2010-2011 Final Year Project Report (1st Term)

Cloud computing technologies and applications

Supervised by Prof. Michael R. Lyu

Group: LYU1005
Chu Yan Shing (1008618841)
Wu Bing Chuan (1008612783)
This is a blank page
Abstract

This report is introducing our group’s final year project. We will describe our project’s:

- Project objective,
- Experiment we did,
- Benefits and trade off of using cloud and Hadoop,
- Problems we solved.

The objective of our project is using a private cloud and Hadoop to handle massive data processing like video searching.

In this final year project, we choose to use two technologies. The first one is Cloud which allows us to allocate the resource with an easy and efficient way. The cloud we choose is Ubuntu Enterprise private Cloud. The second technology we use is Hadoop, it is a software framework to implement data-intensive program with thousand of machines. Eucalyptus provides the resource in form of individual virtual machine, and Hadoop manage all the resource of the virtual machine. Hadoop evenly allocate the storage, computation power and divide large jobs to separate machines with many tiny jobs.

Since we are using two ‘new’ technologies (cloud and Hadoop) to handle a large scale job, we must fully understand the characteristic of these systems. Aim to fully understand two technologies, we did many experiment on testing the performance, especially the scalability of two technologies and optimize the environment.

We find private cloud powerful but with much limitation, especially the performance trades off. Hadoop allow us to easily process large amount of data in parallel which is difficult to handle in traditional ways.

At the middle of this report, we will describe the advantages and disadvantages of using Cloud and Hadoop.
# Table of content

Chapter 1: Introduction ............................................................................................................................... 7
   1.1 Motivation........................................................................................................................................ 7
   1.2 Project Objective ............................................................................................................................. 8
   1.3 Idea of our project............................................................................................................................ 8
   1.3 Project resource ............................................................................................................................... 9
      1.3.1 Hardware .................................................................................................................................. 9
      1.3.2 Software and framework .......................................................................................................... 11

Chapter 2: Cloud Computing ....................................................................................................................... 12
   2.1 Introduction to cloud’s concept ....................................................................................................... 12
   2.2 From point of views of resource ..................................................................................................... 12
      2.2.1 Types of cloud ...................................................................................................................... 12
      2.2.2 Types of cloud service .......................................................................................................... 13
      2.2.3 User interface ....................................................................................................................... 15
      2.2.4 Why cloud is so powerful .................................................................................................... 16
      2.2.5 Usage of public cloud ........................................................................................................... 17
      2.2.6 Private cloud ......................................................................................................................... 20
   2.3 From point of views of application structure ................................................................................... 23
      2.3.1 Using Cloud Layer ............................................................................................................... 23
      2.3.2 Using Cloud instance with other framework ......................................................................... 26
   2.4 Summary of concept of Cloud ........................................................................................................ 26
   2.5 Eucalyptus (UEC) ........................................................................................................................ 27
      2.5.1 Principal of eucalyptus cloud ............................................................................................... 27
      2.5.2 Our private cloud .................................................................................................................. 29
   2.5.3 Technical trouble shoot ............................................................................................................. 32
      2.5.4.1 Startup time .................................................................................................................... 35
      2.5.4.2 Overall performance ........................................................................................................ 35
      2.5.4.3 IO performance ............................................................................................................... 36
      2.5.4.4 Computation performance ................................................................................................ 37
      2.5.4.5 Memory Access time ....................................................................................................... 37
      2.5.4.6 Multi-platform .................................................................................................................. 38
      2.5.4.7 KVM and XEN ................................................................................................................. 39
2.5.4.8 Exception in running speed ........................................40
2.5.4.9 Virtio .................................................................40
2.5.4.10 Limitation and weakness of Eucalyptus ....................41

2.5.6 Cloud Summary ......................................................43

Chapter 3: Apache Hadoop .................................................44
3.1 Introduction to Hadoop .................................................44
3.2 HDFS .........................................................................47

3.2.1 HDFS Structure ......................................................47
3.2.1.1 Namenode ..........................................................47
3.2.1.2 SecondaryNamede .................................................48
3.2.1.3 Datanode ..........................................................49
3.2.1.4 Other Features ....................................................49
3.2.1.5 Compare with NFS .............................................50
3.2.1.6 Limitation of HDFS .............................................52

3.3 MapReduce .....................................................................52
3.3.1 Machine Roles ........................................................52
3.3.2 Programming Model .................................................53
3.3.3 Execution flow ........................................................53
3.3.4 Anatomy of a Job .....................................................55
3.3.5 Features .....................................................................57
3.3.6 Our testing ..............................................................58

3.3.6.1 Image processing .................................................59
3.3.6.2 Hadoop Word count example ................................60
3.3.6.3 MD5 Generator ....................................................62
3.3.6.3 Video Conversion ................................................64

3.4 Trouble shoot ..............................................................65

3.5 Hadoop summary .........................................................66

Chapter 4: Overall conclusion .............................................68
4.1 Difficulty of this project ................................................69

Chapter 5: What we will do next .........................................70
Chapter 6: Acknowledgement .............................................70
Chapter 7: Reference list ....................................................71
Appendix ............................................................................74
Progress table ........................................................................................................74
Cloud performance testing outputs .................................................................75
  Performance testing of our private cloud.........................................................75
Chapter 1: Introduction

1.1 Motivation

File size keeps increase

Although the computing power of machines is keeping increase in a very high speed. Almost every 3 years, CPU's computing power increase twice. However size of the files keeps increasing also in an amazing rate.

20 years ago, the common format is only text file. Later, computer can handle the graphics well, and play some low quality movies. In recent year, we are not satisfied in the quality of DVD, and introduce the Blu-ray disk. The file is changed from a few KBs to now nearly 1 TB.

Limitation of CPU core

CPU is the most important part of a computer. However, with current technology, CPU speed has reached the upper bound of speed. Directly increase the frequency of a CPU core is not the solution to improve the computation power as the temperature of such core may increase to thousand of degree. The current solution is putting more cores into a single CPU, but the best solution is running the application in multi cores and also in multi machine in parallel. Which are the ultimate aims of most of the application?
Parallel system can make computation faster

File size is increasing but the algorithms don’t improve so much. The improvement of using faster CPU but similar algorithms is limited. Moreover, newer format always require more complicated decode algorithms and make the processing time longer. The only way to greatly speed up the process is make the job parallel running on different machines.

Cloud reduce the cost

Since buying expensive facilities for only a single purpose is unreasonable but using the cloud charge only how long has we use. It enables many people to run large scale project with an acceptable cost.

We hope using cloud and some related technologies to implement application which are difficult to implement.

1.2 Project Objective

This project aims to implement large applications with Hadoop (a parallel programming software framework) on a cloud. Instead of programming very fast applications on a few machines, we are focusing on programming applications that has high scalability from a few machine to thousands of machines.

1.3 Idea of our project

Our idea is very simple. Assume there is a very large video data base. Giving a set of video or frames, we hope to find it from that database, and tell the position of that video.
The idea is simple but it is very useful in different aspects. If we change the algorithm to an object detection algorithm, it can be used in a surveillance video application. If we put face detection algorithm, it becomes a video diary.

The key point of this project is building an application with high scalability. When the database is increased, the application can still handle it.

### 1.3 Project Resource

Our project is going to introduce cloud and its performance. View Lab gives us machines to set up a cloud.

#### 1.3.1 Hardware

| Dell Desktop PC: |
|------------------|--------------------------------------------------|
| CPU: Intel Pentium D 3Ghz (support VT-x) |
| Memory: 1GB      |
| Storage: 160GB   |

Be the node server of the cloud. Kvm requires the physical hardware support VT-x or AMD-v, these are the
only machine we can use to build the cloud

<table>
<thead>
<tr>
<th>5 Desktop PC</th>
<th>One machine plays the roles as cloud controller and all machines has installed Hadoop and FFmpeg. So we can evaluate the performance of cloud and MapReduce, especially the video processing performance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X CPU: Pentium 4 2.8Ghz</td>
<td>Memory: 2GB</td>
</tr>
<tr>
<td></td>
<td>Storage: 160-240GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1 D-link wireless router</th>
<th>It plays the roles as DHCP sever and switch and help setting up a local network</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>1 Asus 1000M switch</th>
<th>Most of the machines are connected to this switch in order to achieve 1000M bandwidth and eliminate the bottle neck of the network</th>
</tr>
</thead>
</table>
This is our own netbook. It is connected to the local network and allows us to control the cloud and all other machines from anywhere within the CSE network.

- CAT 5E cables

1.3.2 Software and framework

Ubuntu Server Edition 10.10 (64Bits)

Ubuntu Enterprise Cloud (UEC) is the most common platform to set up a private cloud. There are more support and discussion on the internet.

Hadoop

MapReduce is a theory appears in recent year. Hadoop (software framework) is the famous open source project; it is very suitable to implement large scale data-intensive applications.
Chapter 2: Cloud Computing

2.1 Introduction to cloud’s concept

Cloud Computing is a web processing with large amount of resource. The user of the cloud can obtain the service thought network (in both internet and intranet). In other words, users are using or buying computing service from others. The resource of the cloud can be anything IT related. In general, cloud provides application, computation power, storage, bandwidth, database and some technologies like MapReduce. As the resource pool is very large, user can increase the application on cloud to any scale. It is fully under users’ control.

2.2 From point of views of resource

2.2.1 Types of cloud

There are four main type of cloud:

1) **Public cloud**: The cloud computing resource is shared outside, anyone can use it and some payment maybe need.

2) **Private cloud**: It is opposite to public cloud, private cloud’s resource is limit to a group of people, like a staff of a company etc.

3) **Hybrid cloud**: this is a mixture of previous two clouds, some cloud computing resource is shared outside but some don’t.

4) **Community cloud**: this is a special cloud to make use of cloud computing’s features. More than one community shares a cloud to share and reduce the cost of computing system. This will be discussing later.
We only focus on the private and public cloud. The typical example of the public cloud is Amazon Web Service (AWS EC2) and WSO2. We will introduce Eucalyptus and AWS later.

![WSO2 and Amazon Web Service](image)

**Figure 3 WSO2 and Amazon Web Service are common public cloud**

### 2.2.2 Types of cloud service

There are three main types of cloud service provided:

1. **Software as a Service (SaaS)**: Clients can use the software provided by the provider. Which usually need not to install and it is usually a one to many service. Like Gmail, search engine.

   The typical cloud application we used is Google Doc, a powerful web application similar to Microsoft Office. It is free for most users and has paid version for the company who want more features.

   ![Google Doc](image)

   **Figure 4 Google Doc is one of SaaS cloud**

2. **Infrastructure as a Service (IaaS)**: Clients can access computing resources (e.g., servers, storage, networking) on demand.

