, were configured for each Namenode four Datanodes. Figure 4-1 details the environment.

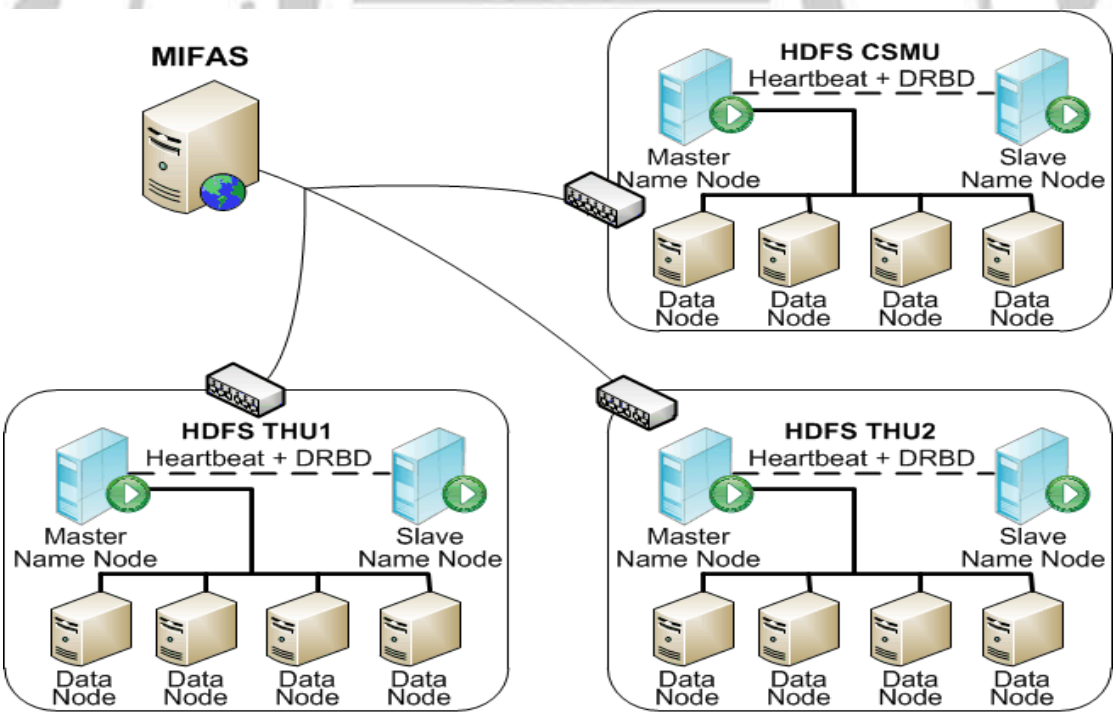


Figure 4-1. Experimental Environments

4.2 Results

4.2.1 Performance Comparison

At this part we do stress testing with JMeter. We set 10 Threads, and Loop count 5 times more physical machines and virtual machine on the environment were to download 1MB, 10MB and 50MB file sizes, etc., the resulting throughput and the ability to download data.

The results of Figure 4-2 mean the smaller of file size will enable greater throughput, and physical machines and VMs will be more obvious differences. In Figure 4-3 shows we download a small file, VMs transmission performance will be better than physical machines.

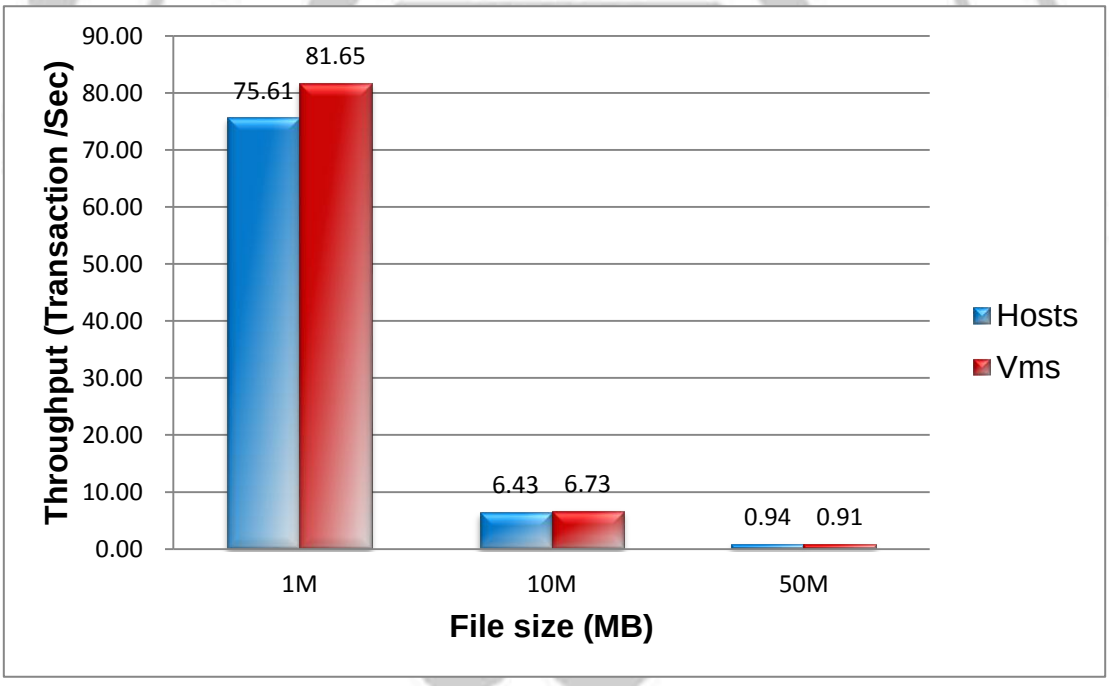


Figure 4-2. Compare of Physical Host and Virtual Machine Throughput

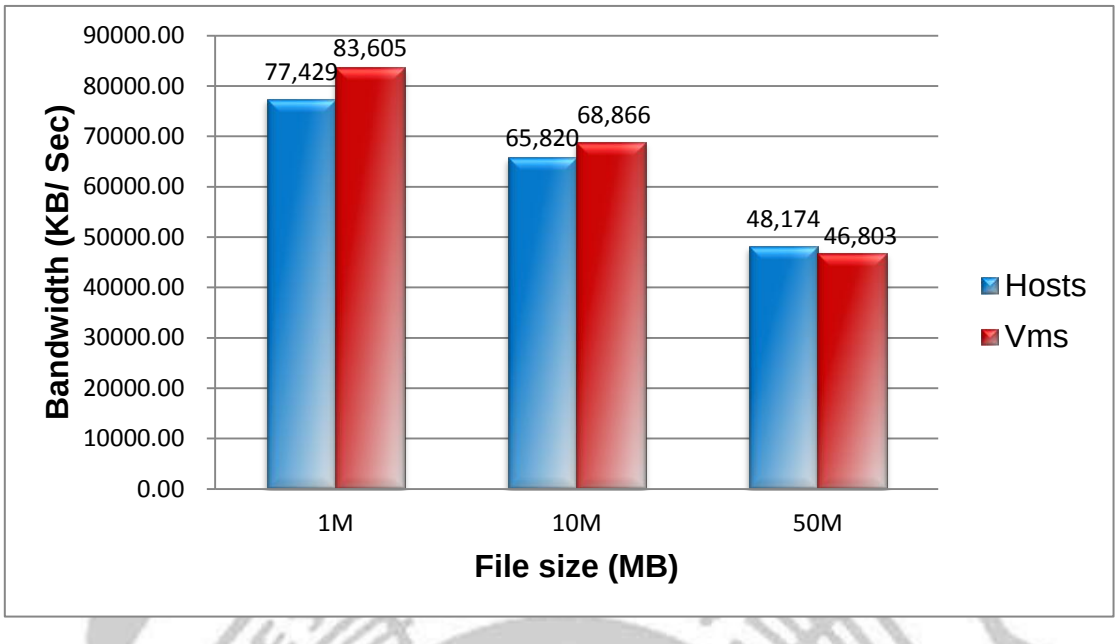


Figure 4-3. Compare of Physical Host and VM Networking Performance

In this experiment, we download the same files from each PACS and MIFAS. The purpose of this experiment is to compare of PACS and MIFAS Networking Performance. Figure 4-4 shows the results, a smaller download file, MIFAS better transmission capacity, whereas in downloading large files, PACS has better transmission performance.

