Investigating interactive visualisation in a Cloud computing environment

A dissertation
submitted in partial fulfilment
of the requirements for the degree of
Bachelor of Software and Information Technology with Honours

At

Lincoln University

By

Johari Abdullah

Lincoln University
2010

Abstract

Scientists working with large datasets without a desktop with advanced capacity may not be able to visualise the simulation output efficiently. This is because visualisation of large datasets is computationally intensive in terms of filtering, mapping, and rendering the datasets. The time taken to visualise an image from a large dataset from an underpowered desktop computer may be prolonged, which would not be an interactive experience for the scientists. The desktop can manage a small dataset efficiently compared to client/server mode; however, larger datasets require more memory and number of processors to visualise.

This project investigates if interactive visualisation is feasible in a Cloud computing environment. A virtual machine (VM) was created which was then deployed in a Cloud environment at The University of Auckland to visualise large datasets. Results showed that ParaView server VM could be deployed in a Cloud environment which offers more memory and processors for the VM to be utilised. Thus, the interactive visualisation of large datasets is feasible in a Cloud computing environment. Results from the performance tests showed larger datasets require more memory and numbers of processes to perform the visualisation. However, the increases in number of processes and memory size would not necessarily improve the performance, which depend on the type and size of datasets and the ParaView operations such as filtering, mapping, and rendering. Future work on even larger datasets is warranted.

Keywords:

Interactive visualisation; On-demand visualisation; Cloud computing; Large datasets; ParaView software

Acknowledgement

I am very thankful to all who supported me until I have completed this research. First of all, I would like to thank my supervisors, Dr. Stuart Charters and Walt Abell for their encouragement, support, and guidance from the start to the final stage which enabled me to develop an understanding of the project. They were very supportive and provided detailed comments and suggestions for this project.

I would like to show my gratitude to Nicholas Jones, Gene Soudlenkov, and Richard Hosking from the University of Auckland’s Centre of eResearch for their time and support with the Aotea Cloud infrastructure.

I wish to thank Mark Anderson for his technical assistance and Caitriona Cameron for helping me improve my dissertation writing.

Last but not least, I would like to thank my wife, Norsila Abdul Wahab for her never ending encouragement and support.

Table of Contents

List of Figures

List of Tables

Abbreviations

AMI	Amazon Machine Image
BeSTGRID	Broadband enabled Science and Technology GRID
FTP	File transfer protocol
HPC	High performance computer
IaaS	Infrastructure as a Service
KVM	Kernel-based Virtual Machine
MPI	Message Passing Interface
MPP	Massively Parallel Processing
np	Number of processes
PaaS	Platform as a Service
RAM	Random Access Memory
SaaS	Software as a Service
SCP	Secure Copy
SMP	Symmetric Multiprocessing
SSH	Secure Shell
TCP	Transmission Control Protocol
VM	Virtual machine
VTK	Visualisation Toolkit

Chapter 1

Introduction

Visualisation is the process of translating data into graphical images. It is sometimes referred to as visual data analysis. Scientific visualisation allows scientists to understand their data and gain insights that could not previously be comprehended. However, this process often requires computational power beyond the locally available resources, particularly to accommodate large datasets. For a scientist who uses an underpowered computer, the desktop may be able to load the data and render a view of it but changing the view can be very slow; it is not an interactive experience for the scientist.

Imagine a scientist using a visualisation package to review the latest output from the computational simulation at his/her desktop. The simulation is likely to produce output data at least 10 times larger than the initial dataset. The scientist's desktop computer will probably not be powerful enough to render the data produced, let alone provide interactive visualisation. Nevertheless, the scientist really wants to run the visualisation of the full simulation results from his/her desktop.

One of the potential solutions to remedy this problem is an upgrade of the desktop computer. However, a desktop upgrade is a short term solution as datasets are likely to grow further with more complex simulations. Thus, further upgrading and expense may be necessary. Generally, scientific data often require resources beyond those available on a desktop, including desktops with advanced computational and graphical capabilities.

Another possibility is to use a remote high performance computing (HPC) system which used parallel processing for visualisation. The remote visualisation uses a client/server architecture. This architecture makes a logical separation between the server that performs the data visualisation and the client that displays the results. There are a few options for remote HPC such as supercomputers, clusters, Grid computing, and Cloud computing.

There is a large amount of literature available on remote visualisation using HPC but most of these systems offer a batch environment which is not interactive. Therefore, further investigation is needed to find out the possibility of having interactive visualisation in a Cloud computing environment.

The proposed solution is to make use of a visualisation server which is deployed in a Cloud computing infrastructure. This would involve a visualisation server running as a virtual machine that can be deployed on request with any amount of processors and memory that might be required for the problem being visualised.

The following chapters describe how this solution has been investigated and evaluated. Chapter 2 covers some background about visualisation, HPC, Cloud computing, and specific information on the Cloud environment used in this study. Chapter 3 outlines the proposed research while Chapter 4 provides the details on the design and implementation of this study. Chapter 5 describes the method for evaluation of the usability and performance of remote visualisation in the Cloud. Chapter 6 summarises and discusses the results of the evaluation. Lastly, Chapter 7 summarises the main results from the evaluation and gives recommendations for potential improvements and future work.

Chapter 2

Background

The visualisation of large scale datasets is one of the biggest challenges in scientific visualisation. Many users do not have sufficient computer power and memory locally to do visualisation effectively for these datasets. In this chapter, the visualisation process, software, and approaches are first introduced. Then, the different types of HPC solutions to visualise large and complex datasets are described. Finally, some forms of Cloud computing are discussed.

2.1 Visualisation

Visualisation is the process of converting raw data to graphical images to get an overview of the data. Haber and McNabb (1990) described a conceptual visualization process in three major transformations as shown in Figure 1.

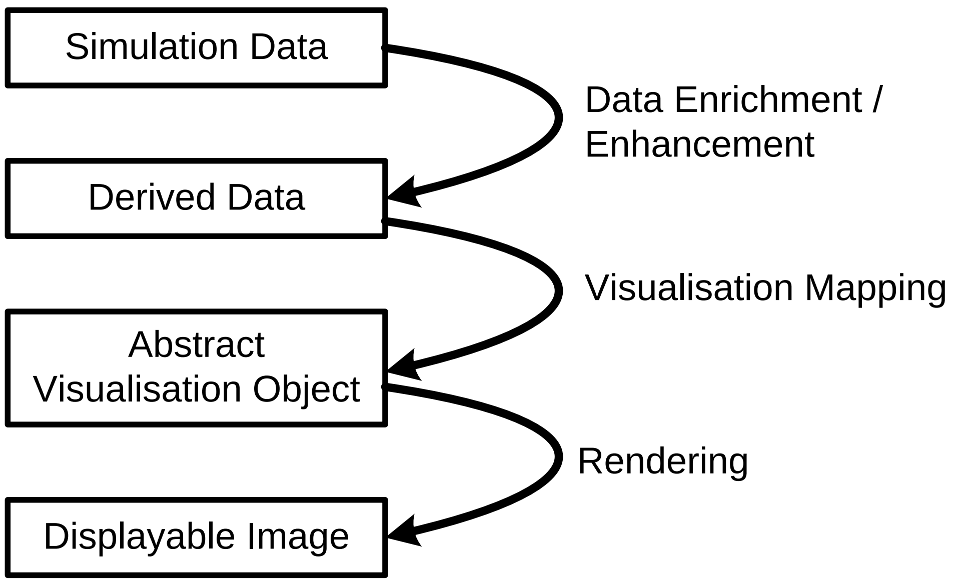


Figure 1: Haber-McNabb conceptual diagram of visualisation.

These transformations occur in most Visualization Pipelines and convert raw simulation data into a displayable image. The first transformation is described as data enrichment/enhancement or data filtering. The raw data from the simulation process are modified into derived data for subsequent visualisation operations. The next transformation is the visualisation mapping. The derived data are mapped into geometric primitives such as points and lines, as well as their attributes such as colour, position and size. The last transformation is rendering where geometric data are transformed into an image. The rendering of large datasets is a computationally intensive process and it is usually beyond the ability of a desktop computer—some form of client/server architecture is required to carry out the rendering process.

There are some large-scale visualisation systems that provide client/server mode such as ParaView, VisIt, and EnSight (Cedilnik, Geveci, Moreland, Ahrens, & Favre, 2006; Moreland, Lepage, Koller, & Humphreys, 2008). According to Moreland et al. (2008), the architecture of the software varies but all comprise a client run locally using a desktop or laptop, which connects via a TCP connection to one or more servers running on a remote parallel machine. For large and complex datasets, it is crucial that all filtering and rendering occur on the server to improve performance.

ParaView, an open source scientific visualisation software, is one of the most commonly used scientific software that supports parallel visualisation of large datasets (Nam, et al., 2009). The software is written in Python and based on the Visualisation Toolkit (VTK). ParaView is able to run in parallel on supercomputers or clusters such that very large datasets may be processed and visualised interactively. It has both a stand-alone mode and a client/server mode. In the stand-alone mode of ParaView, users run the application like any other application using data on the local machine. The processing is also done on the local machine. In the client/server mode, users run a ParaView client on the local machine and the ParaView server on a separate remote machine. The data are loaded into the server which carries out all computations. Rendering can be done on either the local machine or at the server, depending on the configuration of ParaView.

In ParaView, the Visualisation Pipeline concept (see Figure 1) is described by Wernet (2010) as in Figure 2. The data is loaded in ParaView (which can be on either the desktop or server, depending on the size of dataset), filtered, mapped, and rendered, and users can view the display on their desktop.