   ![Eucalyptus](image)

   **Figure 5 Eucalyptus**

3. **Platform as a Service (PaaS)**: It provides a platform that allows applications to be developed, run, and managed.

   ![Cloud Foundry](image)

   **Figure 6 Cloud Foundry**
a very powerful and can do almost all basic effects same as other desktop application. After finishing your work, it can compile the final product with a magical high speed and give you a download link. You can deliver the movies by passing the links to others and need not to worry about the storage problem.

Pixlr is an online version of Photoshop. It is also free and has many features similar to Photoshop e.g. layers and steps management. Actually, it is more user friendly than the Photoshop.

2) **Platform as a Service (PaaS):**

Clients can run their own applications on the platform provided; General platforms are Linux and Windows.

3) **Infrastructure as a Service (IaaS):** Client can put their own operation system on cloud. For example, user can put an optimized Linux for networking ability.

In practical usage, PaaS and IaaS are actually very similar with the difference that whether the image is provided by user or not. If you use the image provided, it is PaaS, otherwise, it is IaaS.

Cloud usually provides many images with pre-set-up environment like SQL server and PHP. Users treat it as online-shopping and buying the IT service.
2.2.3 User interface

Apart from access the application with web browsers. There are many other tools are developed to provide user-friendly interface. For example, Eucalyptus and AWS EC2 can manage instance control with Elastic Fox with is developed by amazon. And Eucalyptus provides the web interface to access the Configuration of the cloud.

Terminal of course is a very useful interface to control the cloud. It is also the best way to handle much complex control like running an instance with other

![Elasticfox](image)

**Figure 8 Elasticfox a tool to control AWS liked cloud**

![Ubuntu cloud's web interface](image)

**Figure 8 Ubuntu cloud’s web interface**
2.2.4 Why cloud is so powerful

First we define physical machines as cloud servers, and instance or virtual machine (VM) as the virtual server provided to the users.

**Resource pool is large**

Public cloud gives users surprise with its elastic property and giant resource pool with thousands of machines. With a few line of commands, user can use these machines to do anything they wanted to do.

**Easy to manage**

Private cloud gives users surprise with its easiness of managing resource and better use of resource. Private cloud users can send request similar to the public cloud users. Apart from that, they can almost get what they want immediately. All these actions are handled by cloud system and administrators need not to set up any new machines.

**Virtualization allows physical machine divide resource and plays many roles**

General cloud (Eucalyptus and AWS EC2) heavily use the virtualization technology and run many virtual machine on a single physical machine. Each virtual machine obtain environment with different amount of resource, like Size of RAM, Storage and number...
of virtual CPU cores. The resource allocated can be very small and extremely large.

### 2.2.5 Usage of public cloud

Public cloud usually charge user per user hour with a reasonable price. Within $20 dollar.

Apart from the resource allocated can be adjust, platforms the instance running can be difference also like Windows, Linux. Since the instance is a virtual machine only, some application can be pre-installed inside the image and provide more different type of images.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard On-Demand Instances</strong></td>
<td>Linux/UNIX Usage</td>
<td>Windows Usage</td>
<td></td>
</tr>
<tr>
<td>Small (Default)</td>
<td>$0.085 per hour</td>
<td>$0.12 per hour</td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>$0.34 per hour</td>
<td>$0.48 per hour</td>
<td></td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.68 per hour</td>
<td>$0.96 per hour</td>
<td></td>
</tr>
<tr>
<td><strong>Micro On-Demand Instances</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Micro</td>
<td>$0.02 per hour</td>
<td>$0.03 per hour</td>
<td></td>
</tr>
<tr>
<td><strong>High-Memory On-Demand Instances</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.50 per hour</td>
<td>$0.62 per hour</td>
<td></td>
</tr>
<tr>
<td>Double Extra Large</td>
<td>$1.00 per hour</td>
<td>$1.24 per hour</td>
<td></td>
</tr>
<tr>
<td>Quadruple Extra Large</td>
<td>$2.00 per hour</td>
<td>$2.48 per hour</td>
<td></td>
</tr>
<tr>
<td><strong>High-CPU On-Demand Instances</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>$0.17 per hour</td>
<td>$0.29 per hour</td>
<td></td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.68 per hour</td>
<td>$1.16 per hour</td>
<td></td>
</tr>
<tr>
<td><strong>Cluster Compute Instances</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quadruple Extra Large</td>
<td>$1.60 per hour</td>
<td>N/A*</td>
<td></td>
</tr>
<tr>
<td><strong>Cluster GPU Instances</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quadruple Extra Large</td>
<td>$2.10 per hour</td>
<td>N/A*</td>
<td></td>
</tr>
</tbody>
</table>

*Windows is not currently available for Cluster Compute or Cluster GPU Instances.

Figure 10 the price list of AWS
Actually, cloud is an IT resource hiring service; there are many usage of the cloud. The aim target of using cloud is reducing the cost. General public cloud allows user pay as you go and gets the only service they wanted.
Using public cloud as the only IT support

For example purely using public cloud can reduce or eliminate the cost of manage an IT system. Since company must put many resource on maintenances other than buying the facilities. Cloud providers usually provide many manage tool and some of them are free. These tools can manage the resource using according from usage and adjust number of instance automatically. Also some detailed report will be sent to user and alert the any issue their instances encountered.

Using expensive facilities

Also, cloud can also provide some expensive facilities with a reasonable price. For example, if you need a few high performance for some experiments for a few day. It does not make sense to buy this machine only for this single purpose. Let see Picture 7, AWS provide you with the most powerful machine with GPU supported and changes.

Using cloud to as part of the IT resource

In some situation, Company’s bandwidth and storage may not enough to serve the user over the world. They may use cloud as storage as cloud pool usually has very large bandwidth.

Public cloud is a service

As a conclusion, cloud is a service and it is a business term rather than technical term. It separates the concern between the user and provider. The cloud is very powerful as it uses virtualization and has a giant resource pool. Using cloud can reduce the cost on IT management.

Many people misunderstood cloud

Cloud is a term make many people confuse. It seems to be the way to solve large scale problem and change the instance’s computation power more than his host machine. Actually, a virtual machine
running on a physical machine cannot run faster than the host in most case (Exception found in testing procedure, and discuss in the part of KVM). To make instance more powerful than it, the only way it using distributed system or similar framework like Hadoop. We will introducing the computation power of virtual machine later and also introduce Hadoop which is important part of our project.

2.2.6 Private cloud

Owning a cloud is not the only right of big company. But of course, private cloud is weaker than public because of smaller resource pool.

These are a few open source cloud project and Eucalyptus is the most famous one in Linux environment. In our project, we chose to use Ubuntu Enterprise Cloud (UEC) as it has more support and documentation. Before we introduce Eucalyptus, we will introduce private cloud.

![Image of private cloud](image)

Figure 13 image manage system of our cloud

Private clouds usually provide only virtual machine and storage as the service. Private cloud has features very similar to the
public cloud, private cloud user can get the service similar to the
public user, but with less amount of resource.

Other than users can obtain certain benefits, administrator will
feel cloud is powerful in point of view of management. It reduces the
management cost and better use the IT resource.

**Convenience**

Private cloud usually provides different types of the user
interfaces to the cloud user similar to the public cloud. For the
administrators, there is another interface to manage the whole
system. For example, user management and instance type is
interface we use most in this project.

**A single machine play different role**

Cloud server generally provides the instance in form of virtual
machine. Since we are not always require a very powerful machine.
Cloud server can run many small instances on a machine and
greatly reduce the cost on buying server. Apart from reduce the
number of machines, the IT resource of instance can be dynamically
adjust (reboot required like virtual box) or move to another machines
even public cloud.

**Fast virtual machines set up**

Just imagine you are setting up 100 machines to run
MapReduce, you may set up the machine one by one and repeat the
jobs. It is surely a slow task. In the worst case, if you need to install
new software in each machine like FFmpeg (Encoder we used in
this project), you must use a long time to modify such minor change
in each machine. Using cloud is totally another case. We
implemented a MapReduce program of movies conversion which
required FFmpeg. We only need to modify a single image and
upload to each server. It requires a few of hours only.
In our project, we only use a few hours to set up a Hadoop cluster with cloud, but almost a whole day in physical machine. Virtual machine environment also help us to prevent the technical driver.

**Easy to replace the hardware**

Generally, operation system binds to the physical machine. Replacing hardware usually requires reinstall all operation system, driver, applications and configuration. It surely cost a long time.

In cloud server, the image and the physical machine is separate by the virtualization, so that even though the physical is replaced, the image can still obtain similar running environment. So whenever a hardware facility is replaced, it replaces only the computation power. After installing the virtualization software like KVM and XEN (two technologies widely applied in cloud computing), images can still run in the new server. Since administrators need not to worry about the configuration of image, they can upgrade and replace the system faster and easier.

**Better use of the resource**

Since the cloud is the resource pool shared through network and can be elastically adjusted. Administrator can allocate suitable amount and types of resource to different users according to their need. For example, an application handling text data, can use less memory and CPU cores; an application simulating 3D world can allocation many more powerful instance.
even with GPU support.

A cloud company provides a solution that: every employee use only a very tiny PC and remote connect to the virtual machine. Each staff can have different type of department can have different type of configuration of virtual machine and can get a more powerful machine if necessary. Whenever the system need upgrade, if only upgrade the cloud server and images. The rest of the system keeps unchanged.

Security
Security is surely an important issue. Virus can easily propagate to other program within a server. It can cost a very serious damage to the server requires times to remove the infected file. Apart from virus, some application may crash and used up all the CPU resource. This prevents other program work properly.

Using cloud can separate the application with different virtual machine within one physical host. Since the virtual machine is independent, the infection change can significantly decrease.

A virtual machine can share limited number of virtual core. Even though an application hanged and used up its resource, other machine won’t be affected and work properly.

2.3 From point of views of application structure

2.3.1 Using Cloud Layer

Usually, a cloud provide user with virtual machines. A virtual machine can be independent to others, just same as the servers inside the server room. But how can a cloud be elastic and fit to users’ business?
There are two main approaches, but both of them do it by adjusting the number of virtual machines running but with two totally different mechanisms.
The first approach is giving the different virtual machine with different roles. The typical example is Microsoft Azure. Let take a large scale web application similar to YouTube as an example. The duty of the application are allowing user to upload the video and play the video.

Azure is actually a cloud layer running on a number of windows sever. It contains many technologies like crash handling, and workload distributed system. If a machine crash, the application can still run on other server and the database won't lose.