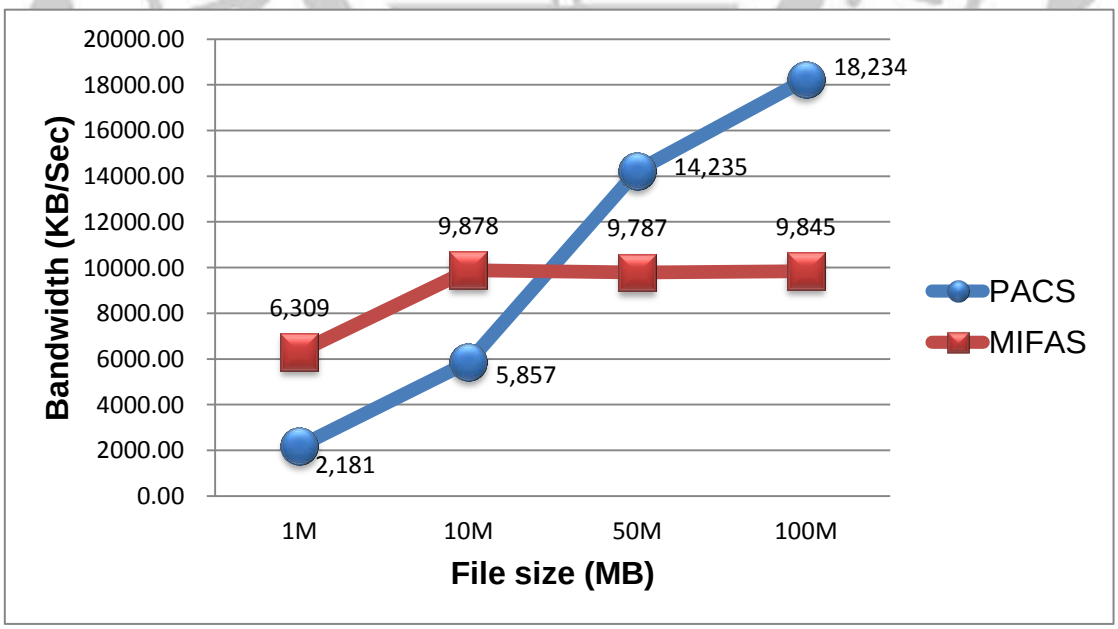


Figure 4-4. Compare of PACS and MIFAS Networking Performance

4.2.2 Network Fault Tolerance

In this system, we used three HDFS node to access data. When a HDFS node which network disconnection, through the co-allocation mechanism, Information service will note that the current network node there is a problem. When users access to data, system will make the current surviving HDFS nodes to do distribution of the current file transfer request to the user. Figure 4-5, we interrupt the THU2 HDFS and CSMU HDSF network, then the system will transfer data THU1 HDFS as the main node.

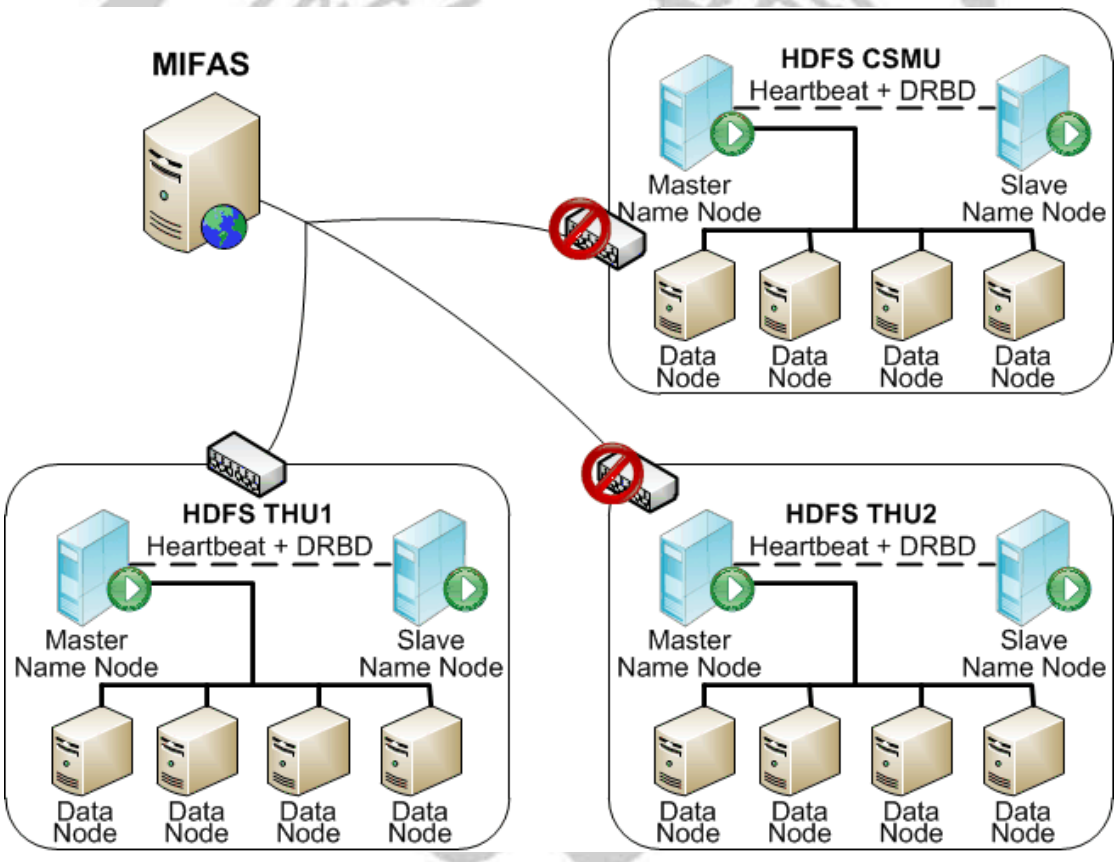


Figure 4-5. Network Fault Tolerance

4.2.3 Datanodes Fault Tolerance

Each of HDFS is deployment four Datanodes of the physical machine. We set the block size of the file 64MB, three copies of each block to store the Datanodes.

HDFS is used to coordinate access by Namenode, through the metadata configuration file to know the block where the Datanodes, We do note HDFS on THU1 (Figure 4-6). In the environment, two of machine failure and does not affect the HDFS data downloads.