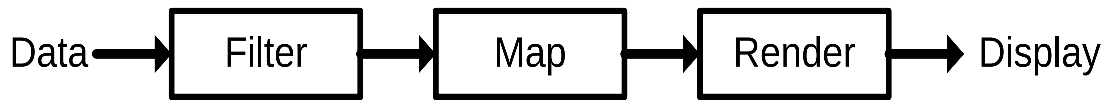
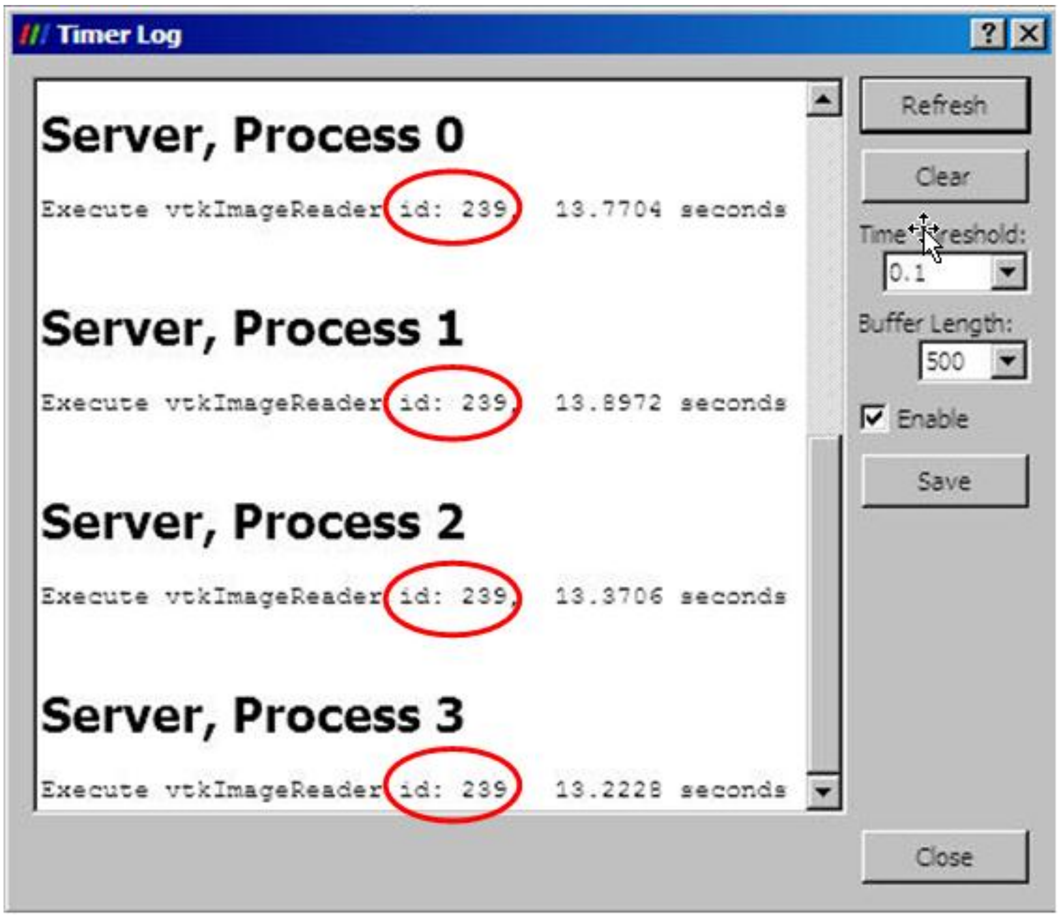


Figure 2: Visualisation Pipeline concept in ParaView.

The ParaView software is able to handle large and complex datasets using several approaches. These approaches include parallel processing, client/server separation, and render server/data server separation (Cedilnik, et al., 2006). For a client/server mode using ParaView, users can either run single or multiple instances in parallel depending on the size of datasets. For small datasets, users can perform the filtering, mapping, and rendering by running a single instance of ParaView. For large and complex datasets, multiple instances running in parallel are preferred. The visualisation output can either be an already rendered image (remote rendering) or a stream of intermediate geometry data which have to be rendered by the client (local rendering). The approaches to process large datasets using ParaView are further discussed below.

2.1.1 Parallel processing

ParaView uses a form of parallelism called data parallelism (Squillacote, 2007). It runs in parallel on distributed and shared memory systems using Message Passing Interface (MPI). The data is divided amongst the processes and each process performs the same operation on its piece of data (Squillacote, 2007). An example of parallel processing is shown in Figure 3.



```
Server, Process 0
Execute vtkImageReader id: 239    13.7704 seconds

Server, Process 1
Execute vtkImageReader id: 239    13.8972 seconds

Server, Process 2
Execute vtkImageReader id: 239    13.3706 seconds

Server, Process 3
Execute vtkImageReader id: 239    13.2228 seconds
```

Figure 3: Parallel processing in ParaView.

Figure 3 shows the "vtkImageReader" computations (a function of ParaView) which are distributed evenly among the processes at the server. The data is divided amongst the processes and each process performs the same operation (id: 239) on its piece of data.

2.1.2 Local rendering

Local rendering is when a single process renders a locally available dataset. When a dataset with low complexity is visualised or the local client is equipped with sufficient graphics hardware, the local rendering mode of ParaView can be selected. In this case, the rendering capabilities of the host server are not used, but the ParaView server sends the geometry data to the client, which renders this geometry locally. This is practical when the datasets are small and require less computation and memory to process. Conversely, for a large dataset where local rendering is not possible, it is imperative that the rendering takes place on the server.

2.1.3 Remote rendering

In the case of remote rendering, the ParaView server uses the graphic processors of the host server to render the image. The rendered image is sent to the client and is shown on the client desktop. This mode is chosen if the client desktop is not equipped with sufficient graphics hardware. For a large dataset, remote rendering is faster than local rendering because it processes the geometry on the server and sends the images to the client. Otherwise, without remote rendering, the server processes the raw data, and then sends the geometry to the client for rendering. The client does all the rendering jobs which can take some time for large datasets. In this case, remote rendering can be faster than local rendering.

2.1.4 Off-screen rendering

Normally, hardware rendering (on-screen) is used for rendering into a window which is displayed on the computer’s screen. For a system with no graphics hardware, it is helpful to use software rendering which renders into an image buffer which is not displayed (McReynolds & Blythe, 2005). This is called off-screen rendering.

One of the problems encountered in client/server mode is the lack of graphics hardware on the server side. This problem can be circumvented by using the OSMesa library (KitwarePublicWiki, 2010c). OSMesa is useful when the server does not have graphic hardware. The first step to compiling OSMesa support is to compile the ParaView with the OSMesa library.

In summary, visualisation of large datasets using ParaView can be performed remotely via some form of servers regardless of the desktop capacity. Next, the types of HPC are described.

2.2 High Performance Computing

HPC uses supercomputers and computer clusters to solve advanced computation problems. One of the advanced computation problems is visualising large and complex datasets. Traditional HPC approaches such as supercomputers, computer clusters, and Grid computing are mainly batch oriented which does not provide on-demand access and interactive visualisation. The Cloud computing approach offers on-demand access and interactive visualisation. The following sections describe supercomputers, computer clusters, Grid computing, and Cloud computing.

2.2.1 Supercomputers

Supercomputers are very fast and powerful custom designed and built computers that can process large quantities of data. These HPC machines are designed to have extremely fast processing speeds; however, they are normally very large and expensive. Most supercomputers perform parallel processing with two approaches: symmetric multiprocessing (SMP) and massively parallel processing (MPP). In the SMP approach, a single copy of the operating system is in charge of all processors and all of the processors use shared memory. An example of SMP is the Cray X1 supercomputer (Dunigan, Vetter, White, & Worley, 2005).

MPP systems consist of multiple processors with each processor using its own operating system and memory. It is a single computer with many networked processors. An example of MPP is the IBM Blue Gene supercomputer (Faraj, Kumar et al, 2009). An example of IBM Blue Gene supercomputer is BlueFern, which is based at the University of Canterbury to provide an HPC e-research facility (BlueFern, 2007).

2.2.2 Computer clusters

A computer cluster is a collection of commodity computers that are connected via a high speed network. Each computer runs a separate instance of the operating system. This setup is often referred as parallel computing since all the computers in the cluster are acting as one machine. With the vast improvement of networking technology, information can be passed throughout the networks with very little lag. This approach has similar characteristics to the MPP supercomputers but it uses commodity hardware for networking, in contrast to the MPP supercomputers which use specialised interconnected networks.

Computer clusters offer more benefits over custom built supercomputers such as scalability, availability, and a reduced cost. While custom built supercomputers have a fixed processing capacity, computer clusters can be expanded as requirements change by adding additional nodes to the network.

According to Schröder (2009), WETA Digital (which produces digital effects for movies such as "The Lord of the Rings" trilogy and "King Kong" in New Zealand) has 5 computer clusters which are used to improve processing performance via massive parallelism.

2.2.3 Grid computing

Grid computing means sharing computing resources such as supercomputers and computer clusters to increase utilisation. It links separate computers to make a large infrastructure by using idle resources (Shuai, Xuebin, Shufen, & Xiuzhen, 2010), which may be owned by diverse organisations. According to Garg, Buyya, and Siegel (2010), Grid computing "enables harnessing of a wide range of heterogeneous and distributed resources for a computational and data intensive application". What distinguishes Grid computing from conventional HPC systems such as supercomputers and computer clusters is that Grid computing tends to be more loosely coupled, heterogeneous, and geographically dispersed. Similar to other HPC system, Grid computing also offer mainly batch computing; hence, would not provide interactive visualisation.

Grid computing is making big contributions to scientific research; it helps scientists around the world to analyse and store massive amounts of data. One example is the Broadband enabled Science and Technology GRID (BeSTGRID)[1] which is supported by New Zealand's Ministry of Research, Science & Technology eResearch programme. BeSTGRID coordinates New Zealand research organisations providing services and infrastructure to conduct research collaboratively with greater computational power and

[1] https://www.bestgrid.org/

data intensity. The current member institutions providing resources, services, and support for BeSTGRID are The University of Auckland, Auckland University Technology, Canterbury University, Industrial Research Limited, Landcare Research, Lincoln University, Massey University, Otago University, Victoria University of Wellington, and Waikato University.