Azure requires programmers set up one or more roles (type of virtual machines) under some framework in the cloud application. Each role processes different types of job. For example:
a) Web Role process the website query, e.g. Display videos
b) Worker role process the background job, e.g. convert the videos to different format.
c) VM role are used by administrators to manage the application. Such configuration even allows the role can run some large scale process in more than a virtual machine parallel.
The workload of each virtual machine is balanced by a Load Balancer. No of the virtual machines of each node can be adjust depending on the situation. For example, only a few users upload the video at mid-night, so, fewer virtual machines of worker role are used. This greatly reduces the cost of hiring the cloud service.

This approach cloud user to adjust the resource fit to the actual need. However, Azure require programmer to modify some features of the applications. In the worst case, it is necessary to re-
develop the whole application. And the applications usually can’t run on platform other than Azure.

2.3.2 Using Cloud instance with other framework

Another approach is migrating application directly into the cloud with form of virtual machine. Administrator is managing the system by managing the virtual machines. The typical example is AWS EC2. Cloud can run different number of machine on different image similar to the previous approach. However, AWS is providing only the instance but not a cloud layer for each user. If a server crash, no handler will take up the jobs. This approach requires the application has elastic property and handle the crash of the system. In order to has the property of elastic and handling crash of the system automatically. Some other technology like distributed system or MapReduce.

![Eucalyptus](image)

Figure 17 Azure is similar to instance with Hadoop technology, Hadoop is the MapReduce application framework) as a layer to connect all machine.

2.4 Summary of concept of Cloud

Private cloud is a way to manage the system, private cloud is a business term rather than technical term. Cloud has feature of elastic and virtualization. The most important benefit of cloud is reducing cost, but not it powerfulness. The power of the cloud is
upper limited by the size of the resource and heavily relies on the
network condition.

2.5 Eucalyptus (UEC)

2.5.1 Principal of eucalyptus cloud

There is different cloud architecture
and in general, a cloud will not provide
physical machine but virtual machine or an
application service to the users. We are
introducing the Eucalyptus which is an open
source project for private cloud and public
cloud.

Eucalyptus has architecture similar to Amazon Elastic
Compute Cloud (EC2). Actually, the image of eucalyptus can directly
run on EC2 and the commands of these two systems is almost the
same?

Eucalyptus highly relies on the Virtualization technology. The
cloud usually provides a virtual machine to the user and a physical
machine can run more than one virtual machine (instance). Users
can run self-created images (operation system with self provided
application) or provided default images. The architecture has many
advantages:
A physical machine can play the rules of different operation systems,
and less physical machines are needed in the network. This greatly
reduces the budget of computer system set up.

All application can be independent and be more secure. Each
virtual machine has its resource bound. Even though an application
used up its resource in a virtual machine or crashed, it will not affect
other virtual machine. Since all application is separated, it can help
preventing attack from others and virus spreading.
Each virtual machine’s resource can be adjusted. Some virtual machines can obtain more CPU cores, memory. If necessary, the virtual machine can be moved to other physical servers and use up almost all the computation power of it.

Whenever a user suddenly wants more virtual machines, he can directly add them first before buying more machines.

Greatly reduce the system upgrade effort. Since the virtual machine is actually an image file, so if the physical machine is updated, the system can be immigra by copying the images files. Administrator need not to reinstall each virtual machine once.

Here is the basic architecture of the Eucalyptus, there are five components:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Cloud Controller</td>
</tr>
<tr>
<td></td>
<td>Any request from user are receipted and handled by this part. All instances are delivered from it.</td>
</tr>
<tr>
<td>2.</td>
<td>Walrus</td>
</tr>
<tr>
<td></td>
<td>However, it delivers the images to the Cloud Controller instead of the Node Controller. All images are compressed and encrypted.</td>
</tr>
<tr>
<td>3.</td>
<td>Cluster Controller</td>
</tr>
<tr>
<td></td>
<td>Since the cloud is large, Cluster Controller help share the workload of monitor.</td>
</tr>
<tr>
<td>4.</td>
<td>Storage Controller</td>
</tr>
<tr>
<td></td>
<td>Storing all disk images, kernel images, ram disk images and User storage here. This is the additional part of Eucalyptus. When an instance is terminated, Eucalyptus will not save the data inside, it need an external component to save these data.</td>
</tr>
<tr>
<td>5.</td>
<td>Node Controller</td>
</tr>
<tr>
<td></td>
<td>It generates the instances with Virtualization technology like KVM, XEN. (In Ubuntu Enterprise Cloud, it support only KVM)</td>
</tr>
</tbody>
</table>
2.5.2 Our private cloud

We built our first private cloud with 4 old machines, the hardware equipment are shown in previous discussion. The most important requirement is having some CPU which supports VT-x.

After building a cloud, we built many images to test the actually performance of the cloud. Each image requires three file: kernel, ramdisk, and hard disk image. Each of these file has an image ID.
This is the status of an instance. There are seven important attributes:
<table>
<thead>
<tr>
<th><strong>Instance ID</strong></th>
<th>This represent the ID of the virtual machine running, it is random generate by default.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image ID</strong></td>
<td>The original virtual uploaded to the cloud</td>
</tr>
<tr>
<td><strong>Public IP</strong></td>
<td>Use this IP to connect this virtual machine. This IP is bind to the cloud controller. The network traffic is first going to the cloud controller and than the host (node), so the bandwidth is bounded by the bandwidth of cloud controller.</td>
</tr>
<tr>
<td><strong>Subnet IP</strong></td>
<td>The virtual machine within the same cloud can communicate within this IP. This is a virtual IP bind to the host. So it is only bounded by the host.</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>It show three status:</td>
</tr>
<tr>
<td></td>
<td>Pending : preparing the instance</td>
</tr>
<tr>
<td></td>
<td>Running: The boot signal is sent to kvm</td>
</tr>
<tr>
<td></td>
<td>Terminated: the instance is killed</td>
</tr>
<tr>
<td><strong>The IP of host</strong></td>
<td>Using this IP to get the display of the instance</td>
</tr>
<tr>
<td><strong>Type of instance</strong></td>
<td>It refer to the configuration of the instance, e.g. no of virtual CPUs.</td>
</tr>
</tbody>
</table>

The instance is similar to the virtual machine in virtual and VMware. It can get the internet service like real machine. But instance only know his subnet IP but no idea of the public IP.

We can view the display of an instance by making a connection to the host with VNC viewer.
### 2.5.3 Technical trouble shoot

#### 2.5.3.1 Image fails to run a image

After the cloud is set up, we found the cloud fail to run any images even the default images. It shows pending and then terminated with not error message displayed. The reason is that the CPUs do not support VT-x and KVM. There are two ways to solve this problem: use CentOS and run xen instead of KVM; using the machine with VT-x support.

At the end, department give us three machines, so that we can install the cloud and run it. So we don’t need to learn to set up cloud in a new environment.
2.5.3.2 Fail to add node server

After installed all required OS in Cluster Controller and nodes. The node can't discover and add any nodes. Only a few discussion on this problem, and we found this is a bug of Eucalyptus 1.6.

Eucalyptus doesn't release any guideline in handling this problem, but it can be fixed with a few simple steps by resting the cloud.
1) Clear all the clusters and nodes in Cluster Controller.
2) Reboot all machines
3) Clear all the .ssh/Known host record
4) Let Cluster Controller discover the nodes again.

2.5.3.3 Fail to de-registered

This is a bug similar to previous problem and can be fixed with the same way to handle previous bug.

2.5.3.4 Instance IPs problem

Whenever any machine's IP changed, the cluster won't work properly.

If Cluster Controller's IP changed, it can execute any "euca-*" command, all the setting of this cluster need to be rested at http://localhost:8443 and the update the credentials.

If Node's IP changed, it will be more complicated. Cluster Controllers may not find the node. The Cluster Controller will not add any other node which is using this IP, and the node with changed IP will be used and the certificate in the Cluster Controller doesn't accept the new IP. Cluster Controllers may give you some warning about some attacks. To fix this problem, you can let Cluster Controllers forget the old nodes by remove the "known hosts" file in Cluster Controller's

//var/lib/eucalyptus/.ssh (.ssh is a hidden folder)

The best way to handle this problem is bind the IPs with MAC address.
2.5.3.5 Connection fail to Instance
A key is required to access the Ubuntu instance and ssh client wills memory the IPs and public keys. If a key is used in different IPs, it will be treated as an attack from others.
This problem can be fixed by generate more keys for different IP or remove the record of the SSH client by remove the “known_hosts” file in ~/.ssh (.ssh is a hidden folder).

2.5.3.6 Slow start up speed Problem
This is mainly the problem of the network bandwidth and the size of the images. We can only build small images (4GB) and replace the cable to CAT-5E and reduce up the network transfer time.

2.5.3.7 Large image file problem
With the limitation of eucalyptus, how large the virtual machine, how large is the image file size. When we want to set up a virtual machine with 40GB storage, it starts up with a very long time.
We found KVM support smaller image file (Only 1/10 the original file size), but eucalyptus doesn’t support.
We did a lot of testing images and found eucalyptus has disabled this function.

2.5.4 Branch mark of our private cloud
To fully understand how cloud work and performance of the cloud. We did a series of testing on both physical machines and virtual machines.
2.5.4.1 Startup time

Startup time of the virtual machines is heavily depended on the network status. Before running a virtual machine, the cloud runs a few steps:

1. Check availability of the cloud to check if there is enough resource.
2. Decrypt and decompress the image
3. At the same time, send the image to one of the node.
4. The node will save the image to the cache
5. And copy a new copy and run the virtual machine. The virtualization used

<table>
<thead>
<tr>
<th>Size of the image</th>
<th>First start up time $^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GB</td>
<td>~5mins</td>
</tr>
<tr>
<td>5GB</td>
<td>~8mins</td>
</tr>
<tr>
<td>10GB</td>
<td>~17mins</td>
</tr>
</tbody>
</table>

Figure 23 Start up time of different instance

When an image is sent to the node server, it will copy will be save it node has sufficient space. Next time if the same image runs on it, it can directly copy the image and save the bandwidth. Second start up requires 1/3 of the first start up time only.

Startup of AWS images require two minutes only.

2.5.4.2 Overall performance

The overall performance of our private cloud (after optimization) is about 73% of the physical machine. It is similar to the value that eucalyptus discussed. The most important factor slowing down the performance is the IO performance of the virtual machine.

$^1$ It heavily rely on network status and it is an average time.
2.5.4.3 IO performance

The IO time of the virtual machines is 1.47 time of the host (physical machine). The disk write speed is only 50.9MB/2s; it is just half of the physical machine’s 103MB/s. If two process is writing a hard disk at the same time, virtual machine can write data with speed 26MB/s, (65MB/s in physical machine). We can see write time of the virtual machine can reach to about 50% of the physical machines in our test case. The read disk speed is about 80% of the Physical machine.

The virtual machine can usually share ~50 % (20MB/s) bandwidth of the physical host (~40MB/s). It is fast enough for out project.