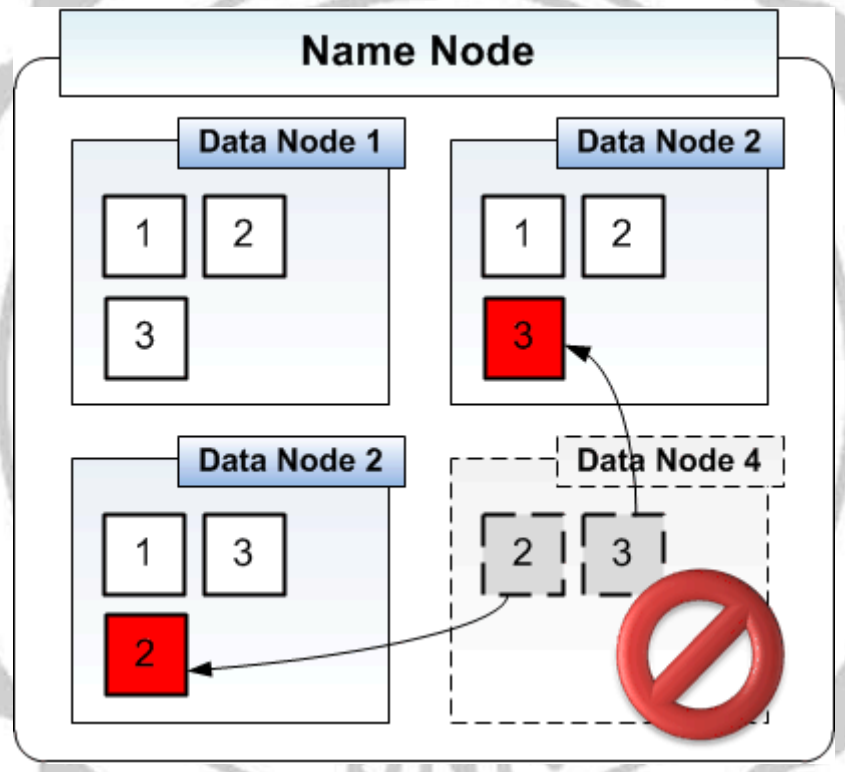


Figure 4-6. Single Site of Hardware Failure

Here we tested less Datanodes and multi Datanodes performance comparison. We set 10 threads and Loop count 5 times more 1-3 Datanodes on the THU1 HDFS environment, the resulting throughput and the ability to download and upload data, file size is 100MB.

The result of Figure 4-7 and Figure 4-8 shows less Datanodes for the transmission performance will not have much impact.

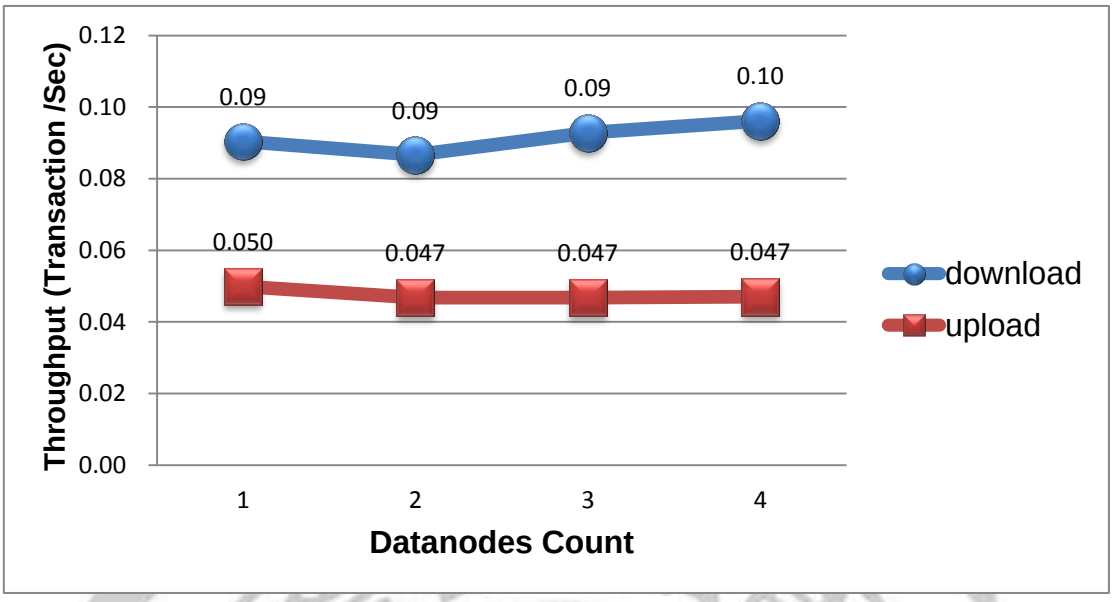


Figure 4-7. Compare of HDFS Download and Upload Throughput

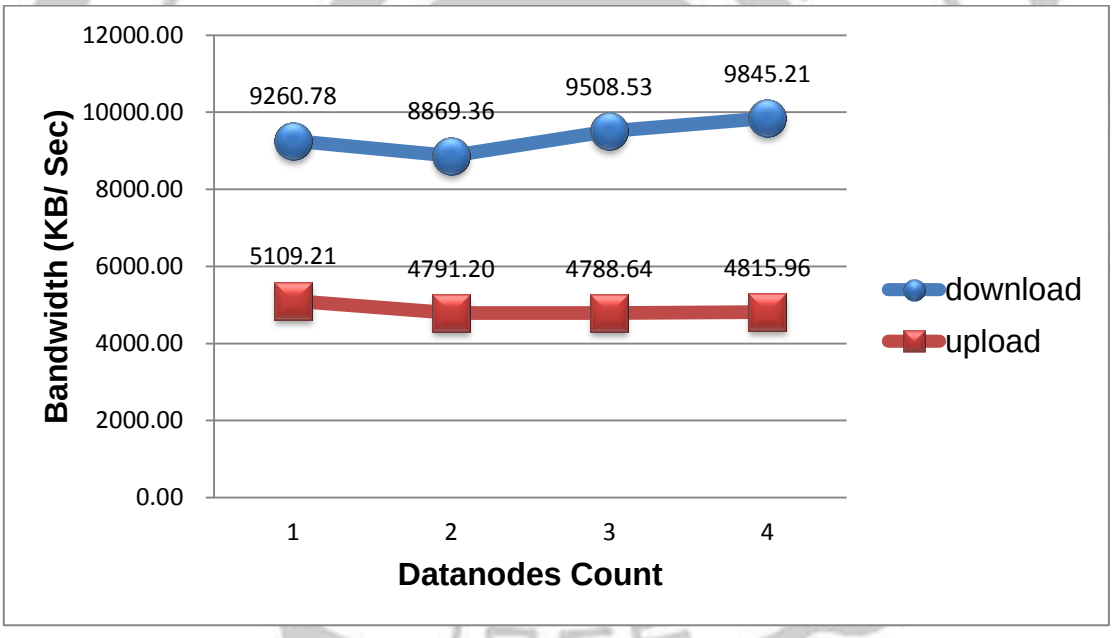


Figure 4-8. Compare of HDFS Download and Upload Networking Performance

4.2.4 Use DRBD with Heartbeat on Namenode

Though make Datanodes have a better fault tolerance by HDFS configure. However, if the Namenode fails, the whole information will can't access HDFS.

Here we added a mechanism DRBD with Heartbeat, DRBD can synchronize the contents of the local disk to another host. With the heartbeat function, you can build high-availability environment.