2.2.4 Cloud computing

In contrast to traditional HPC approaches, Cloud computing offers on-demand resource provisioning (Shuai, et al., 2010). Cloud computing offers a new possibility to scientific communities by providing users with control over the remote resources such as control over their configuration (Marshall, Keahey, & Freeman, 2010). The real advantage of the Cloud is the on-demand access compared to other HPC systems. The following section further elaborates the Cloud computing approach.

2.3 Cloud computing

There are many definitions for Cloud computing (Geelan, 2009). For example, Wong (2009) views Cloud computing as an "evolution of the old client/server paradigm, a transformation prompted by the ubiquitous presence of the Internet and the availability of high bandwidth connections". According to Foster et al. (2008), Cloud computing is:

> A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualised, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the internet.

The purpose of Cloud computing is to provide the users with applications or infrastructure without having to purchase or maintain sophisticated hardware.

2.3.1 Forms of Cloud computing

Chengtong et al. (2010), Razak (2009), and Marshall, Keahey, and Freeman (2010) divided Cloud computing into three layers: (a) Infrastructure as a Service (IaaS), (b) Platform as a Service (PaaS), and (c) Software as a Service (SaaS). This is shown in Figure 4.

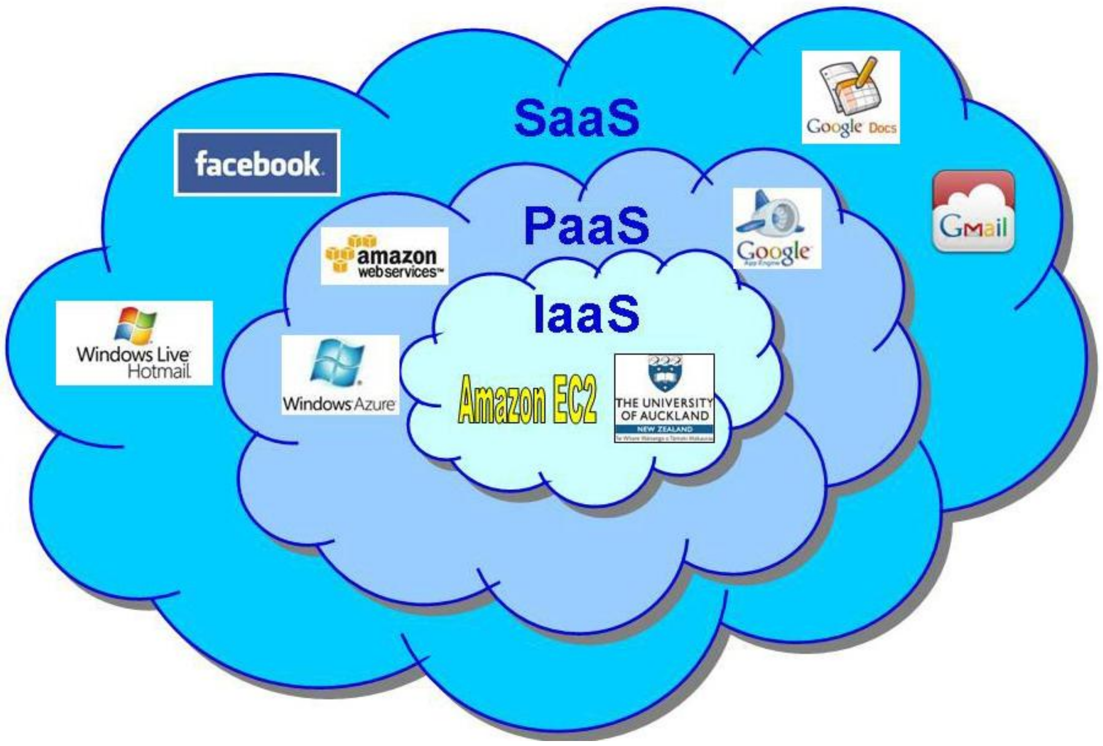


Figure 4: Forms of Cloud computing. SaaS, Software as a Service; PaaS, Platform as a service; IaaS, Infrastructure as a Service.

(a) Infrastructure as a Service (IaaS)

IaaS delivers raw computer infrastructure in the form of virtual machines which can be scaled up and down based on application resource needs. Typical examples are Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3) where compute and storage infrastructures are open to public access with a utility pricing model.

The IaaS has two types of usage—public and private. Amazon EC2 is an example of a public Cloud which offers cloud infrastructure via web services (Akioka & Muraoka, 2010). With Amazon EC2, customers create their own Amazon Machine Images (AMIs) containing an operating system, applications, and data. The customers control how many instances of each AMI runs at any given time. They pay for the instance-hours (and bandwidth) used, adding computing resources at peak times and removing them when they are no longer needed. Private Clouds are for internal corporate data centres or institutions and their services are provided to a limited number of people behind a firewall. A private Cloud is normally for organisations that need more control over their data than they can get by using third-party hosted services such as Amazon EC2 or Amazon S3.

(b) Platform as a Service (PaaS)

PaaS adds a higher level to the Cloud infrastructure, providing a platform upon which applications can be written or deployed. It offers full or partial application development software and libraries for developers such as Google Apps Engine and Windows Azure which are callable over the internet.

(c) Software as a Service (SaaS)

SaaS refers to application software running on Cloud infrastructures, which provide on-demand access via the Internet. Examples include Facebook, Google Docs, Gmail, and Hotmail. It is typically delivered as a direct service to the customer via a web browser rather than installed on their computer.

2.3.2 Virtualisation

Virtualisation is the creation of a virtual version of, for example, an operating system, a server, a storage device, or network resources. Among the reasons for virtualisation's growing popularity is the rise of Grid and Cloud computing. Cloud computing is based on virtualisation technology (Dawoud, Takouna, & Meinel, 2010) which offers a secure, scalable, shared, and manageable environment.

A virtual machine (VM) provides a software-based virtualisation of a host machine. A VM representation (VM image) is composed of a full image of a VM RAM, disk images, and configuration files. Therefore, a VM can be paused or stopped, and later resumed at a different time and in a different location. Once the VM is configured with a full software stack, it can be deployed on remote resources in a matter of milliseconds (Keahey, Freeman, Lauret, & Olson, 2007). Users can develop applications within a familiar environment, and then port this environment between local and remote resources as the need arises. This makes resource provisioning via VM very attractive; the VM can be run easily on local resources as well as on remote resources.

Hypervisor / Virtual Machine Monitor

Software running on a host supporting VM deployment—typically called a Virtual Machine Monitor (VMM) or hypervisor—provides an interface allowing a user to start, pause, and shut down multiple guest VMs. Unlike conventional virtual computing programs, a hypervisor runs directly on the host hardware, instead of as a program under another operating system. This allows both the hypervisor and guest operating systems to perform more efficiently. Examples of hypervisors are VMWare, Xen, and Kernel-based Virtual Machine (KVM).

VMWare is a proprietary product while Xen and KVM are Open Source software. The main different between Xen and KVM is that Xen is an external hypervisor; it assumes control of the machine and divides resources among guests. On the other hand, KVM can be loaded as part of Linux and uses the regular Linux scheduler and memory management. Thus, it is much smaller and simpler to use compared to Xen.

KVM is a full virtualisation solution for Linux on x86 hardware containing virtualisation extensions (Intel VT or AMD-V). Users can run multiple VMs from one or more disk images. Each VM has private virtualised hardware such as network card, disk, and graphics adapter. KVM supports the standard Linux TUN/TAP model for Ethernet bridging (Fenn, Murphy, Martin, & Goasguen, 2008). Each VM gets its own networking resources by using this model. It makes the VM virtually indistinguishable from a physical machine.

2.4 Science cloud

Most Cloud computing infrastructures such as Amazon EC2 and SalesForce.com (SalesForce, 2010) are commercially based with business users in mind. Although scientists can benefit from them as well, scientific applications often have different requirements than business applications such as the use of parallel processing. Therefore, Science clouds were started in mid-2008 by various institutions on a voluntary basis to make it easy for scientific projects to experiment with the IaaS style of the Cloud computing deployed on their clusters (Keahey, Figueiredo, Fortes, Freeman, & Tsugawa, 2008).

According to Keahey et al. (2008), the first Science cloud became available on 3 March 2008 at the University of Chicago. It was named “Nimbus” which was eventually adopted as the name of their software project. It provides compute capability in the form of Xen VMs that are deployed on physical nodes of the University of Chicago TeraPort cluster (16 nodes) using Nimbus software. It is available to all members of the scientific community wanting to run software in the Cloud.

This model was adopted in building Aotea Auckland Research Cloud (Hosking, 2010), a Science cloud as a service inside the Grid computing infrastructure, which is part of BeSTGRID and the University of Auckland’s Centre for eResearch.

2.4.1 Nimbus

Nimbus (Nimbus, 2010) is an open source toolkit that can turn a cluster into an IaaS Cloud. It allows a client to use remote resources and deploy VMs on those resources and configure them to represent an environment desired by the user. It is also known as a Cloud IaaS solution (Ogrizovic, Svilicic, & Tijan, 2010). The Nimbus client allows users to submit their VM images to the Cloud, browse VM images inside the Cloud, deploy VMs, query VM status, and finally access the VM (Lizhe, et al., 2008). The main advantage of Nimbus is that it provides a public IP address to the VM; thus, complex network configurations are not needed to access the VM (Nimbus, n.d.).

2.4.2 Job Scheduler

A Job Scheduler such as Condor (Condor, 2010) is a program that enables batch job scheduling. It can initiate and manage jobs automatically by processing prepared job control language statements or through equivalent interaction with a human operator. It is a single point of control for all the work in a distributed network of computers. For example, users submit their jobs to Condor, and Condor places them into a queue, chooses when and where to run them, monitors their progress, and informs users upon completion.