Some related report of hard disk branch mark show that virtual machine can reach ~90% performance of the physical machines. It seem to be the problem of configuration or the facilities' configuration.

```bash
#!/bin/bash
# init
function pause()
{
    read -p "$*"
}
pause "1).Download Test"
time wget http://www.cse.cuhk.edu.hk/~csci3150/mp3/20100913_csci3150.mp3
pause "2).CompressFileTest"
time tar -zcvf file.tar.gz 20100913_csci3150.mp3
rm -f 20100913_csci3150.mp3
pause "3).DecompressFileTest"
time tar -zxvf file.tar.gz
rm -f file.tar.gz
pause "4).copyfile"
time cp 20100913_csci3150.mp3 20100913_csci3150_2.mp3
rm -f 20100913_csci3150.mp3 20100913_csci3150_2.mp3
pause "5).Write500MBFile"
time dd if=/dev/zero bs=1M count=500 of=/500MB.file
pause "6).CompressFileTest2"
time tar -zcvf file.tar.gz 500MB.file
pause "7).DecompressFileTest2"
rm -f 500MB.file
time tar -zxvf file.tar.gz
rm -f 500MB.file file.tar.gz
pause "Finished."
```

Figure 24 Our testing script

---

2 1MB/s = 1MByte/s
2.5.4.4 Computation performance

We use CPU system time and user time to measure actually performance of the instance. Over all system time of virtual machine is 1.512 x of the host (66% performance of host). The user time of instance is 1.11 x of the host (89.6% performance of host). To fully understand the actually performance of the CPU, we implemented a simple multi-threads programs which use 100% user time only. We find self-define function call can obtain 97% performance of the host machine. This really gives us surprise and guides us to implement the application with more user time than system time.

Another testing is running two identical programs in the same host but different instance. The CPU time is almost the same.

<table>
<thead>
<tr>
<th>Over all performance of virtual machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host machines</td>
</tr>
<tr>
<td>Real Time</td>
</tr>
<tr>
<td>User Time</td>
</tr>
<tr>
<td>Sys Time</td>
</tr>
<tr>
<td>I/O time</td>
</tr>
</tbody>
</table>

![Figure 25 Running time of instance and host](image)

2.5.4.5 Memory Access time

Another testing of accessing large amount of memory shows that the memory access time of the Virtual machine is relatively slower than the host. It has only 76% of the Host machine. We can
see that even VT-x and Virtio( will be discuss later) has improved the performance of virtual machine, virtual machine still can not directly access the physical memory.

```c
int nu = __noOfLoop;
int mSize=100*MB;
int i,j;
char * list=malloc(sizeof(char)*mSize);
for(i=0;i<nu;i++)
    for(j=0;j<mSize;j++)
        list[j]++;
free(list);
return NULL;
```

Figure 27 our memory testing program

---

**Memory access speed test**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Time(s)</td>
<td>16.070</td>
</tr>
<tr>
<td>Virtual machine time(s)</td>
<td>20.977</td>
</tr>
<tr>
<td>Ratio(host / Virtual)</td>
<td>0.7660771</td>
</tr>
</tbody>
</table>

Figure 28 Result of our testing on memory access

---

### 2.5.4.6 Multi-platform

An important feature of cloud is supporting Linux and Windows running on it. After the cloud is set up, we tried to build our own image instead of the default image provided by eucalyptus.

Surely, setting up a Linux Image is easy and same as installing it on a physical machine. The performance of the self-built images is faster than the default image and support GUI Desktop version of cloud. There is a virtual monitor for each virtual machine.

We can install and setup the windows into an image, however, it show kernel panic when the image is upload up the cloud.
2.5.4.7 KVM and XEN

KVM and xen are the most common open source virtualization technology used in Eucalyptus and AWS EC2. Both of them are using the same image format and compatible to each other. However, their hardware require are different. KVM requires host CPU support Intel’s VT-x or AMD-v. These two technologies are similar and allow the code of a virtual machine can directly run on host’s CPU without many change. Xen and kvm both run very fast and kvm has better performance.
Ubuntu Enterprise Cloud doesn’t support xen after Ubuntu 9.04 even eucalyptus still support it.

2.5.4.8 Exception in running speed

Virtual machine running slower than host seems to be trivial, but not always true, in some test case, if the host is very powerful and guest (virtual machine) is running some tedious jobs and heavily rely on the I/O, virtual machine can run faster than host. The reason is that virtual environment may cache up the hard disk read/write and greatly reduce the I/O time of the virtual machine.

![Figure 33: An instance runs faster than its host](image)

2.5.4.9 Virtio

Virtualization generally slows down the overall performance of the virtual machine. There are two main technologies to optimize the running speed. The first one is Intel’s VT-x or AMD-v. These CPU technologies allow virtual machine run the code directly in the Physical CPU. In previous discussion, we introduce the virtual machine can has almost the same user time of host. Another technology is Virtio\(^3\), it allow the virtual machine directly read/write on the Physical machine.

\(^3\) Eucalyptus default turn Virtio off
After turning the Virtio on, Overall IO time reduce to 1/4 of the original. It greatly increases the write disk speed from 13MB/s to 50MB/s and bandwidth from 10MB/s to 20MB/s. The read disk speed is similar to the physical machine.

2.5.4.10 Limitation and weakness of Eucalyptus

Although eucalyptus is a pretty good open source cloud project with many powerful feature. It has many limitations on it.

The instance won’t be save after termination

Eucalyptus allows a user to copy an image to many instances. You can reboot the instance and the data in the hard disk still exist. However, if you terminated the instance, all the data in virtual machines will be freed. So before terminating an instance, user must upload the data to cloud storage like Walrus. Another approach is implementing the application only use the central storage. But this approach requires a very large bandwidth of the local network and may be charged a lot on public cloud. Getting worse, the poor IO performance many reduce the overall performance of the whole system.

Poor instance start up mechanism
The design of Eucalyptus’s instance start up mechanism is quit silly. The images uploaded up the cloud will be encrypted compressed and cut into many blocks. The decrypting and decompressing steps are process in the cloud controller before sending the images out. Whenever many instances (e.g. 10) are pending at the same time, the workload of the control is huge and limited to the bandwidth of the controller machine. This design can actually be improved greatly by forming a peer to peer network within the nodes. The blocks of the images can be cache in the node if it was running on it. When a machine need to start up that image, just get the blocks of this images form all others having it and decompress on its own. This can greatly reduce the time to start up images and better use the bandwidth.

Large image file problem

The size of an image file is identical to the capacity of the virtual machine. That mean if I a virtual machine has a 10GB virtual hard disk, even 90% of the space is not been used. Getting worse, this 10GB image must be sent to the node through local network and used up most of the bandwidth.

Figure 35 A command to create a small size image

```
uecadmin@client1:~$ kvm-img create -f qcow2 image.img 5G
```

The menu page of Kvm show that the kvm can create a special image format similar to the format using is Virtual Box and VMware. This format can free the space out space of the holes (large space not being used) in the images. We did many testing on this format and built some image and found that image support 10GB storage requires only 1GB disk space. But eucalyptus doesn’t support this format and return error message of hard disk no found.
In our project, virtual machines play the roles of storage and computation and have size of tens of GBs. This surely gives our tiny private a great challenge.

**Sensitive to the network**

Since the cloud always transfer the images to nodes. It gives the local network a great challenge. Apart from that, if the cloud controller crash, all public IPs of the instance will become invalid and require reboot of all instance.

**Non-static IP**

Since the number of IPs allocated to the instance is limited. The cloud will reuse the IPs.

### 2.5.5 Difficulties in setting up private cloud

Setting up a cloud actually require only a few

### 2.5.6 Cloud Summary

Cloud is actually a very powerful technology. The most significant feature of cloud is elastic and reduces cost. It is not a tool to implement faster application since virtual machines seldom run faster than his host. But virtualization makes manage and share resource to different possible and allocate resource fit to any situation.

However, we have shows some limitation of typical. There are many trades off and silly design in cloud technology, and the performance of the cloud is limited to the power of the hosts and the network stability.
Chapter 3: Apache Hadoop

3.1 Introduction to Hadoop

In 1965, Gordon E. Moore claim that transistor counts had doubled every year. Recent years, the technology has reached the limit of what is possible with one CPU, that's why parallel computing comes, due to the convenience of management, could computing architecture and the most widely-used, open source framework Hadoop comes out.

Apache Hadoop is a software framework that supports data-intensive distributed applications; it enables applications to work with thousands of nodes and petabytes of data.

Hadoop contains

![Figure 36 The computation power of machines has a bound](image)

![Figure 37 Hadoop family](image)
1. The projects Hadoop Common: the common utilities that support the other Hadoop subprojects.
2. Hadoop distribute file system (HDFS).
3. Hadoop MapReduce: a software framework for distributed processing of large data sets on compute clusters.
5. Chukwa: A data collection system for managing large distributed systems.
6. HBase: A scalable, distributed database that supports structured data storage for large tables.
7. Hive: A data warehouse infrastructure that provides data summarization and ad hoc querying.

In our project, we mainly focus on HDFS, MapReduce and Hive.

Who is using Hadoop?

![Facebook](facebook.com)  [![Yahoo](yahoo.com)](yahoo.com)  [The New York Times](nytimes.com)  [Adobe](adobe.com)  [Amazon](amazon.com)  [Hulu](hulu.com)  [Baidu](baidu.com)  [AOL](aol.com)  [Microsoft](microsoft.com)
Hadoop benchmark
In year 2009, Hadoop sorts 100 TB in 173 minutes (0.578 TB/min) on 3452 nodes x (2 Quad core Xeons, 8 GB memory, 4 SATA)

HDFS
Hadoop Distributed File System (HDFS) is an open source storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute nodes throughout a cluster to enable reliable, extremely rapid computations; it is designed in fault tolerance.

MapReduce
MapReduce is a programming model and software framework introduced by Google to support distributed computing on large data sets on clusters of computers. Hadoop MapReduce is an open source project from Apache, it allow us develop parallel applications without any parallel programming techniques, and the applications could be easy deployed and executed, it works with the HDFS and processes huge datasets (e.g. more than 1TB) on distributed clusters (thousands of CPUs) in parallel. It allow programmer writing the code easily and fast.