Figure 4-9 shows the HDFS in the THU1, we use this mechanism to establish a Master Namenode and Slave Namenode, and the data real-time synchronization. When the Master Namenode abnormal network (cable break or NIC failure), Slave Namenode to detect this condition, we will stop the Primary Node in Hadoop, then took over by the Secondary Node for the Primary Node.

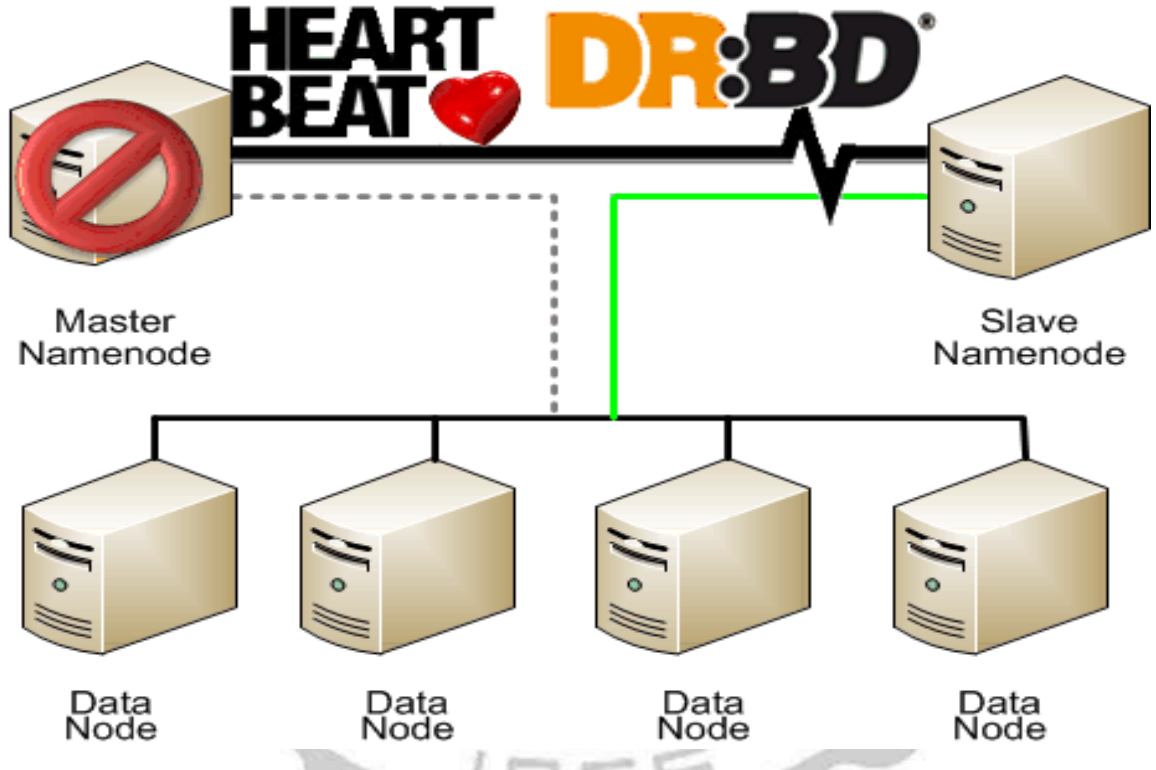


Figure 4-9. DRBD with Heartbeat on physical environment

4.2.5 Use DRBD with Heartbeat under Live Migration

Abnormal or major network nodes at the right time to die, although through DRBD with Heartbeat way to do high availability Failover. But the abnormal Primary node fails, and

Secondary node fails too, will result this HDFS is not available.

Created through our virtual environment, we will Namenode based on the hypervisor in Xen. When the Master Namenode network interruption or failure of a physical machine, secondary node will then convert the primary node, and then through opennebula's live migration technology to convert Primary Namenode to other survivors of the physical machine execution.

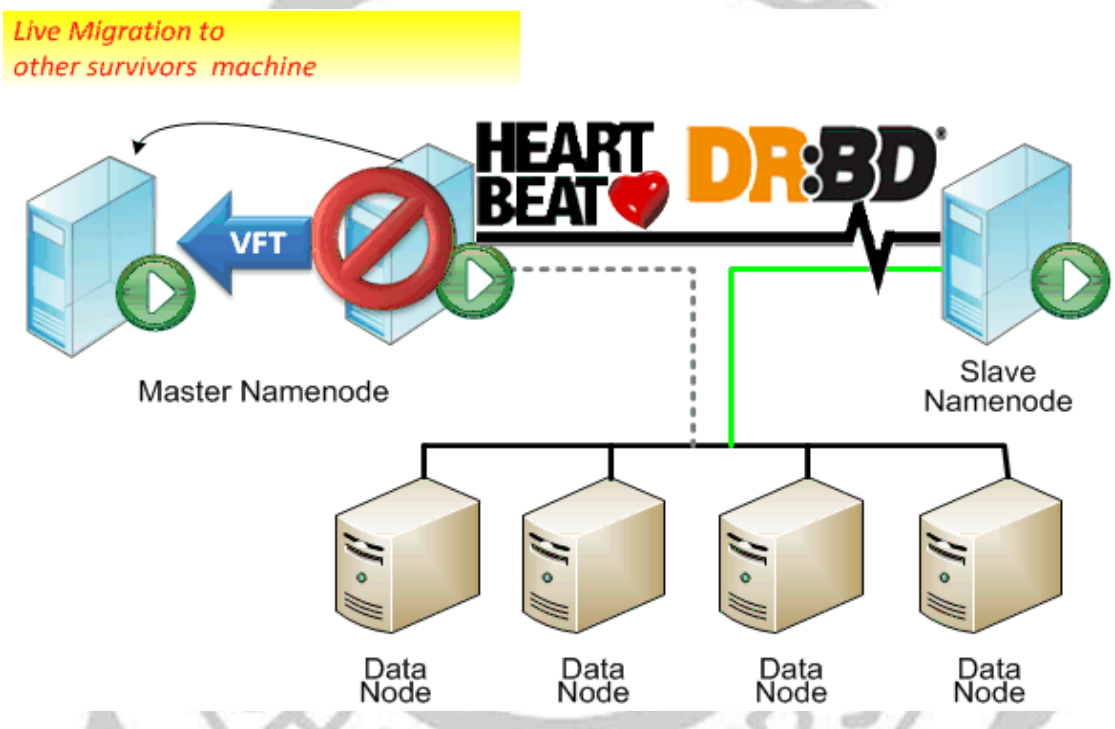


Figure 4-10. DRBD with Heartbeat on virtual environment

This section describes the access HDFS in order to increase the reliability of MIFAS to THU1 HDFS in the experiment, in the Namenode failure of the system's fault tolerance. Due to the system to view or download DICOM files, are required in the HDFS's Namenode through to each Datanodes to fetch files. Therefore, Namenode failure state, the node HDFS is invalid.

Service IP: stand for provide service channel to external users, users through this IP to access services. VM2 is primary node, VM1 is secondary. The different between primary and

secondary please refer to section Ⅱ.A. Debian1, Debian2 and Debian3 are the hosts, VM2 is living on Debian 1, and the secondary node VM1 is living on Debian2. The whole environment is as shown illustration as Figure 4-10.