2.4.3 Cloud Scheduler

Cloud Scheduler (Igable, 2009) is a Cloud-enabled distributed resource manager back-end. It manages VMs on Clouds like Nimbus, OpenNebula (OpenNebula, 2010), or Amazon EC2 to create an environment for batch job execution. For the most part, users do not interact with Cloud Scheduler at all. For example, users prepare a VM image loaded with the desired software, and then upload it to an image repository. Users can then specify a VM image as a processing job to the Job Scheduler. Cloud Scheduler looks at the job queue to discover which VM images are needed to complete the jobs in queue. It then boots the VM images on the clusters it has access to. These VM images run the jobs from the queue, and the Cloud Scheduler then shuts them down when they are no longer needed.

2.4.4 Globus

Globus is an open source Grid software. The Globus Toolkit is a "technology for building Grids that allow distributed computing power, storage resources, scientific instruments, and other tools to be shared securely across corporate, institutional, and geographic boundary"[2]. The Grid Computing infrastructure in BeSTGRID uses Globus to run jobs (Bestgrid, 2010).

2.4.5 Aotea Auckland Research Cloud

Aotea Auckland Research Cloud (Aotea Cloud) was built to allow clients to access the Nimbus service, allowing provisioning and management of VMs. Nimbus uses the Cloud Scheduler to allow Grid jobs to be run on this Cloud as shown in Figure 5.

[2] http://www.globus.org

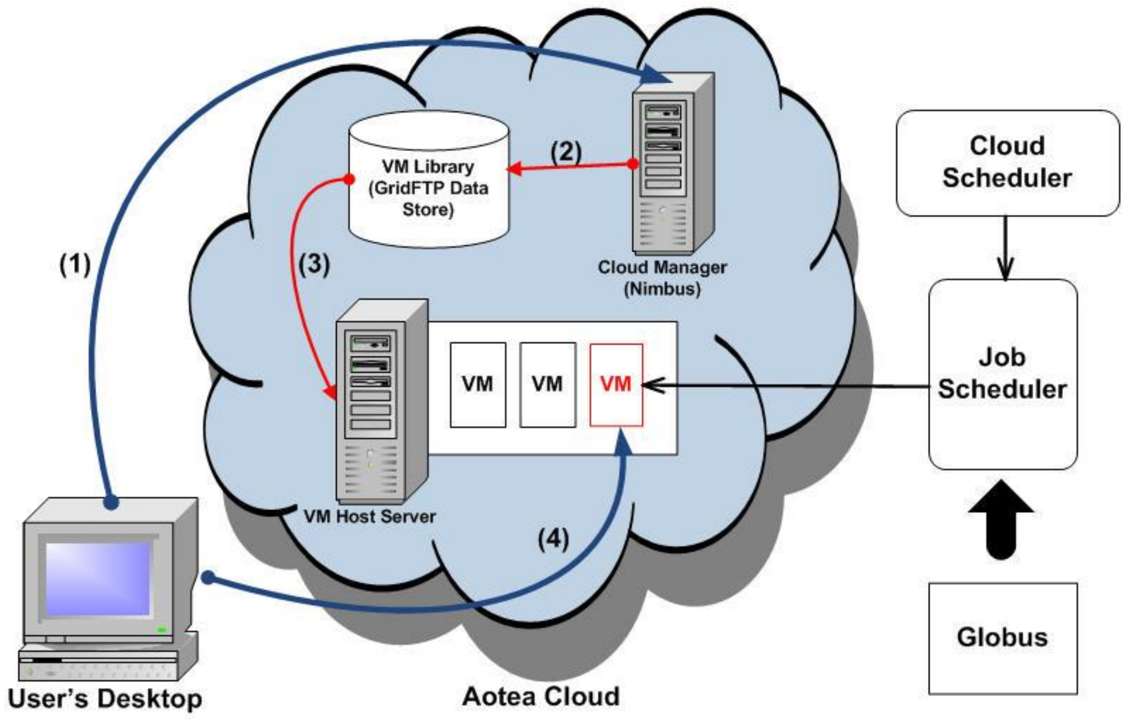


Figure 5: Provisioning and deploying of ParaView server VM.

Figure 5 shows the steps in deploying a VM in the Cloud. (1) First, the user requests the Cloud Manager to provision the VM. (2) Next, the image is downloaded to the VM Host (3) from the VM Library. The Job Scheduler treats this VM as an available node to run jobs. Then, the Cloud Scheduler queries the job queue and discovers which VMs are needed to run the jobs and boots the VM on the grid. (4) Finally, the user connects to the VM and starts the visualisation. The Cloud Scheduler shuts down the VM when it was no longer needed.

2.5 Summary

From the above discussion, it can be noted that visualisation of large datasets is computationally intensive in terms of filtering, mapping, and rendering such that a desktop computer is unlikely to be able to process the data. Moreover, using an underpowered desktop PC would take a lot of time to process the data, thereby lessening the interactive experience of the scientist. Therefore, remote visualisation using a Cloud computing environment that can provide interactive visualisation is recommended.

Chapter 3

Proposed Research

A scenario where a scientist could not visualise large datasets from his/her desktop was discussed in Chapter 1. Visualisation of large datasets, which is computationally intensive, and the solutions were described in Chapter 2. Users tend to use the remote visualisation via some form of HPC to deal with increasing data or complex visualisations. However, the traditional HPC is normally subjected to a batch processing queue-based system which is unsatisfactory to the end users. The on-demand and interactive nature of Cloud computing offers an attractive new dimension to HPC. This project attempts to deploy a visualisation server VM in a Cloud computing environment to provide high performance interactive visualisation to end users. The following section describes the research aims and objectives of this study.

3.1 Aim of the research

The goal of this study was to investigate if users can visualise large, complex datasets from their desktop by using a Cloud visualisation server as shown in Figure 6.

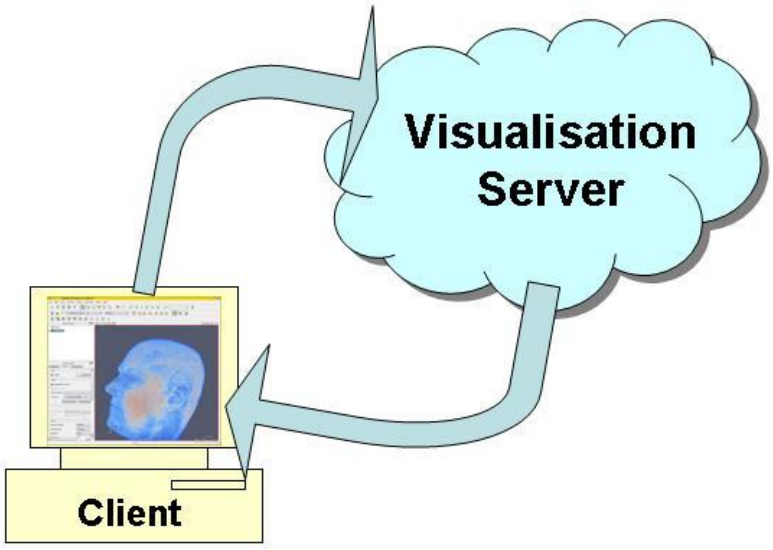


Figure 6: Visualisation in a Cloud environment.

Figure 6 shows the schematic representation of a remote visualization in a Cloud environment where a high-specification VM running a visualisation server is deployed. Users connect to VM and the visualisation process is performed in the Cloud and the resulting image is sent to the user’s low-specification desktop.

To evaluate the success of this solution, the usability issues in deploying the VM on request and connecting the local client to the server are measured, as well as the performance when processing large datasets.

3.2 The objectives

The objectives of this project are:

1. To investigate the ability to deploy a visualisation system in a Cloud environment.
2. To assess the ease of use of Cloud visualisation from the point of view of a scientist.
3. To assess the performance and scalability of the solution when multiple processors are used and RAM is increased.
4. To compare the performance of visualisation via client/server mode in a Cloud environment with stand-alone mode using a desktop.

Chapter 4

Design and Implementation

The goal of this study is to investigate if users could visualise large datasets from their desktop using the Cloud. The visualisation should be:

1. interactive,
2. easy to use,
3. able to visualise large datasets and
4. available on-demand.

4.1 Design

To meet the goals, it was decided to use an IaaS approach because of its scalability based on application resource needs. A customised VM image can be created containing the operating system, application, and data. The number of CPU cores and size of memory can be allocated according to the size of the datasets. The Aotea Cloud was used as it is the only available Science cloud in New Zealand.

ParaView was used because of it's ability to process and visualise massive datasets via parallel processing. It supports client/server mode and can be compiled with MPI and OSMesa off-screen rendering libraries. The advantages of Cloud resources could be utilised to run the computation and rendering processes in parallel. A ParaView server VM (Cloud VM) was created using KVM (which is similar to the Aotea's Cloud environment). In order to deploy the Cloud VM at Aotea Cloud, the following steps were implemented:

1. Set up a test environment.
2. Build a ParaView server VM (Cloud VM).
3. Test the Cloud VM locally.
4. Provision the Cloud VM in the Aotea Cloud.
5. Test the Cloud VM remotely.

The client version of ParaView was installed in a desktop PC at Lincoln University and the ParaView server was deployed on the Cloud VM in the Aotea Cloud. The connection of the ParaView client at Lincoln University to the ParaView server VM in the Cloud environment at The University of Auckland via KAREN high-speed network is shown in Figure 7.