Working Environment:
Currently the newest version of Hadoop is 0.21 which is not well tested, it may get problems when starting the HDFS service, so, we choose the version 0.20, it was supported by Linux environment and tested with 2000 nodes, theoretically speaking, it also works in windows, but rare people tried it. As it is Java based system, so, get Java installed is a must.
3.2 HDFS

The average of bandwidth of the world is increasing, lots of service provider needs more and more storage, such as Facebook, in 2009, 4TB compressed new data was generated to their storage each day, 135TB compressed data scanned each day, parallel computing is the choose, this also works for many years, but the needs is still growing, they needs more and more machines, hard disks, more than 1000 machines working together is not special now. So, the other big problem appears: hardware failure, a report from Google claims that the hard drive gets 7% failure rate per year, so, in average, they have hard drive failure every day. One of the possible solutions is choosing higher level hard disks which get smaller failure rate, but the cost is extremely expensive. HDFS is designed to handle this large scale data read and write and hard drive failure, data stored in different machines maybe 3 or more copies, if hard drive failure detected, the rebalance will do the work to keep the file completed.

3.2.1 HDFS Structure

3.2.1.1 Namenode

The Namenode server in HDFS is the most important part, it store all the metadata of the stored files and directories, such as the list of files, list of blocks for each file, list of DataNodes for each block and also File attributes. The other role is to serve the client queries, it allows clients to add/copy/move/delete a file, it will records the actions into a transaction log. For the performance, it save the whole file structure tree in RAM and hard drive. A HDFS only allow one running namnode, that's why it is a single point of
failure, if the namenode failed or goes down, the whole file system will goes offline too. So, for the namenode machine, we need to take special cares on it, such as adding more RAM to it, this will increase the file system capacity, and do not make it as DataNode, JobTracker and other optional roles.

Figure 38 Namenode monitoring

3.2.1.2 SecondaryNamede

From the name, it may like the active-standby node of the NameNode, in fact, it is not. What it works is to backup the metadata and store it to the hard disk, this may helping to reduce the restarting time of NameNode. In HDFS, the recent actions on HDFS will be stored in to a file called EditLog on the NameNode, after restarting HDFS; the NameNode will replay according to the Editlog. SecondaryNameNode will periodically combines the content of EditLog into a checkpoint and clear the Editlog File, after that, the NameNode will replay start from the latest checkpoint, the restarting time of NameNode will be reduced.

SecondaryNamede was running on the same machine as NameNode in default which is not a good idea, it is because if the NameNode machine crashed, it will hard to restore. Placing the SecondaryNamede on the other machine, it may help the new NameNode to restore the file structure.
3.2.1.3 Datanode

Datanode is the scaled up part of HDFS, it may have thousands, it mainly use to store the file data. On startup, DataNode will connect to the NameNode and get ready to respond to the operations from NameNode. After the NameNode telling the position of a file to the client, the client will directly talk to the DataNode to access the files. DataNodes could also talk to each other when they replicating data. The DataNode will also periodically send a report of all existing blocks to the NameNode and validates the data block checksums (CRC32).

Figure 39 Simple structure of HDFS

3.2.1.4 Other Features

HeartBeats

DataNodes send heartbeat to the NameNode, and NameNode used heartbeats to detect DataNode failure. If failure detected, the NameNode will choose New DataNodes for new replicas to balances disk usage and communication traffic to DataNodes.

Rebalancer

The rebalance preferred similar percentage of disk usage of each DataNode. It works when DataNodes are added or removed. It is throttled to avoid network congestion.
Block Placement

Any single file stored in Hadoop will be stored in different machines, for the metadata, it will be stored on NameNode, the file content will be split to data blocks and store each block to different DataNodes, the block side is 64MB in default, and it will has 3 replicas, the first one stored on local node, the 2nd and 3rd replicas are stored on remote nodes, for the other replicates, it will be placed randomly. When the client read a file, it will be directed to the nearest replicate.

Data Correctness

In HDFS, if use CRC32 to do the checksum validation, if validation failed, the client will try the other replicas.

Data Pipelining

When the client want to write a file to the file system, he will retrieve a list of DataNodes on which to place replicas of a block and writes block to the first DataNode, the first DataNode forwards the data to the next DataNode in the Pipeline
When all replicas are written, the client moves on to the next block of the file.

3.2.1.5 Compare with NFS

Network File System(NFS) is one of the famous DFS, the design is straightforward, it provides remote access to single logical volume stored on a single machine, but there has limitations such as the files in an NFS volume all reside on a single machine, lots of information as can be stored in one machine, and does not provide any reliability guarantees if that machine goes down, another limitation is that all the data is stored on a single machine, all the clients must go to this machine to retrieve their data. This can overload the server if a large number of clients must be handled. HDFS solve the problems.
HDFS is designed to store a very large amount of information (terabytes or petabytes). This requires spreading the data across a large number of machines. It also supports much larger file sizes than NFS.

HDFS should store data reliably. If individual machines in the cluster malfunction, data should still be available.

HDFS should provide fast, scalable access to this information. It should be possible to serve a larger number of clients by simply adding more machines to the cluster.

HDFS should integrate well with Hadoop MapReduce, allowing data to be read and computed upon locally when possible.

Applications that use HDFS are assumed to perform long sequential streaming reads from files. HDFS is optimized to provide streaming read performance; this comes at the expense of random seek times to arbitrary positions in files.

Data will be written to the HDFS once and then read several times; updates to files after they have already been closed are not supported. (An extension to Hadoop will provide support for appending new data to the ends of files; it is scheduled to be included in Hadoop 0.19 but is not available yet.)

Due to the large size of files, and the sequential nature of reads, the system does not provide a mechanism for local caching of data. The overhead of caching is great enough that data should simply be re-read from HDFS source.

Individual machines are assumed to fail on a frequent basis, both permanently and intermittently. The cluster must be able to withstand the complete failure of several machines, possibly many happening at the same time (e.g., if a rack fails all together). While performance may degrade proportional to the number of machines lost, the system as a whole should not become overly slow, nor should information be lost. Data replication strategies combat this problem.
3.2.1.6 Limitation of HDFS

The latest version of Hadoop is just 0.21, so there exists some limitations, for the NameNode, it is a single point of failure and it allows single Namespace for entire cluster, the development team claims that they are working on work on Backup Node to solve this problem. It's also does not rebalance based on access patterns or load, the average performance will be reduced. It's designed in Write-once-read-many access model, then the client can only append to existing files, no general file modify operation.

3.3 MapReduce

3.3.1 Machine Roles

There has 2 roles for MapReduce jobs, one is JobTracker and TaskTracker.

**JobTracker**

The JobTracker is used to farm out the MapReduce tasks to the clusters, it will allocate the task to the cluster which have the data that the task needed in highest priority to maximize the performance. When the client submit a job to the JobTracker, it will talks to the NameNode to find the data location, then it could determine which node is the nearest to the data and submits the task to the chosen TaskTracker nodes. When the TaskTracker finished the task, JobTracker will get the signal and assigns another task to it. It runs in the NameNode machine in default.

**TaskTracker**

TaskTracker just like the DataNode, it will be running in every clusters, it received the task from JobTracker and mainly do the Map, Reduce, Shuffle operations.

Every TaskTracker has a set of slots, each slot accept a task. When the JobTracker tries to find TaskTackers to run a MapReduce task, it first looks for an empty slot on the same server that hosts the
DataNode containing the data, and if not, it looks for an empty slot on a machine in the same rack. The TaskTracker runs the actual work in separate JVM process, if the task crashed, it will not take down the TaskTracker. When the task is running, TaskTracker may monitoring these processes and store the output and return value. It also send heartbeat signals to JobTracker and let the JobTracker knows that it was alive, the heartbeat signal also including the information such as number of empty slots. After finishing the task, it sends a signal to notify the JobTracker.

### 3.3.2 Programming Model

MapReduce is an easy programming model for computer or non-computer scientists. The whole structure of a MapReduce program is mainly work by 2 functions, the Mapper and the Reducer. One key-value pair as a input for the Mapper, and the Map functions run in parallel, generates any pairs of intermediate key-value pairs from different input data sets, and these will be sent to the Reduce function as input, and the reducer will sort the input key-value pairs according to the key and combine the value of key-value pairs into a list which have the same key, at last, the reducers which are also run in parallel generates the output key-value pairs. All the tasks are working independently.

### 3.3.3 Execution flow

The user program splits the data source into small ones; normally it will be split into 16~64MB blocks, and copy the program to the selected JobTrackers to get ready for processing data.

The JobTracker distributes the job to JobTaskers.

The Mapper Read the records from the data source such as lines of text files, data row from database, list of text as file path and
so on. After mapping process, it sends the intermediate key-value pairs to reducers. Prototype of Map function:

\[
\text{Map}(\text{in\_key}, \text{in\_value}) \rightarrow \text{list}(\text{out\_key}, \text{intermediate\_value})
\]

The Reducer received the key-value pairs from mappers, it process on the key-value pairs and finally it generates the output key-value pairs

Prototype of Reduce function

\[
\text{reduce}(\text{out\_key}, \text{list(intermediate\_value)}) \rightarrow \text{list(out\_value)}
\]

The full map of MapReduce flow

Figure 40 Output of a running Hadoop program

Figure 41 Output of Hadoop
3.3.4 Anatomy of a Job

1. Client creates a job, configures it, and submits it to job tracker
2. JobClient computes input splits (on client end)
3. Job data (jar, configuration XML) are sent to JobTracker
4. JobTracker puts job data in shared location, enquires tasks
5. TaskTrackers poll for tasks
Figure 42 Coding flow
3.3.5 Features

Automatic parallelization and distribution

When we run our application, the JobTracker will automatically handle all the messy things, distribute tasks, failure handling, report progress.

Fault-tolerance

The TaskTracker nodes are monitored. If they do not submit heartbeat signals in a period of time, they are deemed to have failed and the work is scheduled on a different TaskTracker. If a task failed for 4 times (in default), the whole job will failed.

A TaskTracker will notify the JobTracker when a task fails. The JobTracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the TaskTracker as unreliable.

Figure 43 Fault tolerance
Locality optimization

He JobTracker will assign the splits which have the same keys to nearest clusters; this may reduce the job distribution time in Map procedure and collection the output key-value pairs in Reduce procedure.

Backup Tasks

In MapReduce, some operations may be come straggler and increase the total running time, one of the most important reason is that some of the Mapper/Reducer will fight for the local resources such as CPU, Memory, local disk, and network bandwidth, etc. these may increase the delay latency. A simple solution in Hadoop is to have more than 1 copies of the Map/Reduce job running in different machines if there has some empty resources/slots. If a copy finished the job, the others will be killed. The overhead of this procedure is small but the total time reduced significantly in large scale clusters operations.

3.3.6 Our testing

JobTracker is a single point of failure for MapReduce service, if it crashed, all the jobs will be failed.

Hadoop accounts are not isolated; they share the same file pool and resources.

Our Pilot programs

We want to have our own MapReduce application, but not just word count (the example), so we implement all the basic interface of the MapReduce framework: InputFormat, InputSplit, RecordReader, Writable, RecordWriter, OutputFormat, Mapper and Reducer. Base on these interfaces, we could write any MapReduce programs easily.
### 3.3.6.1 Image processing

A program that runs edge detection algorithm on images, we built it on MapReduce by Java, the input parameters are: [input file directory on HDFS] [output path on HDFS]

![Output of image processing testing](image)

**Working flow**

The program will get the file list from input directory, and create a split array by treating each file path as a split.