In Figure 4-11, we shutdown Debian1 doesn't cause the Service IP stop providing service. You can see the connection status of Service IP in the bottom of Figure 4-12, it only lost one pack. The reason is under our VFT mechanism (Figure 4-13), the secondary node is replaced primary node immediately. In HA speak it called FAILOVER. It is a good practice of HA mechanism. And VFT also boot on the VM2 to on-line host Debian3. In this case, we can say VFT is a good solution to solve HA problem on virtualization.

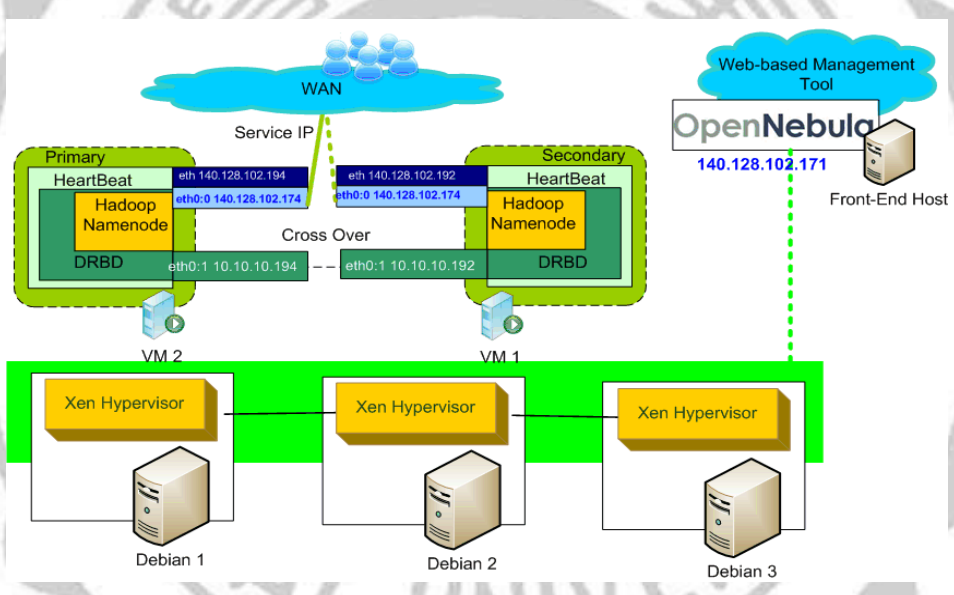


Figure 4-11. Experimental Environments

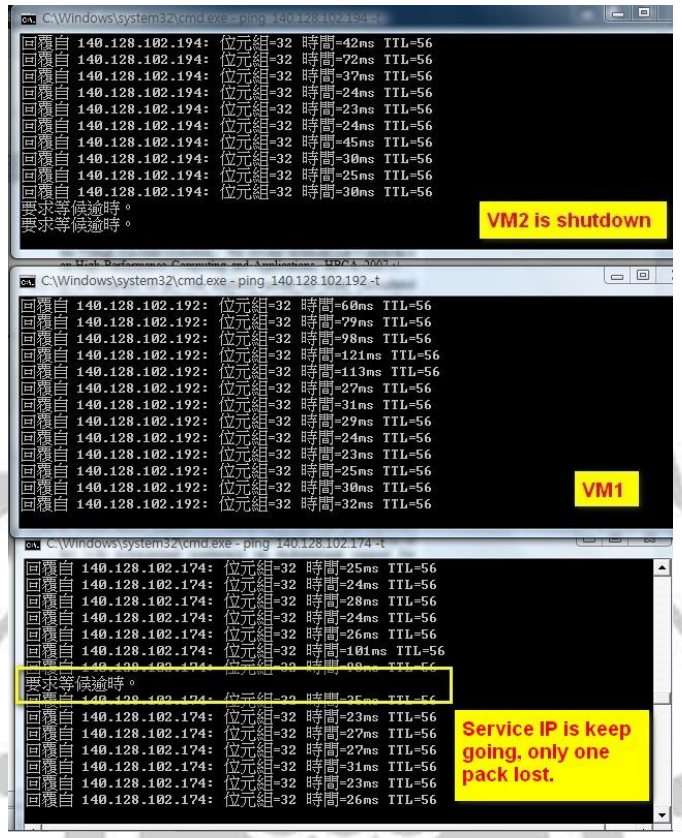


Figure 4-12. Shutdown Host and VM2

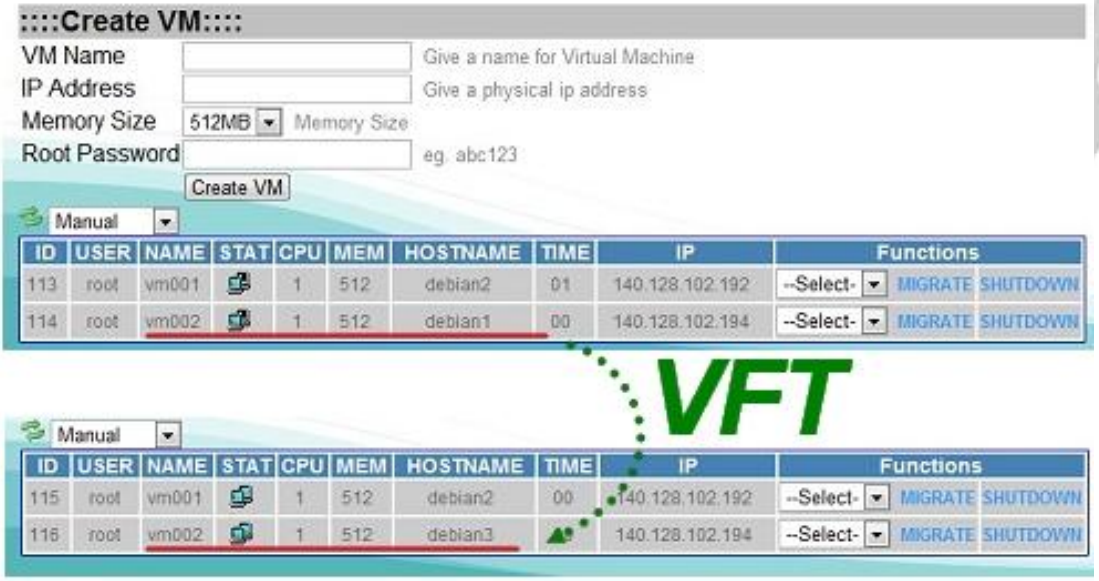


Figure 4-13. VFT Mechanism

Chapter 5
Conclusion and Future Work

5.1 Concluding Remark

At present the computer node failure can usually be detected through the monitoring mechanism for system health. Compared to passive solutions, the recovery has occurred in response to failure, we are actively promoting the virtualization fault tolerance (VFT). Systems that exhibit truly continuous availability are comparatively rare and higher priced, and most have carefully implemented specialty designs that eliminate any single point of failure and allow online hardware, network, operating system, middleware, and application upgrades, patches, and replacements. Zero downtime system design means that modeling and simulation indicates mean time between failures significantly exceeds the period of time between planned maintenance, upgrade events, or system lifetime.

We use VFT mechanism to build a high availability of HDFS. Combining virtualization techniques, load balancing, health monitoring and live migration, to be able to run virtual machines from hardware failure on one machine and restart on another machine without losing any state. Xen's live migration allows a guest OS to be relocated to another node, including running tasks of a Namenode job. DRBD can avoid a single hardware failure affecting the host system operation, and the two hosts of the information produced by DRBD machine instant synchronization, data inconsistencies will not occur. We exploit this feature when a health-deteriorating node is identified, which allows computation to proceed on a healthy node, thereby avoiding a complete restart necessitated by node failures. Experimental results show that this solution can effectively increase the system availability.