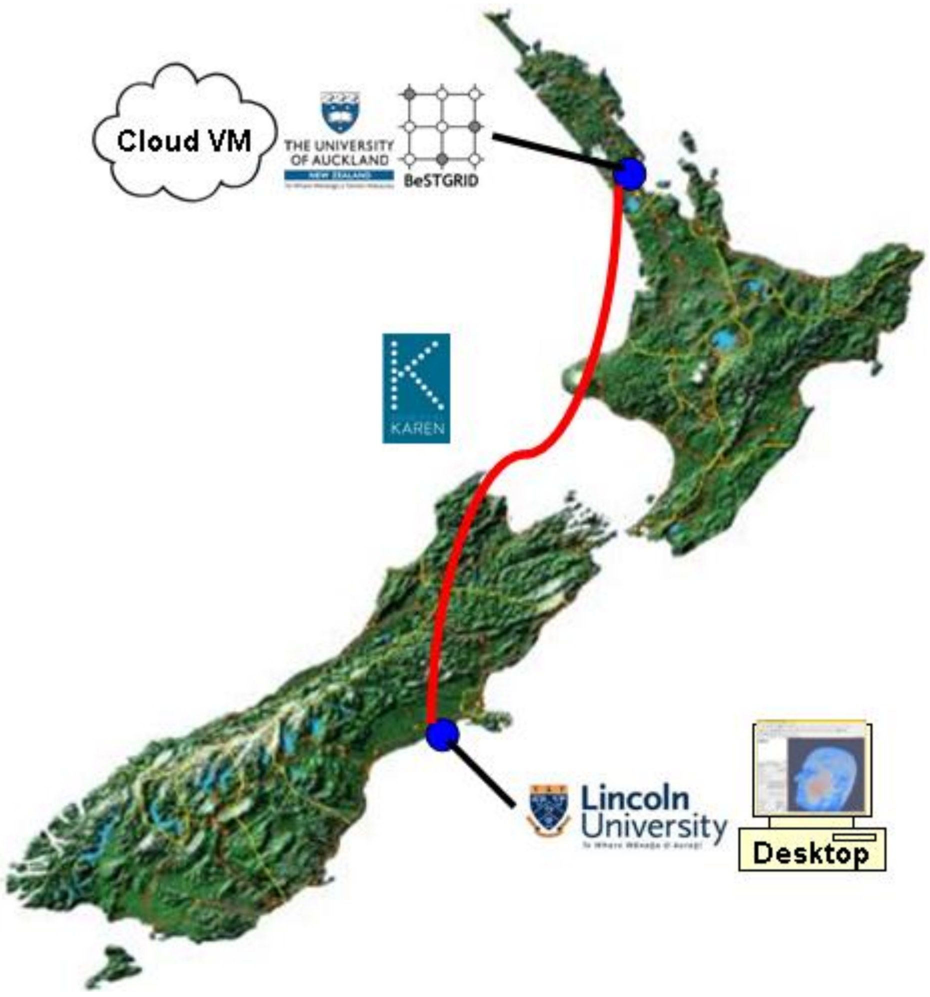


Figure 7: Connecting desktop client with the Cloud VM.

4.2 Implementation

The following describes the implementation phases of the study.

4.2.1 Set up a test environment

First, a Lincoln Test Host machine was installed with Ubuntu Server 10.04 as the host OS as shown in Figure 8. The KVM was then installed on the host. A step-by-step guide to configure a KVM on Ubuntu Server 10.04 and the management of the virtual machines was described by Tan (2010). The Cloud VM was then created using KVM.

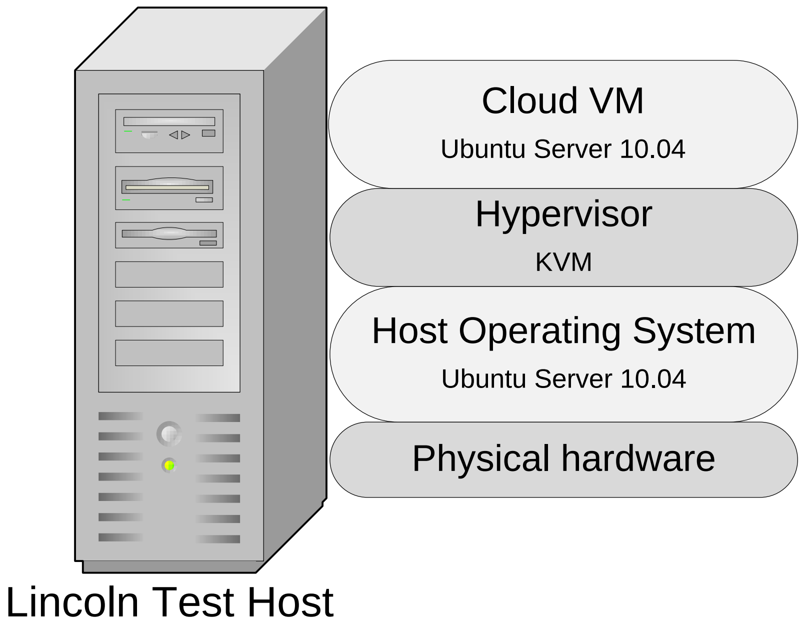


Figure 8: Test environment setup.

The Ubuntu Server 10.04 32-bit architecture can only utilise a maximum memory of 4GB (Lentz, 2010) compared to 64-bit architecture which can address up to 16.8 TB of memory. Therefore, the Ubuntu Server 10.04 64-bit was chosen as an OS for both the Cloud VM and the Lincoln Test Host because more than 4 GB of memory was needed to do visualisation with large datasets.

4.2.2 Build a ParaView server VM (Cloud VM)

The Cloud VM was created on the test environment as shown in section 4.2.1. A ParaView server was built on the Cloud VM, compiled with MPI and OSMesa libraries, on Ubuntu Server 10.04 operating system as shown in Figure 8.

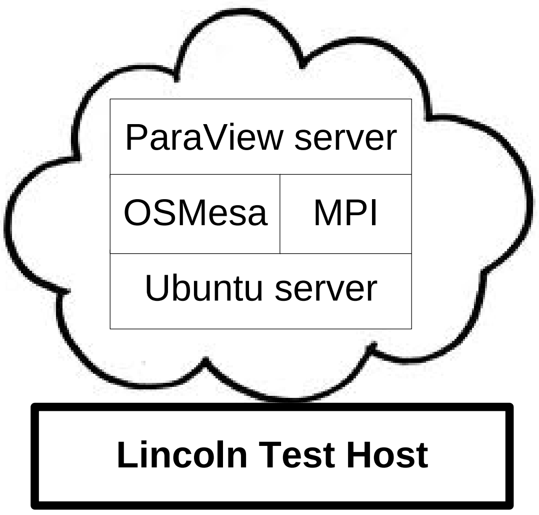


Figure 9: Building a Cloud VM.

The standard install files or ready-made binary packages of ParaView do not support MPI for parallel processing and OSMesa for off-screen rendering. Therefore, the ParaView server version 3.6.2 was built from the source code and compiled with the Open MPI library version 1.4.2 (OpenMPI, n.d.) and OSMesa library version 7.8.2 (Mesa, n.d.). Details on how to build and install ParaView are provided at ParaView Public Wiki (KitwarePublicWiki, 2010b).

4.2.3 Test the Cloud VM locally

The connectivity of the Cloud VM was tested from another VM within the Lincoln Test Host. Then, the Cloud VM was tested from the desktop client.

Testing the ParaView client/server connection from another VM

The second VM, ParaviewClient VM was created with Ubuntu Desktop 10.04 OS. The ping command was used to test connectivity between the two VMs as shown in Figure 10.

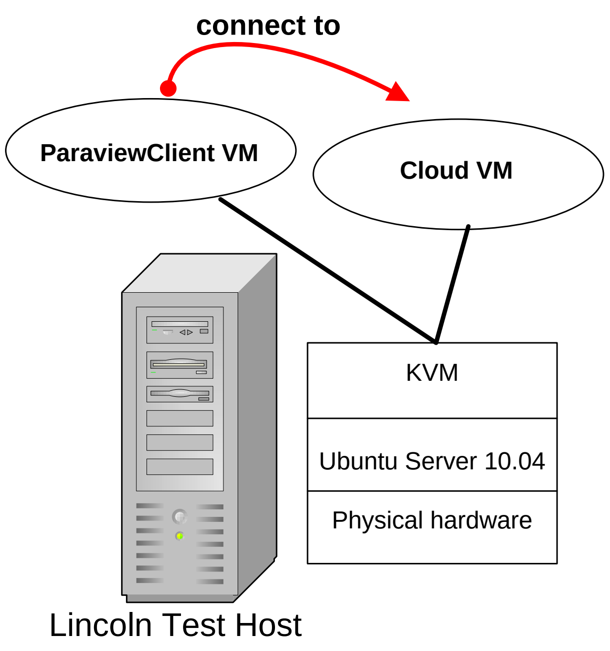
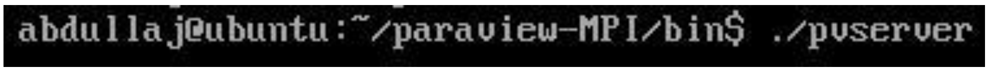


Figure 10: Testing connection between two VMs.

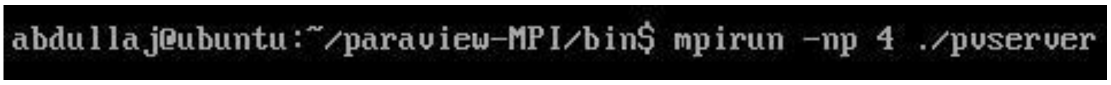
Once the VMs were able to communicate with each other, the ParaView client/server mode operation was tested. ParaView was started on both the server and client sides. The executable functions were in the "bin" folder. The standard command for single-threaded operation of the ParaView server is shown in 11.



```
abdullaj@ubuntu:~/paraview-MPI/bin$ ./pvserver
```

Figure 11: Standard command for a single-threaded operation.

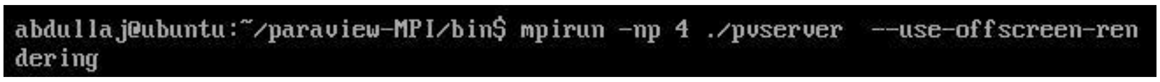
To run the ParaView in parallel, such as running four processors, the command is shown in Figure 12.



```
abdullaj@ubuntu:~/paraview-MPI/bin$ mpirun -np 4 ./pvserver
```

Figure 12: Running ParaView in parallel. –np = number of processes.