1. Treat each file path as a key, file content as value.
2. Map function received the key-value, runs the edge detection algorithm on it, file path as the intermediate key, and processed image content as intermediate value.
3. Reducer received the key-value pair and save the value to HDFS as output.

**Test result**

<table>
<thead>
<tr>
<th># of node</th>
<th>min</th>
<th>sec</th>
<th>total sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>8</td>
<td>608</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>54</td>
<td>354</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>14</td>
<td>254</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>48</td>
<td>228</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>26</td>
<td>206</td>
</tr>
</tbody>
</table>
As the mapper and reducer are both placed in a single machine, so, if the mapper finished its tasks, the reducer may speed up. This will be significant in the small scale of clusters (not enough nodes).

3.3.6.2 Hadoop Word count example

**Description**

Find out all the words in the text files, and count how many times it appears.

Test on 10 * 64MB text files.
Running environment:
private cloud environment, from 1 to 5 instance

instance specification:
OS: Ubuntu 10.04
RAM: 370MB
CPU: Single virtual core of Pentium D

Program prototype

Map (String docid, String text):
for each word w in text:
    Emit(w, 1);

Reduce(String term, Iterator<Int> values):
int sum = 0;
for each v in values:
    sum += v;
Emit(term, value);

Test result

<table>
<thead>
<tr>
<th># of node</th>
<th>min</th>
<th>sec</th>
<th>total sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>56</td>
<td>1076</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>12</td>
<td>552</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>25</td>
<td>385</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>306</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>13</td>
<td>253</td>
</tr>
</tbody>
</table>
From the graph we see that speed is almost linear proportional to the number of nodes. This program tests on the I/O of HDFS, Job splitting, Shuffling and Sorting of Hadoop, the result shows that Hadoop is good on data processing.

### 3.3.6.3 MD5 Generator

**Description**

This program is used to calculate the MD5 checksum of 1 to n digits numbers, the result may be stored to HDFS, and after that, we could checking the MD5 mapping in few seconds by Hive. The bottleneck of this program is on CPU and uploading data to HDFS.

Calculate the md5 checksum of numbers from 1 to 10^7

**Running environment:**

From 1 to 5 machines

**Machine specification:**

OS: Ubuntu 10.04

![Figure 45 Output of MD5 testing](image-url)
RAM: 1GB
CPU: Pentium 4

**Working flow**

The program needs 3 parameters, start_from max_number and output directory.

For each split, it contains 2 values, the offset and length. Offset is the key. We set the length for all the splits to 10000. In map function, it will loop from offset to offset+length to calculate the MD5 checksum, for each iteration, it add an intermediate key-value pair: `<number, md5 checksum>`

In reduce function, it collects all the intermediate key-value pair and stored it to HDFS.

**Test result**

<table>
<thead>
<tr>
<th># of node</th>
<th>min</th>
<th>sec</th>
<th>total sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>59</td>
<td>419</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>14</td>
<td>374</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>34</td>
<td>274</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>42</td>
<td>282</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>55</td>
<td>235</td>
</tr>
</tbody>
</table>

This program mainly tests on file appending on HDFS and computation performance of TaskTrackers. From this graph, we have a
unexpected result, 3 nodes is faster than 4 nodes, we tried to run the test on 3 and 4 nodes more times, but the result is similar.

### 3.3.6.3 Video Conversion

**Description**

This program aims to convert the video files in parallel.

Convert 30 x 50MB mp4 files to flv files.

**Running environment:**

From 3 to 5 machines

Machine specification:
- OS: Ubuntu 10.01
- RAM: 1GB
- CPU: Pentium 4

**Working flow**

It takes 2 parameters: `input_dir` as the input directory in HDFS, `output_dir` as the output path.

For each split, it contains 1 value, the file path.

In map function, it will download the video file from HDFS according to the key value, then, it uses the shell to call the FFmpeg to convert the video file to any format, after finished the conversion, and it uploads the converted file to HDFS. It will not return any output key-value pairs.

In reduce function, it will do nothing.

**Test result**

<table>
<thead>
<tr>
<th># of node</th>
<th>min</th>
<th>sec</th>
<th>total sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>13</td>
<td>25</td>
<td>805</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>41</td>
<td>521</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>22</td>
<td>382</td>
</tr>
</tbody>
</table>
In this program, it only uses the Mappers and also I/O of HDFS. The result graph is nice, the speed is also linear increasing with the number of node, and the programming is easy and flexible.

### 3.4 Trouble shoot

Failed to start the NameNode: Installing Hadoop is relative simple, but there may have some problems, after setting up the environment, we need to specify the Java installation path by setting the parameter JAVA_HOME in $Hadoop\_path/conf/hadoop-env.sh, otherwise, the HDFS will get error when it try to boot up.

No TaskTracker available: the NameNode runs normally and also exist some DataNode, but no TaskTracker exist, that's because if HDFS is not ready, the JobTracker will no accept any TaskTracker to join, HDFS will be in safe mode when it start, it will not allow any modifications to the file system, if the number of DataNode is not enough, it will report error and state in safe mode. It may return "The ratio of reported blocks 0.9982 has not reached the threshold 0.9990. Safe mode will be turned off automatically." Turn on enough DataNode to reach the threshold will solve this problem, if some of DataNode were crashed and the available ratio is not enough, the only choice is to re-format the whole HDFS. So, less than 3 DataNodes is dangerous.
MapReduce job always 0 Map input and 0 reduce input: in the RecordReader, the "next" function should return true although there has no next value, we should use a local variable as a flag to check if any other value pair exists.

In map function, not HDFS instance: We could access the HDFS by using the shell to call the HDFS API directly.

Heap Space Error: Memory on TaskTracker is not enough, in default, the heap size of JVM is 128MB, so, we could set it in client programs: conf.set("mapred.child.java.opts","-Xmx1024m"); Lack of memory may cause a lots of problems, such as the task may always failed and the whole job failed, or too much content swapping, the performance will extremely low.

Figure 46 error message of our application

3.5 Hadoop summary

For those test cases, the scalability of Hadoop is very significant, most of them show the linear increasing trend, we just test on 5 machines, if the number of clusters is large enough, and it will be very powerful for any jobs. On the performance aspect, it is not the same as we expected at beginning, for empty jobs, it may also takes 30 to 60 seconds to run. The overhead of Hadoop on small jobs is very significant, that’s why Hadoop is for large scale, data intensive jobs, otherwise, MPI will be the better choice.

Hadoop is also not good for frequently I/O access, the MD5generator shows that the performance is much lower than
running on a machine locally, that’s because frequently appending data to a file is very slow. For the word count example, we are very surprised that processing 40MB text and 400MB text data are using nearly the same time, it’s because each task is too small for 40MB data, the overhead is larger than the effective working time, in average, on our environment, each task will use at least 2 seconds on the whole flow, if the effective working time is near to this value, the efficiency may become very low, if the effective working time is too long, a task may hold the machine and occupied lots of resources, if the task failed, the overhead will be much large as the task will start from zero again. The suggestion from Cloudera which is the enterprise Hadoop developer is 14 to 17 seconds for a task runs in map function, this may get better efficiency, but some of the Job could not be controlled to fit this optimal value, such as the video conversion, if we need to meet the time, we need to cut a video into small pieces, each piece may used about that period of time to convert, after the conversion, we need an additional step to combine the pieces together, the cutting and combining task may also use up lots of time, the overall efficient may be lower, but for the task such as MD5 generator, we could turn the length of each split and let the map function run for any time we expect. For the jobs that have a lot of small pieces of input files, we may pack the files together, it may get performance improvement.

Theoretically speaking, any jobs could be ran in a single machine, it could be paralleled by MapReduce in Hadoop, the programming is easy, but it is not always a good choice, Jobs mainly focused on computation is not good in Hadoop, Jobs do not consume too much data may have better choices other than Hadoop. For the convenience of parallel programming, Hadoop maybe the easiest one to learn, the programmer don’t need to have any parallel programming skills, it will easily make it parallel by MapReduce, it is good for the non-computer scientist.

Hadoop is made for scalability not performance, it is good in data processing!
Chapter 4: Overall conclusion

We can see cloud is powerful IT resource pool (public cloud), or management tool (private cloud). Its elastic property allows application resizing itself to any scale. The almost important benefit of using cloud is paying and using the IT resource as you go. How much has you used, how much you should pay. These features of cloud enable small company or organization like school running large scale application or experiment with a reasonable cost.

Even though the cloud provide users with elastic resource, not all application can directly run on the cloud, it re-development of application and put it into cloud layer (e.g. Azure), or distribute the workload on its own (e.g. AWS). In order to fully shows up the performance of cloud. Second approach requires some powerful framework on top of all virtual machines and fully used up all resource form each virtual machine.

Hadoop (MapReduce) is one of the very powerful frameworks that enable easy development on data-intensive application. It objective is help building a supplication with high scalability with thousands of machines. We can see Hadoop is very suitable to data-intensive background application and perfect fit to our project’s requirements.

Apart from running application in parallel, Hadoop provides some job monitoring features similar to Azure. If any machine crash, the data could be recovered by other machines, and it will take up the jobs automatically.

When we put Hadoop into cloud, we also see the convenience in setting up Hadoop. With a few command lines, we can allocate any
number of clusters to run Hadoop, this may save lot of time and
effort.

We found the combination of cloud and Hadoop is surely a common
way to setup large scale application with lower cost, but higher
elastic property.

4.1 Difficulty of this project

There are three main difficulty of our project.

Rapid update of softwares
Both Hadoop and Eucalyptus are not mature application. There is
not enough documentation and user manual, Apart from that, both of
them keep changing and has different usage method. Most
guidelines are written by user and wrong. We put lot of effort on set
up Eucalyptus and Hadoop. We sent a email to Cloudera (Hadoop
supporting website), they haven’t rely us until now.

Testing time is very long
It is different form running a single program on a single machine.
Our testing cycle can last for a few hours to build an image. Since
cloud is and Hadoop are sensitive to network status, the network
always crash and require us to repair and re-set up the environment
with an hour.

Limited resource
Our cloud is very tiny, we can only allocate 370MB memory to each
instance and they have only about 80MB free memory. Such
environment can’t support any large scale computation, like movie
conversion in cloud. Even run all application in physical machines,
we still need a few more machines to evaluate the performance of
Hadoop.
Chapter 5: What we will do next

In this semester, we have not finished the video search yet. We have done lots of effort on building the cloud environment and configuring Hadoop, writing applications to learn the characteristic of Hadoop DFS and MapReduce, in the next semester, we will mainly focused on video searching algorithm in Hadoop MapReduce, such as the data locality, file splitting, comparing spitted data, and also using the Hive interface to store and search the comparison result.