5.2 Future Work

In the future, for the entire system to enhance the part of the following:

- Because the DICOM standard is updated constantly, for the DICOM parser part, we have to correspond, in order to achieve a complete inspection data.
- In the current Web page can view the patient's examination picture, but can’t be editing notes, and that it will do this part of the plan.
- HDFS parameters for performance optimization to do the adjustment.

Bibliography

[1] *Prepare for Disasters & Tackle Terabytes When Evaluating Medical Image Archiving*. Available: http://www.frost.com/

[2] C. C. Teng, J. Mitchell, C. Walker, A. Swan, C. Davila, D. Howard, and T. Needham, "A medical image archive solution in the cloud," in *Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on*, 2010, pp. 431-434.

[3] L. A. B. Silva, C. Costa, and J. L. Oliveira, "A PACS archive architecture supported on cloud services," *International Journal of Computer Assisted Radiology and Surgery*, pp. 1-10, 2011.

[4] D. D. J. D. Macedo, A. V. Wangenheim, M. A. R. Dantas, and H. G. W. Perantunes, "An architecture for DICOM medical images storage and retrieval adopting distributed file systems," *Int. J. High Perform. Syst. Archit.*, vol. 2, pp. 99-106, 2009.

[5] C. T. Yang, L. T. Chen, W. L. Chou, and K. C. Wang, "Implementation of a Medical Image File Accessing System on Cloud Computing," in *Computational Science and Engineering (CSE), 2010 IEEE 13th International Conference on*, 2010, pp. 321-326.

[6] L. Faggioni, E. Neri, C. Castellana, D. Caramella, and C. Bartolozzi, "The future of PACS in healthcare enterprises," *European Journal of Radiology*, 2010.

[7] E. Bellon, M. Feron, T. Deprez, R. Reynders, and B. V. d. Bosch, "Trends in PACS architecture," *European Journal of Radiology*, vol. 78, pp. 199-204, 2010.

[8] L. N. Sutton, "PACS and diagnostic imaging service delivery--A UK perspective," *European Journal of Radiology*, vol. 78, pp. 243-249, 2011.

[9] G. Ganapathy and S. Sagayaraj, "Circumventing Picture Archiving and Communication Systems Server with Hadoop Framework in Health Care Services," *Journal of Social Sciences*, vol. 6, pp. 310-314, 2010.

[10] *National Institute of Standards and Technology(NIST)*. Available: http://www.nist.gov/

[11] J. Venner, *Pro Hadoop. 1st Edn*, 2009.

[12] R. L. Grossman, Y. Gu, M. Sabala, and W. Zhang, "Compute and storage clouds using wide area high performance networks," *Future Generation Computer Systems*, vol. 25, pp. 179-183, February 2009 2009.

[13] J. Shafer, S. Rixner, and A. L. Cox, "The Hadoop distributed filesystem: Balancing portability and performance," in *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*, 2010, pp. 122-133.

[14] J. Liu, L. Bing, and S. Meina, "THE optimization of HDFS based on small files," in *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, 2010, pp. 912-915.

[15] W. v. Hagen, *Professional Xen Virtualization*: Wrox; 1 edition (January 29, 2008), 2008.

[16] J. P. Walters, V. Chaudhary, M. Cha, S. G. Jr., and S. Gallo, "A Comparison of Virtualization Technologies for HPC," in *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, 2008, pp. 861-868.

[17] J. Zhu, Z. Jiang, Z. Xiao, and X. Li, "Optimizing the Performance of Virtual Machine Synchronization for Fault Tolerance," *Computers, IEEE Transactions on*, vol. PP, pp. 1-1, 2010.

[18] T. C. Bressoud and F. B. Schneider, "Hypervisor-Based Fault-Tolerance," *SIGOPS Oper. Syst. Rev.*, vol. 29, pp. 1-11, 1995.

[19] J. Walters and V. Chaudhary, "A fault-tolerant strategy for virtualized HPC clusters," *THE JOURNAL OF SUPERCOMPUTING*, vol. 50, pp. 209-239, 2009.

[20] *Clinical Data Update System (CDUS)*. Available: http://www.cdus.org

[21] M. V. Rafael, R. S. Montero, and I. M. Llorente, "Elastic management of cluster-based services in the cloud," presented at the Proceedings of the 1st workshop on Automated

control for datacenters and clouds, Barcelona, Spain, 2009.

[22] C. T. Yang, H. Y. Cheng, W. L. Chou, and C. T. Kuo, " A Dynamic Resource Allocation Model for Virtual Machine Managemant on Cloud," in *Symposium on Cloud and Service Computing* 2011.

[23] B. Varghese, G. McKee, and V. Alexandrov, "Implementing intelligent cores using processor virtualization for fault tolerance," *Procedia Computer Science*, vol. 1, pp. 2197-2205, 2010.

[24] O. Villa, S. Krishnamoorthy, J. Nieplocha, and D. M. J. Brown, "Scalable transparent checkpoint-restart of global address space applications on virtual machines over infiniband," presented at the Proceedings of the 6th ACM conference on Computing frontiers, Ischia, Italy, 2009.

[25] J. E. Simons and J. Buell, "Virtualizing high performance computing," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 136-145, 2010.

[26] *DICOM*. Available: http://medical.nema.org/

[27] C. T. Yang, C. H. Tseng, K. Y. Chou, S. C. Tsaur, C. H. Hsu, and S. C. Chen, "A Xen-Based Paravirtualization System toward Efficient High Performance Computing Environments," in *Methods and Tools of Parallel Programming Multicomputers*. vol. 6083, C.-H. Hsu and V. Malyshkin, Eds., ed: Springer Berlin / Heidelberg, 2011, pp. 126-135.

[28] *VMware*. Available: http://www.vmware.com/

[29] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive fault tolerance for HPC with Xen virtualization," presented at the Proceedings of the 21st annual international conference on Supercomputing, Seattle, Washington, 2007.

[30] *OpenNebula*. Available: http://www.opennebula.org

[31] *Apache Hadoop Project*. Available: http://hadoop.apache.org/hdfs/

[32] *DRBD Official Site*. Available: http://www.drbd.org

[33] P. Pla, "Drbd in a heartbeat," *Linux J.*, vol. 2006, p. 3, 2006.

[34] A. L. N. Reddy and J. Wyllie, "Disk scheduling in a multimedia I/O system," presented at the Proceedings of the first ACM international conference on Multimedia, Anaheim, California, United States, 1993.

[35] 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," presented at the Proceedings of the Second Workshop on Grid Technologies and Applications (WoGTA' 05), 2005.

[36] C. T. Yang, S. Y. Wang, and C. P. Fu, "A Dynamic Adjustment Mechanism for Data Transfer in Data Grids," presented at the Network and Parallel Computing: IFIP International Conference, NPC 2007, 2007.

[37] 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," *ALGORITHMS AND ARCHITECTURES FOR PARALLEL PROCESSING*, vol. 4494, pp. 242-253, 2007.

[38] C. T. Yang, I. H. Yang, S. Y. Wang, C. H. Hsu, and K. C. Li, "A Recursively-Adjusting Co-allocation scheme with a Cyber-Transformer in Data Grids," *Future Generation Computer Systems*, vol. 25, pp. 695-703, July 2009 2009.