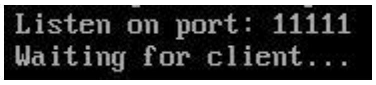
To use pure software rendering, the “use-offscreen-rendering” flag was used as shown in Figure 13.



```
abdullaj@ubuntu:~/paraview-MPI/bin$ mpirun -np 4 ./pvserver  --use-offscreen-ren
dering
```

Figure 13: Running ParaView in parallel with off-screen rendering.

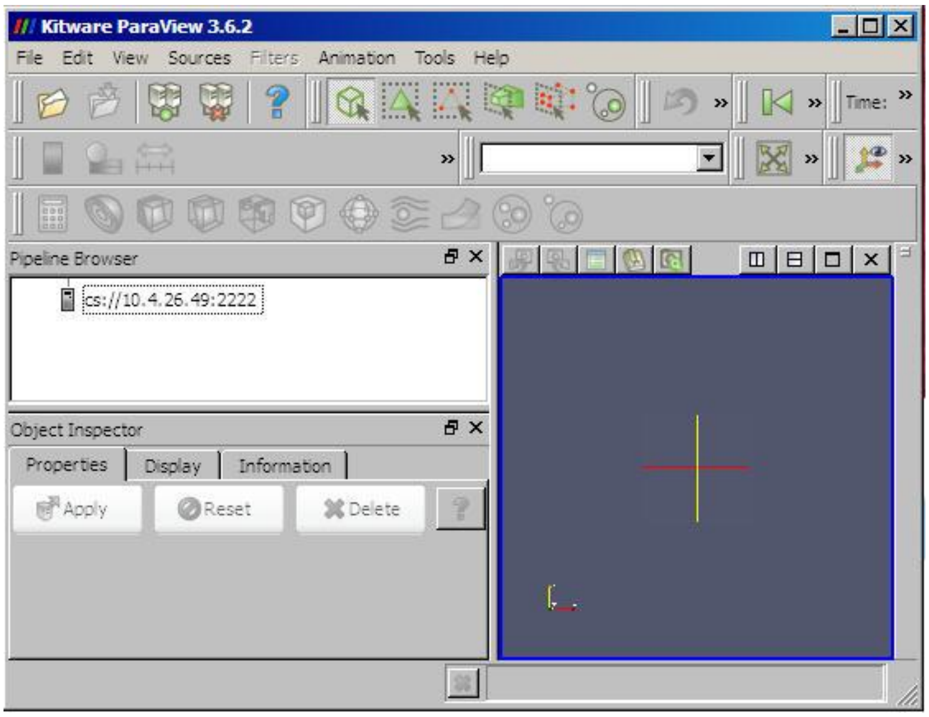
Figure 14 shows the response from the server.



```
Listen on port: 11111
Waiting for client...
```

Figure 14: ParaView server waiting for client’s connection.

Then, the client was able to connect to the server. The client would connect to the server if the “Pipeline Browser” in the ParaView client user interface was set to the server IP address such as 10.4.26.49 at port 2222 as shown in Figure 15.



Pipeline Browser

cs://10.4.26.49:2222

Figure 15: ParaView client connected to ParaView server.

Testing the ParaView client/server connection from the desktop

The next step was to test the client-server connection from outside the Lincoln Test Host, such as from the desktop client as shown in Figure 16.

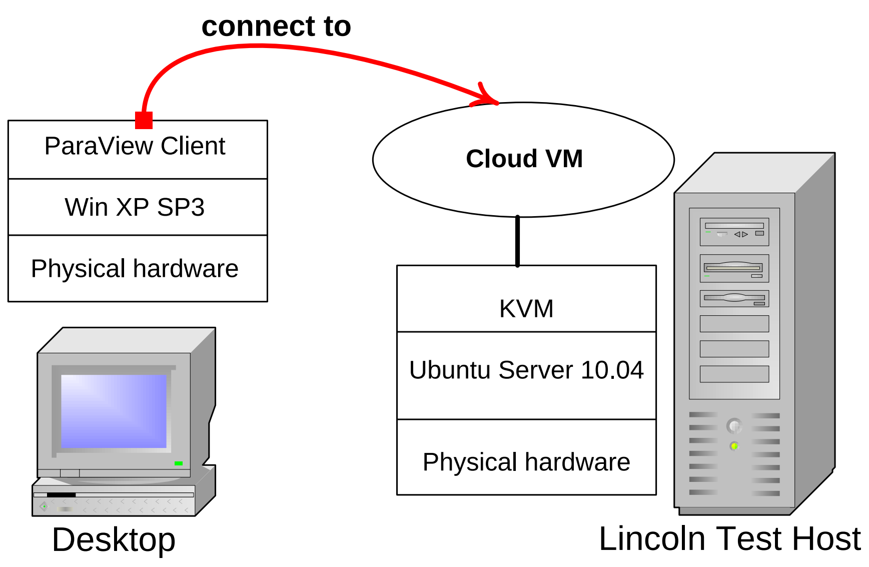


Figure 16: Testing connection from desktop.

The Cloud VM was allocated a private IP address which was not accessible from outside the Lincoln Test Host. Therefore, in order to allow outside connections to the Cloud VM, port forwarding to the Cloud VM was set up on the Lincoln Test Host as shown in Figure 17.

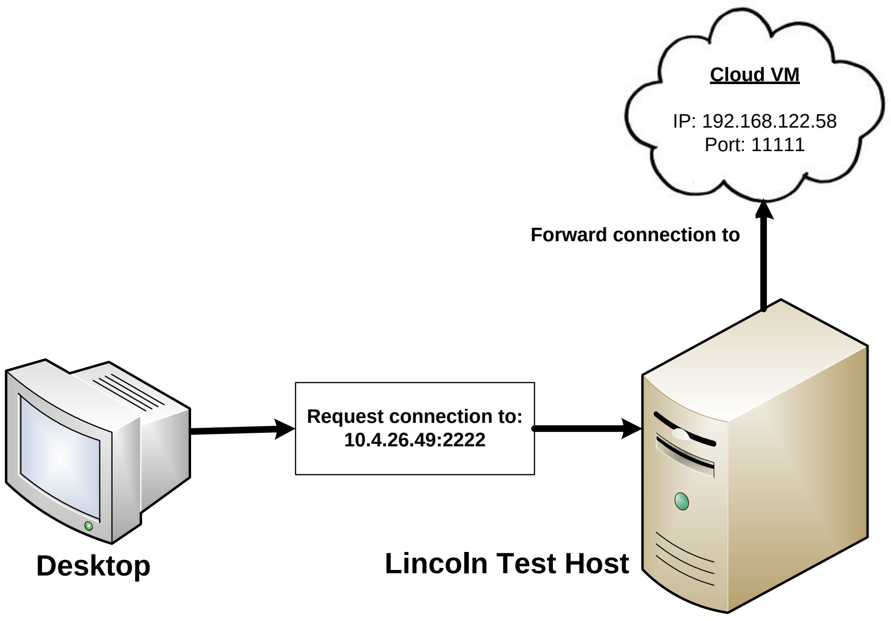


Figure 17: Port forwarding from the Lincoln Test Host to the Cloud VM.

Every connection request from desktop client went to the Lincoln Test Host IP at 10.4.26.49 port 2222. The connection was automatically forwarded to the Cloud VM at the IP 192.168.122.58 port 11111.

4.2.4 Provisioning the VM in the Cloud

The following steps were implemented in order to provision the VM in the Cloud.

Uploading the VM image to the VM Library

The location of the Cloud VM image on the Lincoln Test Host was in /var/lib/libvirt/images/. The image, pvserver.img was uploaded via the file transfer protocol (FTP) to the VM library (GridFTP Data Store) in the Aotea Cloud, as shown in Figure 18.

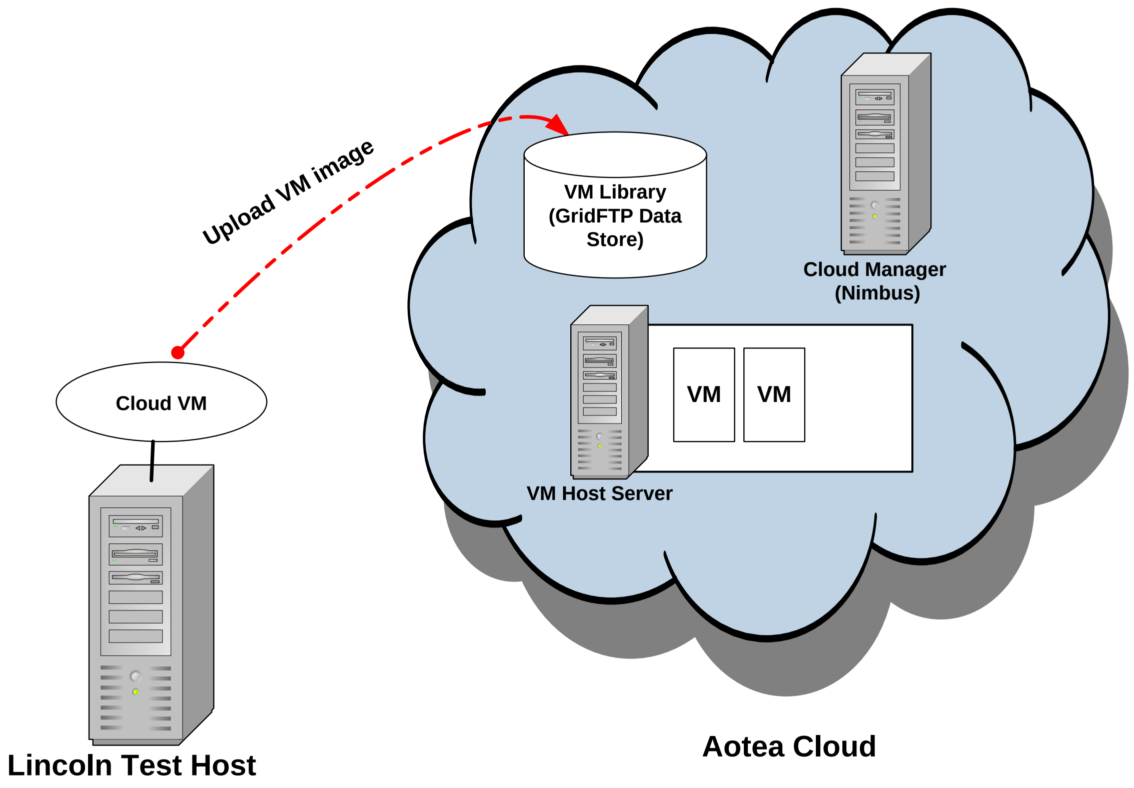


Figure 18: Uploading the Cloud VM to the Aotea Cloud's VM Library.