Chapter 6: Acknowledgement

We would like to thank our final year project supervisor: Professor Michael Lyu. He gives us valuable advices for our project and how to present well. Also, he reminds us to take a good scheduling.

Besides, we would like to thank Mr.Edward Yan and Mr. Un Tze Lung in VIEW Lab. They provide us with facilities and technical support when we are setting up the cloud, and some algorithm related to video processing.

Lastly, we thank to Kenny (PhD Student RM 101). He gives us guideline on cloud computing and Mapreduce. so we can better prepare out platform for our project.
Chapter 7: Reference list


http://kiranmurari.wordpress.com/2010/03/19/uec-cc-nc-single-machine/

http://kiranmurari.wordpress.com/2010/03/29/uec-bundling-windows-image/

[6] Ben Selinger , “Slow disk IO for all guests with all disc modes and types.”,
http://web.archiveorange.com/archive/v/4feNTkhvMpte1aDL4w3h


[8] “UECImages”,
https://help.ubuntu.com/community/UEC/Images

[9] “UECCreateYourImage”,
https://help.ubuntu.com/community/UEC/CreateYourImage

[10] “Eucalyptus user handbook”,
http://trac.nchc.org.tw/grid/wiki/Eucalyptus/User


[12] “How To Customize UEC Images”,
http://www.ossramblings.com/customizing-uec-images

[13] “Run your own Ubuntu Enterprise Cloud, part 1-3”,
http://fnords.wordpress.com/2009/10/04/run-your-own-uec-part-1/

[14] “Private cloud setup using eucalyptus and xen”,
http://knol.google.com/k/private-cloud-setup-using-eucalyptus-and-xen#

physical-infrastructure.html

[16] David Chappell,” INTRODUCING WINDOWS AZURE”, MICROSOFT
CORPORATION,2010

[17] “KVM-Virtio”
http://www.linux-kvm.org/page/Virtio

[18] “Virtio”,
http://wiki.libvirt.org/page/Virtio

[19] “Paravirtualization is State of the Art Virtualization”,
http://staging.xen.org/about/paravirtualization.html

http://www.ivedasolutions.net/index.php/cloud-computing/

[21] “Surveillance Video Mining”,
http://www.zdnetasia.com/whitepaper/surveillance-video-mining_wp-
1387385.htm

[22] “MapReduce and HDFS”, University of Washington, 2009 Cloudera, Inc.


[29] Jimmy Lin, “Hadoop: Nuts and Bolts”, University of Maryland, Data-Intensive Information Processing Applications — Session #2, Data-Intensive Information Processing Applications — Session #2

http://www.cloudera.com/
## Appendix

### Progress table

<table>
<thead>
<tr>
<th>Period</th>
<th>Chu Yan Shing</th>
<th>Wu Bing Chuan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summer</td>
<td>Studying the Cloud Computing programming model</td>
<td>Studying the Cloud Computing programming model</td>
</tr>
<tr>
<td></td>
<td>Studying Linux</td>
<td>Studying the surveillance video analysis algorithms and techniques.</td>
</tr>
<tr>
<td></td>
<td>Trying to set up a Cloud but fail with limited resource</td>
<td></td>
</tr>
<tr>
<td>September</td>
<td>Building a new cloud</td>
<td>Studying MapReduce</td>
</tr>
<tr>
<td></td>
<td>Testing the cloud environment</td>
<td>Studying Hadoop</td>
</tr>
<tr>
<td></td>
<td>Building images</td>
<td></td>
</tr>
<tr>
<td>October</td>
<td>Optimizing the cloud</td>
<td>Testing Hadoop with virtual machines</td>
</tr>
<tr>
<td></td>
<td>installing MapReduce to our cloud</td>
<td>Implemented application framework our environment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>November</td>
<td>Testing cloud + Hadoop performance</td>
<td>Implement a few Hadoop applications</td>
</tr>
<tr>
<td></td>
<td>Testing Hadoop application with large amount of data</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Further testing on Hadoop</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Writing report</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Cloud performance testing outputs

#### Performance testing of our private cloud

**(Single process running)**

<table>
<thead>
<tr>
<th></th>
<th>Physical Machine</th>
<th>Instance without virtio</th>
<th>Instance with virtio</th>
<th>Virtio / SCSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (s)</td>
<td>mb/s</td>
<td>s</td>
<td>mb/s</td>
<td>s</td>
</tr>
<tr>
<td>Download</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>15.406</td>
<td>2.02</td>
<td>14.757</td>
<td>2.04</td>
</tr>
<tr>
<td>User Time</td>
<td>0.072</td>
<td>0.044</td>
<td>0.152</td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.486</td>
<td>0.816</td>
<td>0.948</td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>15.447</td>
<td>14.962</td>
<td>11.434</td>
<td></td>
</tr>
<tr>
<td>User Time</td>
<td>0.052</td>
<td>0.048</td>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.42</td>
<td>0.984</td>
<td>1.192</td>
<td></td>
</tr>
<tr>
<td>Compress</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a 30MB mp3 file</td>
<td>Real Time</td>
<td>3.144</td>
<td>3.48</td>
<td>3.469</td>
</tr>
<tr>
<td>User Time</td>
<td>2.94</td>
<td>3.12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.148</td>
<td>0.264</td>
<td>0.252</td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>3.05</td>
<td>3.595</td>
<td>3.754</td>
<td></td>
</tr>
<tr>
<td>User Time</td>
<td>2.892</td>
<td>3.268</td>
<td>2.984</td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.16</td>
<td>0.316</td>
<td>0.292</td>
<td></td>
</tr>
<tr>
<td>Decompress this file</td>
<td>Real Time</td>
<td>0.368</td>
<td>1.758</td>
<td>0.707</td>
</tr>
<tr>
<td>User Time</td>
<td>0.332</td>
<td>0.388</td>
<td>0.368</td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.112</td>
<td>0.144</td>
<td>0.116</td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>0.385</td>
<td>0.643</td>
<td>0.77</td>
<td></td>
</tr>
<tr>
<td>User Time</td>
<td>0.34</td>
<td>0.4</td>
<td>0.388</td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.152</td>
<td>0.108</td>
<td>0.112</td>
<td></td>
</tr>
<tr>
<td>Copy a 30MB file</td>
<td>Real Time</td>
<td>0.074</td>
<td>0.115</td>
<td>0.528</td>
</tr>
<tr>
<td>User Time</td>
<td>0</td>
<td>0.004</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.072</td>
<td>0.1</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>0.104</td>
<td>0.101</td>
<td>0.226</td>
<td></td>
</tr>
<tr>
<td>User Time</td>
<td>0.004</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
### Write a 500MB File

<table>
<thead>
<tr>
<th></th>
<th>Sys Time</th>
<th>User Time</th>
<th>Real Time</th>
<th>User Time</th>
<th>Real Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.076</td>
<td>0.164</td>
<td>5</td>
<td>103</td>
<td>61.87</td>
</tr>
<tr>
<td></td>
<td>0.096</td>
<td>0.388</td>
<td>8.3</td>
<td>10.153</td>
<td>50.9</td>
</tr>
<tr>
<td></td>
<td>0.072</td>
<td>0.548</td>
<td>10.153</td>
<td>50.9</td>
<td>6.0938</td>
</tr>
</tbody>
</table>

### Compress this file 490kb

<table>
<thead>
<tr>
<th></th>
<th>Sys Time</th>
<th>User Time</th>
<th>Real Time</th>
<th>User Time</th>
<th>Real Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.164</td>
<td>0.426</td>
<td>5.2</td>
<td>3.584</td>
<td>1.4509</td>
</tr>
<tr>
<td></td>
<td>0.156</td>
<td>0.48</td>
<td>3.584</td>
<td>1.4509</td>
<td>0.8917</td>
</tr>
<tr>
<td></td>
<td>2.784</td>
<td>3.128</td>
<td>1.4509</td>
<td>0.8917</td>
<td>1.7711</td>
</tr>
</tbody>
</table>

### Decompress this file

<table>
<thead>
<tr>
<th></th>
<th>Sys Time</th>
<th>User Time</th>
<th>Real Time</th>
<th>User Time</th>
<th>Real Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.432</td>
<td>4.968</td>
<td>3.092</td>
<td>1.6067</td>
<td>0.9336</td>
</tr>
<tr>
<td></td>
<td>7.46</td>
<td>8.333</td>
<td>3.092</td>
<td>1.6067</td>
<td>0.9942</td>
</tr>
<tr>
<td></td>
<td>1.492</td>
<td>3.732</td>
<td>1.6067</td>
<td>0.9942</td>
<td>1.4733</td>
</tr>
</tbody>
</table>

### MD5 checksum a 500MB file

<table>
<thead>
<tr>
<th></th>
<th>Sys Time</th>
<th>User Time</th>
<th>Real Time</th>
<th>User Time</th>
<th>Real Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.856</td>
<td>5.258</td>
<td>2.248</td>
<td>2.4591</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.907</td>
<td>67.056</td>
<td>11.033</td>
<td>6.0778</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.228</td>
<td>2.328</td>
<td>1.0567</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.988</td>
<td>3.186</td>
<td>2.4634</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Overall

<table>
<thead>
<tr>
<th></th>
<th>time(s)</th>
<th>ratio</th>
<th>time(s)</th>
<th>ratio</th>
<th>Virtio / SCSI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>88.756</td>
<td>0.25602</td>
<td>120.42</td>
<td>0.7371</td>
<td>2.8789</td>
</tr>
<tr>
<td>User Time</td>
<td>28.436</td>
<td>0.88904</td>
<td>31.734</td>
<td>0.8961</td>
<td>1.0079</td>
</tr>
<tr>
<td>Sys Time</td>
<td>15.07</td>
<td>0.395</td>
<td>22.8</td>
<td>0.661</td>
<td>1.6733</td>
</tr>
<tr>
<td>I/O time</td>
<td>45.25</td>
<td>0.16363</td>
<td>65.885</td>
<td>0.6868</td>
<td>4.1972</td>
</tr>
</tbody>
</table>

(I/O time = real time - user time - sys time)
## Performance testing of our private cloud