[39] C. T. Yang, I. H. Yang, K. C. Li, and S. Y. Wang, "Improvements on dynamic adjustment mechanism in co-allocation data grid environments," *J. Supercomput.*, vol. 40, pp. 269-280, 2007.

[40] C. T. Yang, S. Y. Wang, and W. Chu, "Implementation of a dynamic adjustment strategy for parallel file transfer in co-allocation data grids," *THE JOURNAL OF SUPERCOMPUTING*, vol. 54, pp. 180-205, 2009.

[41] *Nanodicom*. Available: http://www.nanodicom.org/

[42] *Hadoop-over-ftp*. Available: http://www.hadoop.iponweb.net/Home/hdfs-over-ftp

[43] *Curlftp*. Available: http://curlftpfs.sourceforge.net/

[44] *Ganglia*. Available: http://ganglia.sourceforge.net/

APPENDIX A

Prepare Software

No	Service	Version	Description	
1	Apache	2.2.15	Application	Web Service
2	MySQL	5.147	Database System	
3	PHP	5.3.3	Scripting Language	
4	phpMyAdmin	2.8.2.1	MySQL Management Tool	
5	Java	6u23	Java Runtime Environment	Hadoop Service
6	Hadoop	0.20.3-dev	One of Apache Projects	
7	HDFS-over-ftp	1.0.1	Connect to HDFS using FTP client	
8	Curlftpfs	0.9.2	file system for accessing FTP hosts	
9	Xen	5.0	Hypervisor	Virtualization Service
10	OpenNebula	2.0	VM Management	

APPENDIX B

Installation Guide

A. Configure networking

The following is an example from our systems.

Edit the file /etc/hosts

```
127.0.0.1          localhost
10.1.1.211         debian-ha1
10.1.1.212         debian-ha2
192.168.123.210 hadoop.namenode
```

Edit the file /etc/network/interface:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
allow-hotplug eth0
allow-hotplug eth1
iface eth0 inet static
        address 192.168.123.211
        netmask 255.255.255.0
        network 192.168.123.0
        broadcast 192.168.123.255
```

```
        gateway 192.168.123.254
        dns-nameservers 168.95.1.1
        dns-search csie.thu.edu.tw
iface eth1 inet static
        address 10.1.1.211
        netmask 255.255.255.0
```

Finally, reboot the system or restart networking:

B. Java

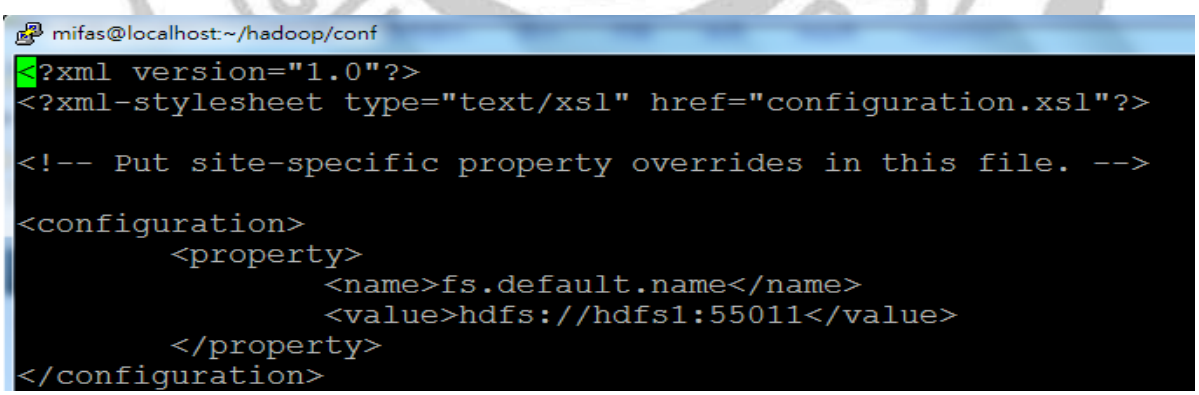
http://www.oracle.com/technetwork/java/javase/downloads/index.html

```
$mv jre-6u23-linux-i586.bin to /usr
$sh jre-6u23-linux-i586.bin
```

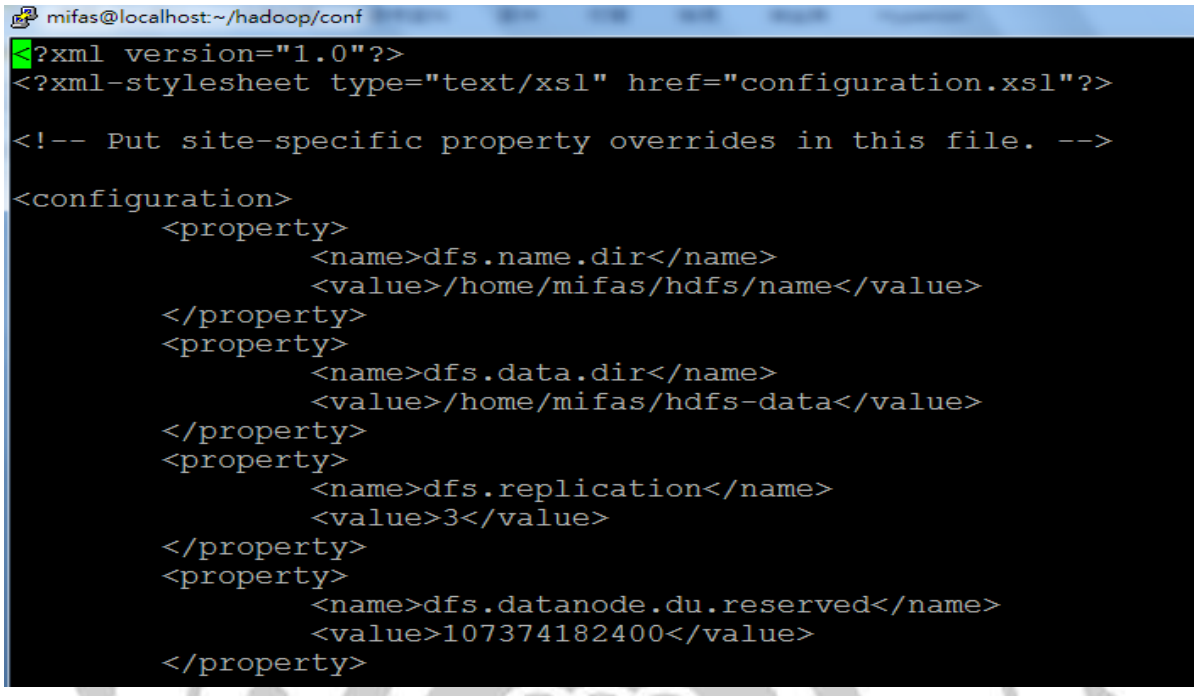
C. Hadoop

http://hadoop.apache.org/

conf/core-site.xml



conf/hdfs-site.xml



dfs.name.dir: Namenode storage

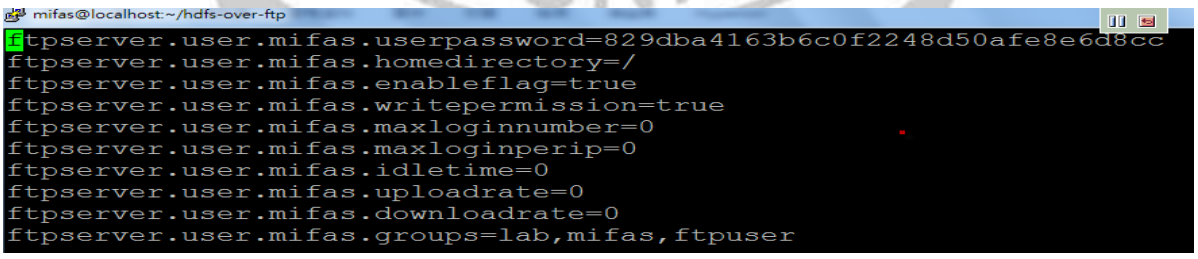
dfs.data.dir: Datanode storage

dfs.replication: Number of replication

D. Hdfs-over-ftp

http://www.hadoop.iponweb.net/Home/hdfs-over-ftp

users.conf



hdfs-over-ftp.conf