The uploading of the VM image only needed to be done once.

Provisioning the Cloud VM via Nimbus

Generally, the Cloud client software is used to connect to the Aotea Cloud. However, during the implementation of this project, the desktop was connected to Nimbus directly via the BeSTGRID gateway in order to provision the Cloud VM. This is shown in Figure 19.

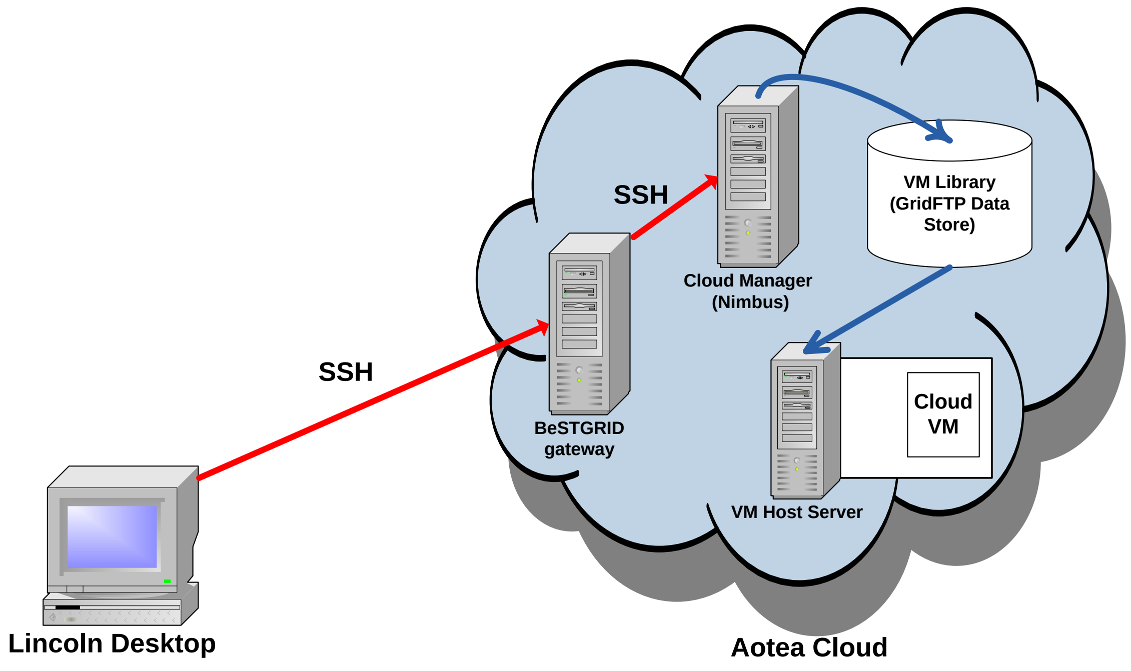


Figure 19: Provisioning of Cloud VM via Nimbus.

Figure 19 shows the process of provisioning the Cloud VM. First, a secure connection was established with BeSTGRID gateway using Secure Shell (SSH) protocol. Next, the connection was established with Nimbus using SSH via the BeSTGRID gateway. A request was then sent to Nimbus to provision the Cloud VM. The following command was used to run the Cloud VM for 2 hours as shown in Figure 20.

```
$cd /home/nimbus/nimbus-cloud-client-017
$ ./bin/cloud-client.sh --run --name pvserver --hours 2
```

Figure 20: Command to provision the Cloud VM.

This started the Cloud VM running with duration of 2 hours. The Cloud VM was launched in the VM Host Server with the private IP address of 192.168.122.87. The ParaView client was then connected to the Cloud VM via VM Host Server using SSH protocol as shown in Figure 21.

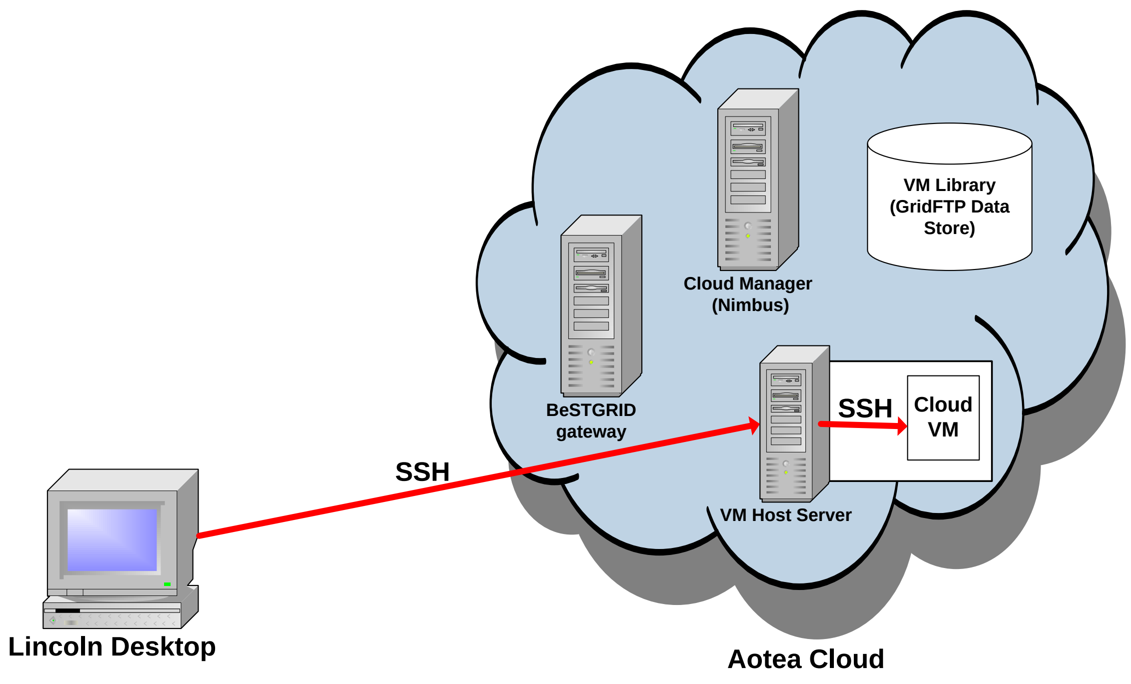


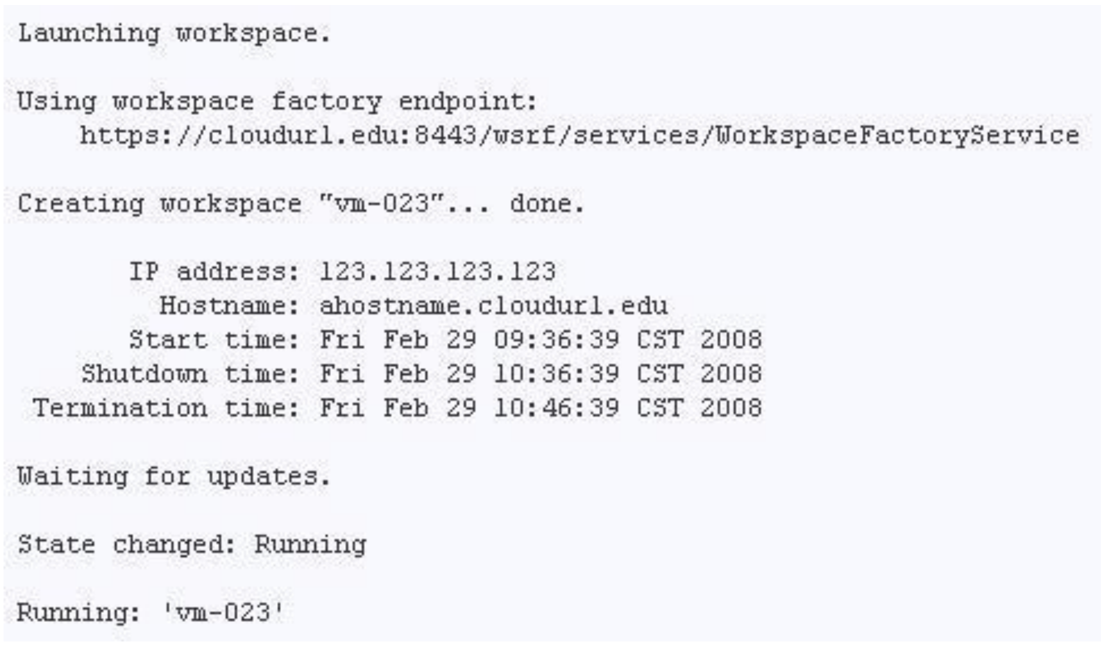
Figure 21: Connecting to the Cloud VM.

4.2.5 Test the Cloud VM remotely

The ParaView server in the Cloud VM accepted connections from the ParaView client on the Lincoln desktop via port forwarding. The connection did work once. However, further testing could not be done as the University of Auckland had internal networking problems which made the VM Host Server inaccessible. Nevertheless, the concept of remote visualisation via a Cloud was proven in the implementation of the Lincoln Test Host.