(Multi processes running)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mb/s</td>
<td>mb/s</td>
<td>mb/s</td>
<td>mb/s</td>
<td>mb/s</td>
<td>mb/s</td>
</tr>
<tr>
<td>Download a 30MB mp3 file</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>29.029</td>
<td>1.12</td>
<td>28.733</td>
<td>1.06</td>
<td>26.178</td>
<td>0.092</td>
</tr>
<tr>
<td>User Time</td>
<td>0.096</td>
<td>0.088</td>
<td>0.092</td>
<td>0.092</td>
<td>0.172</td>
<td>0.184</td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.552</td>
<td>0.84</td>
<td>0.796</td>
<td>1.224</td>
<td>1.228</td>
<td>0.66721</td>
</tr>
<tr>
<td>Real Time</td>
<td>26.766</td>
<td>1.16</td>
<td>26.766</td>
<td>1.16</td>
<td>27.786</td>
<td>1.16</td>
</tr>
<tr>
<td>User Time</td>
<td>0.68</td>
<td>0.044</td>
<td>0.092</td>
<td>0.128</td>
<td>0.128</td>
<td>0.41463</td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.452</td>
<td>1.124</td>
<td>0.98</td>
<td>1.228</td>
<td>0.66721</td>
<td></td>
</tr>
<tr>
<td>Compress A 30MB mp3 file</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>3.268</td>
<td>3.663</td>
<td>3.899</td>
<td>2.69</td>
<td>3.839</td>
<td>1.15822</td>
</tr>
<tr>
<td>User Time</td>
<td>2.928</td>
<td>3.176</td>
<td>3.508</td>
<td>2.268</td>
<td>3.016</td>
<td>1.26495</td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.116</td>
<td>0.38</td>
<td>0.252</td>
<td>0.228</td>
<td>0.292</td>
<td>1.21538</td>
</tr>
<tr>
<td>Real Time</td>
<td>3.33</td>
<td>9.707</td>
<td>10.118</td>
<td>3.795</td>
<td>3.703</td>
<td>2.64404</td>
</tr>
<tr>
<td>User Time</td>
<td>3.088</td>
<td>3.136</td>
<td>3.156</td>
<td>3.068</td>
<td>2.992</td>
<td>1.03828</td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.1</td>
<td>0.352</td>
<td>0.344</td>
<td>0.26</td>
<td>0.272</td>
<td>1.09027</td>
</tr>
<tr>
<td>Decompress this file</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>0.451</td>
<td>1.954</td>
<td>14.671</td>
<td>0.816</td>
<td>1.117</td>
<td>8.60062</td>
</tr>
<tr>
<td>User Time</td>
<td>0.328</td>
<td>0.348</td>
<td>0.36</td>
<td>0.26</td>
<td>0.388</td>
<td>1.09259</td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.128</td>
<td>0.24</td>
<td>0.192</td>
<td>0.216</td>
<td>2.48</td>
<td>0.16024</td>
</tr>
<tr>
<td>Real Time</td>
<td>0.464</td>
<td>2.919</td>
<td>5.413</td>
<td>0.547</td>
<td>0.546</td>
<td>7.62306</td>
</tr>
<tr>
<td>User Time</td>
<td>0.316</td>
<td>0.348</td>
<td>0.336</td>
<td>0.344</td>
<td>0.316</td>
<td>1.03636</td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.12</td>
<td>0.172</td>
<td>0.188</td>
<td>0.16</td>
<td>0.2</td>
<td>1.09027</td>
</tr>
</tbody>
</table>
| Copy a Real Time      | 0.083  | 0.174  | 10.351 | 0.09   | 0.09   | 58.4722
<table>
<thead>
<tr>
<th>30MB file</th>
<th>User Time</th>
<th>0</th>
<th>0</th>
<th>0.004</th>
<th>0.004</th>
<th>0</th>
<th>0.004</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sys Time</td>
<td>0.08</td>
<td>0.084</td>
<td>0.084</td>
<td>0.092</td>
<td>0.068</td>
<td>0.076</td>
<td>1.22222</td>
</tr>
<tr>
<td></td>
<td>Real Time</td>
<td>0.104</td>
<td>0.118</td>
<td>0.316</td>
<td>0.338</td>
<td>0.107</td>
<td>0.09</td>
<td>3.3198</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>0.004</td>
<td>0</td>
<td>0.008</td>
<td>0.004</td>
<td>0.004</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Sys Time</td>
<td>0.08</td>
<td>0.08</td>
<td>0.252</td>
<td>0.264</td>
<td>0.073</td>
<td>0.076</td>
<td>3.46309</td>
</tr>
<tr>
<td></td>
<td>Real Time</td>
<td>7.676</td>
<td>65</td>
<td>7.648</td>
<td>65</td>
<td>223.44</td>
<td>2.3</td>
<td>222.05</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>0.148</td>
<td>0.14</td>
<td>0.372</td>
<td>0.312</td>
<td>0.584</td>
<td>0.412</td>
<td>0.68675</td>
</tr>
<tr>
<td></td>
<td>Sys Time</td>
<td>4.044</td>
<td>4.184</td>
<td>5.236</td>
<td>5.056</td>
<td>4.084</td>
<td>4.456</td>
<td>1.20515</td>
</tr>
<tr>
<td></td>
<td>Real Time</td>
<td>14.1</td>
<td>36</td>
<td>8.962</td>
<td>57.2</td>
<td>204.93</td>
<td>2.5</td>
<td>209.79</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>0.152</td>
<td>0.132</td>
<td>0.32</td>
<td>0.368</td>
<td>0.512</td>
<td>0.384</td>
<td>0.76786</td>
</tr>
<tr>
<td></td>
<td>Sys Time</td>
<td>3.752</td>
<td>2.824</td>
<td>4.988</td>
<td>5.352</td>
<td>3.824</td>
<td>4.38</td>
<td>1.26036</td>
</tr>
<tr>
<td>Write a 500MB File</td>
<td>Real Time</td>
<td>16.062</td>
<td>16.275</td>
<td>45.058</td>
<td>44.386</td>
<td>52.801</td>
<td>52.547</td>
<td>0.84903</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>7.428</td>
<td>7.448</td>
<td>8.405</td>
<td>8.333</td>
<td>8.365</td>
<td>8.165</td>
<td>1.01258</td>
</tr>
<tr>
<td></td>
<td>Sys Time</td>
<td>1.396</td>
<td>1.364</td>
<td>5.06</td>
<td>4.864</td>
<td>3.092</td>
<td>3.228</td>
<td>1.57025</td>
</tr>
<tr>
<td></td>
<td>Real Time</td>
<td>14.178</td>
<td>14.763</td>
<td>42.423</td>
<td>43.07</td>
<td>49.587</td>
<td>49.046</td>
<td>0.86678</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>7.536</td>
<td>7.476</td>
<td>8.857</td>
<td>8.393</td>
<td>8.137</td>
<td>8.213</td>
<td>1.05505</td>
</tr>
<tr>
<td></td>
<td>Sys Time</td>
<td>1.472</td>
<td>1.116</td>
<td>5.024</td>
<td>5.396</td>
<td>3.532</td>
<td>2.96</td>
<td>1.60505</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>2.228</td>
<td>2.236</td>
<td>2.368</td>
<td>2.46</td>
<td>2.58</td>
<td>2.572</td>
<td>0.93711</td>
</tr>
<tr>
<td></td>
<td>Sys Time</td>
<td>1.68</td>
<td>1.692</td>
<td>5.336</td>
<td>5.276</td>
<td>3.912</td>
<td>4.076</td>
<td>1.32849</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>2.156</td>
<td>2.196</td>
<td>2.312</td>
<td>2.564</td>
<td>2.5</td>
<td>2.404</td>
<td>0.99429</td>
</tr>
<tr>
<td></td>
<td>Sys Time</td>
<td>1.956</td>
<td>2.26</td>
<td>5.464</td>
<td>5.284</td>
<td>3.972</td>
<td>3.876</td>
<td>1.36952</td>
</tr>
<tr>
<td>Decompress this file</td>
<td>Real Time</td>
<td>15.293</td>
<td>14.869</td>
<td>34.154</td>
<td>35.86</td>
<td>55.292</td>
<td>54.673</td>
<td>0.63668</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>1.064</td>
<td>1.116</td>
<td>1.384</td>
<td>1.332</td>
<td>1.24</td>
<td>1.22</td>
<td>1.10407</td>
</tr>
</tbody>
</table>

LYU1005: Cloud computing technologies and applications
### Performance testing of pure user time

<table>
<thead>
<tr>
<th></th>
<th>Host</th>
<th>Instance</th>
<th>Host</th>
<th>2 instance at the same node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32bit</td>
<td>64bit</td>
<td>Proc A</td>
<td>Proc B</td>
</tr>
<tr>
<td></td>
<td>time (s)</td>
<td>time (s)</td>
<td>s</td>
<td>mb/s</td>
</tr>
<tr>
<td><strong>Pure user time Test</strong></td>
<td>Real Time</td>
<td>46.165</td>
<td>45.447</td>
<td>90.946</td>
</tr>
<tr>
<td></td>
<td>User Time</td>
<td>89.438</td>
<td>89.45</td>
<td>90.926</td>
</tr>
<tr>
<td></td>
<td>Sys Time</td>
<td>0.06</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Real Time</td>
<td>46.02</td>
<td>45.443</td>
<td>90.953</td>
</tr>
</tbody>
</table>

(IO time = real time - user time - sys time)
<table>
<thead>
<tr>
<th>User Time</th>
<th>89.49</th>
<th>89.44</th>
<th>90.91</th>
<th>89.514</th>
<th>89.514</th>
<th>91.734</th>
<th>91.822</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys Time</td>
<td>0.024</td>
<td>0.03</td>
<td>0.016</td>
<td>0.02</td>
<td>0.012</td>
<td>0.028</td>
<td>0.04</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>92.185</td>
<td>181.9</td>
<td>184.34</td>
<td>183.2</td>
<td>184.44</td>
<td>182.81</td>
<td></td>
</tr>
<tr>
<td>User Time</td>
<td>178.93</td>
<td>181.84</td>
<td>179.02</td>
<td>179</td>
<td>184.29</td>
<td>182.74</td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>0.084</td>
<td>0.036</td>
<td>0.048</td>
<td>0.072</td>
<td>0.052</td>
<td>0.052</td>
<td></td>
</tr>
<tr>
<td><strong>Performance ratio</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real Time</td>
<td>0.5068</td>
<td>overall</td>
<td>0.9843</td>
<td></td>
<td>1.0008</td>
<td>overall</td>
<td>0.9755</td>
</tr>
<tr>
<td>User Time</td>
<td>0.984</td>
<td></td>
<td></td>
<td></td>
<td>0.9754</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sys Time</td>
<td>2.3333</td>
<td></td>
<td></td>
<td></td>
<td>1.1538</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Memory access speed test**

<table>
<thead>
<tr>
<th>Host Time(s)</th>
<th>16.07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual machine time(s)</td>
<td>20.977</td>
</tr>
<tr>
<td>Ratio(host Virtual)</td>
<td>0.76608</td>
</tr>
</tbody>
</table>