```
mifas@localhost:~/hdfs-over-ftp
#uncomment this to run ftp server
port = 2222

data-ports = 2223-2500

#uncomment this to run ssl ftp server
ssl-port = 2501
ssl-data-ports = 2502-2500
keystore-password = 333333

# hdfs uri
hdfs-uri = hdfs://hdfs1:55011

# have to be a user which runs HDFS
# this allows you to start ftp server as a root to use 21 port
# and use hdfs as a superuser
superuser = mifas
```

Check service status

```
mifas     1898  0.0  2.1 1013964 67836 ?       Sl   Jul26   3:33 java -cp .:
lib/ftplet-api-1.0.0.jar:lib/slf4j-api-1.5.2.jar:lib/ftpserver-core-1.0.0.ja
r:lib/jcl-over-slf4j-1.5.2.jar:lib/slf4j-log4j12-1.5.2.jar:lib/hadoop-core-0
.20.0.jar:lib/log4j-1.2.14.jar:lib/hdfs-over-ftp-1.0-SNAPSHOT.jar:lib/mina-c
ore-2.0.0-M4.jar HdfsOverFtpServer
```

```
[mifas@localhost hdfs-over-ftp]$ netstat -an | grep 2222
tcp       25      0 140.128.102.175:59282       140.128.98.15:2222          CLOSE_WAIT
tcp       24      0 127.0.0.1:32841             127.0.0.1:2222              ESTABLISHED
tcp        0      0 :::2222                     :::*                        LISTEN
```

E. Curlftpfs

http://sourceforge.net/projects/curlftpfs/

$ apt-get install curl

$mkdir /home/mifas/public_html/www/pacsdata

$cd /home/mifas/public_html/www

$sudo curlftpfs -v -o allow_other ftp:// ****:****@140.128.98.20:2222 pacsdata

Check service status

```
[mifas@localhost ~]$ ps aux | grep curlftpfs
root      1928  0.0  0.1 132484  3884 ?        Ssl  Jul26   0:02 curlftpfs -o allow_other ftp:/
/mifas:1qazxdr5@127.0.0.1:2222 pacsdata
root      2297  0.0  0.1 1648892 5696 ?        Ssl  Jul26   0:07 curlftpfs -o allow_other ftp:/
/mifas:1qazxdr5@140.128.98.15:2222 pacsdata2
```

F. Xen

```
$ sudo aptitude -y install xen-tools

$ cd /etc/xen-tools

$ sudo gedit xen-tools.conf

dir = /data/xen

size      = 4Gb

dist      = lenny

gateway      = 192.168.200.254

netmask      = 255.255.255.0

broadcast = 192.168.200.255

passwd = 1

mirror = http://free.nchc.org.tw/debian

serial_device = hvc0

output        = /data/domains
```

G. OpenNebula

http://dev.opennebula.org/

```
$ wget http://dev.opennebula.org/attachments/download/103/one-1.4.0.tar.gz

$ tar zxvf one-1.4.0.tar.gz

$ cd one-1.4

$ sudo scons

$ sudo mkdir /home/one

$ sudo ./install.sh -d /home/one

$ sudo su
```

```
# echo export ONE_LOCATION=/home/one >> ~/.bashrc
# echo export ONE_XMLRPC="http://localhost:2633/RPC2" >> ~/.bashrc
# echo export PATH='$ONE_LOCATION/bin:$PATH' >> ~/.bashrc
# echo export ONE_AUTH=/home/one/.one/one_auth >> ~/.bashrc
# mkdir /home/one/.one
# echo "root:cloud123" >> /home/one/.one/one_auth
```

H. DRBD and Heartbeat

http://www.drbd.org/

```
# apt-get -y install drbd82 kmod-drbd82 heartbeat
```

/etc/drbd.conf

```
global {     usage-count yes;    }
common {     syncer { rate    30M; }    }
resource r0 {
   protocol C;
   startup {
      wfc-timeout 0;
      degr-wfc-timeout 120;
   }
   disk {
      on-io-error      detach;
      # no-disk-flushes;
      # no-md-flushes
      # size 1G;
    }
```

```
   net {
    }
on debian-ha1 {
    device      /dev/drbd0;
    disk        /dev/sdb1;
    address     10.1.1.211:7789;
    meta-disk internal;
   }
on debian-ha2 {
    device      /dev/drbd0;
    disk        /dev/sdb1;
    address     10.1.1.212:7789;
    meta-disk internal;
   }
}
admin@debian-ha1:/etc/network$ clera
-bash: clera: command not found
admin@debian-ha1:/etc/network$ clear
admin@debian-ha1:/etc/network$ cat /etc/drbd.conf
global {
   usage-count yes;
}
common {
   syncer { rate    30M; }
}
```

```
resource r0 {
    protocol C;
    startup {
        wfc-timeout 0;
        degr-wfc-timeout 120;
    }
    disk {
        on-io-error     detach;
        # no-disk-flushes;
        # no-md-flushes
        # size 1G;
      }
    net {
      }
on debian-ha1 {
      device      /dev/drbd0;
      disk        /dev/sdb1;
      address     10.1.1.211:7789;
      meta-disk internal;
    }
on debian-ha2 {
      device      /dev/drbd0;
      disk        /dev/sdb1;
      address     10.1.1.212:7789;
      meta-disk internal;
```

```
  }
}
```

I. Heartbeat Configuration

Create Soft Link

```
#cd /etc/ha.d/resource.d
      #ln -s /etc/init.d/hadoop-0.20-namenode hadoop-namenode
      #ln -s /etc/init.d/hadoop-0.20-jobtracker hadoop-jobtracker
```

/etc/ha.d/ha.cf

```
## start of ha.cf
logfile /var/log/ha-log
logfacility       local0

keepalive 2          #Detection period
warntime 5
deadtime 20
initdead 120
#hopfudge 1

udpport 694          #Using UDP 694
auto_failback off    #if failback, resume to master
#baud 19200
bcast eth1           #using eth1, to be the heartbeat network card
ucast eth0 192.168.123.211
ucast eth1 10.1.1.211
```

```
node debian-ha1      #Node 1, Server Name
node debian-ha2      #Node 2, Server Name

ping 192.168.123.254          #Ping our Gateway, check heart self

respawn hacluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=haclient uid=hacluster
## end of ha.cf
```

/etc/ha.d/haresources

```
#vim /etc/ha.d/haresources
debian-ha1 192.168.123.210/24 drbddisk::r0 Filesystem::/dev/drbd0::/drbd::ext3::noatime
hadoop-namenode
```

Heartbeat Restart

```
#/etc/init.d/heartbeat stop
#/etc/init.d/heartbeat start
```