The University of Auckland had run out of public IP addresses to give out to Cloud VMs. In addition, machines in Aotea Cloud had internal networking problems where the VM Host Server had to be reformatted a few times. The provisioning of the Cloud VM was successful initially, up to the time when the VM Host Server was inaccessible due to networking problems. A test to carry out the visualisation in the Cloud could not be done using the Aotea Cloud. Therefore, as an alternative for the visualisation and performance testing, a local server at Lincoln University (Lincoln Hoiho Server) was set up.

Figure 22 shows a screenshot, taken from Nimbus website[3] of how the provisioning process of a VM would have proceeded if Aotea Cloud had been functioning properly.



```
Launching workspace.

Using workspace factory endpoint:
    https://cloudurl.edu:8443/wsrf/services/WorkspaceFactoryService

Creating workspace "vm-023"... done.

       IP address: 123.123.123.123
         Hostname: ahostname.cloudurl.edu
       Start time: Fri Feb 29 09:36:39 CST 2008
    Shutdown time: Fri Feb 29 10:36:39 CST 2008
 Termination time: Fri Feb 29 10:46:39 CST 2008

Waiting for updates.

State changed: Running

Running: 'vm-023'
```

Figure 22: Provisioning a VM using Nimbus.

Figure 22 shows the public IP address of 123.123.123.123 as well as the start time and shutdown time of the VM. Users could connect to the VM using an SSH connection and start the ParaView server. Then, the ParaView client would be started, and the connection to 123.123.123.123 at port 11111 would be established. Finally, the datasets would be loaded and visualised in the Cloud environment until the shutdown time of the VM.

Test environment to substitute Aotea Cloud

The test environment to substitute for the Aotea Cloud was built on the Lincoln Hoiho Server. This environment was not a VM environment and the ParaView server was installed directly on the Lincoln Hoiho Server for performance testing. The Lincoln Hoiho Server was running under Ubuntu Server 9.04, 64 bits, on an 8-core machine with 8 GB RAM. ParaView version 3.6.2, 64 bit, compiled with MPI and Mesa support

[3] http://www.nimbusproject.org/docs/current/clouds/cloudquickstart.html

was built and installed on the Lincoln Hoiho Server. The desktop and Lincoln Hoiho Server were connected via Lincoln University network at 100 Mb/s.

4.2.6 Summary

This study aimed to investigate the effectiveness of using the Cloud computing in visualising large datasets. The Cloud VM image was successfully built and tested on the Lincoln Test Host. It was uploaded to the VM Library in the Aotea Cloud and successfully provisioned to the VM Host Server via Nimbus.

Due to network problems with the Aotea Cloud, the performance of the Cloud solution could not be directly tested. However, the use of a new test environment (Lincoln Hoiho Server) allowed testing of the client/server performance aspects.

Chapter 5

Evaluation

The objectives of this project are:

1. To investigate the ability to deploy a visualisation system in a Cloud environment.

2. To assess the ease of use of Cloud visualisation from the point of view of a scientist.

3. To assess the performance and scalability of the solution when multiple processors are used and RAM is increased.

4. To compare the performance of visualisation via client/server mode in a Cloud environment with stand-alone mode using a desktop.

Objective 1 was demonstrated by the implementation which was described in Chapter 4. To assess objective 2, a usability test was designed as described in Section 5.1. To assess objectives 3 and 4, the performance tests were designed as described in Section 5.2.

5.1 Usability test

To evaluate objective 2, the scenario described in the introduction chapter will be used. The procedures the scientist would do were using SSH to connect to Nimbus to provision the VM, then using SSH to connect to the VM via VM Host Server to start the ParaView server. Finally, the desktop client was connected to the server to perform the visualisation. The usability test will compare the ease of running ParaView using the desktop for stand-alone mode with client/server mode via the local server and the Cloud if it worked properly.

5.2 Performance tests

The performance tests are designed to evaluate objectives 3 and 4. Three medical datasets with increasing size and memory will be used. To assess objective 4, the timing will be measured using ParaView's built-in timer for comparisons.

5.2.1 Datasets for performance tests

The data was downloaded from the University of Tubingen, Germany website[4] as shown in Table 1. All the medical datasets were in RAW format produced from medical 3D scanners. The image of each dataset and the details are shown in Appendix 1.

Table 1: Medical datasets used in performance tests

Datasets	Size (MB)	Cells (million)	Dimension (voxels)
Head MRI	7.8	8	256 x 256 x 124
Abdominal Aorta	43.5	45.2	512 x 512 x 174
Head Aneurysm	256	133.4	512 x 512 x 512

Default configurations in ParaView such as Level of Detail parameter and Isosurface value will be used unless otherwise stated. The remote rendering threshold will be set to 0 MB so all rendering will be performed on the server.

Three tests which will be measured using ParaView built-in timer will be carried out. The tests are data loading time, isosurface filtering time, and data rendering time. Every test will be repeated five times and the averaged results will be used.

Data used in this study is considered large in accordance to Ahmed, Latiff, Abu Bakar, and Rajion (2007). The authors used 2 sets of medical data in RAW format for testing. They consider datasets of 5 million polygons as large datasets. Their dataset sizes were 15.1 MB with 4.28 million polygons and 23.1 MB with 11.17 million polygons.

[4] http://www.gris.uni-tuebingen.de/edu/areas/scivis/volren/datasets/new.html

CD-adapco, the provider of engineering simulation solutions for fluid flow, heat transfer, and stress to the industry, consider datasets with more than 2 million cells as large case (Benchmarks STAR-CD V3.20, 2008). In this study, datasets of 8, 45.2, and 133.4 million cells were used. This corresponds to 7.8, 43.5, and 256 MB of dataset size, respectively.

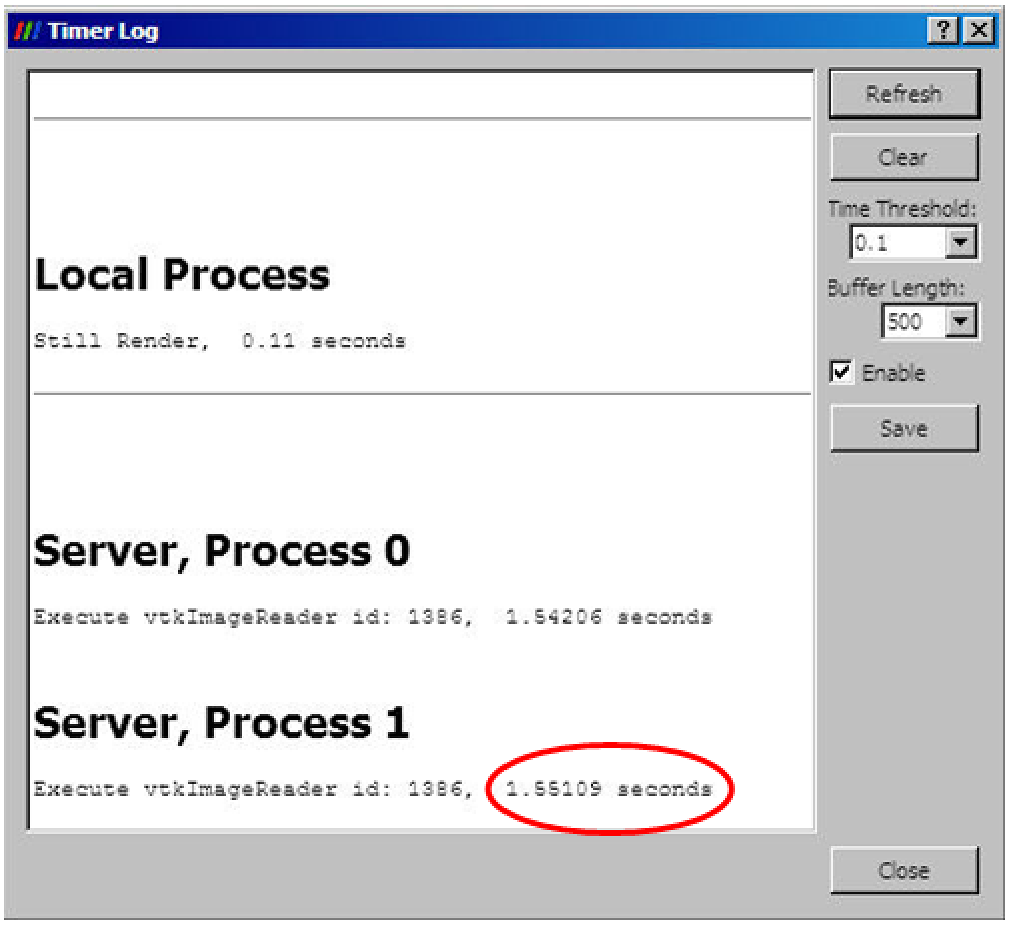
5.2.2 Data loading times

This test will compare the performance of the desktop, Cloud VM on Lincoln Test Host, and Lincoln Hoiho Server in loading the datasets using single-threaded operation. This will show the data loading time for the three different sizes of datasets using the "pvserver" command. Data loading time of the client/server modes will be further tested with 1, 2, and 4 processors using "pvserver" with MPI and OSMesa support. Additionally, the Lincoln Hoiho Server will be tested on 6 and 8 processors. For these datasets, ParaView uses "vtkImageReader" function to read the datasets and displays the outlines of the 3D volume.

The datasets are in RAW format. The ParaView RAW data reader automatically spreads the file among all the ParaView servers that were running (KitwarePublicWiki, 2010a). For this type of dataset, the settings to load the datasets into ParaView are:

> Data Type: Unsigned char
> Byte Order: Little Endian
> Extent: According to the dataset size dimension e.g. 0 511 0 511 0 511 for a dimension of 512 x 512 x 512.

Figure 23 shows the screenshot of the ParaView client's Timer Log on how the time will be calculated for loading datasets. For these server processes, the longest time is Process 1 (1.551 s). This value must be added to the local process time (0.11 s) for a total loading time of 1.66 s.



Local Process

```
Still Render,  0.11 seconds
```

Server, Process 0

```
Execute vtkImageReader id: 1386,  1.54206 seconds
```

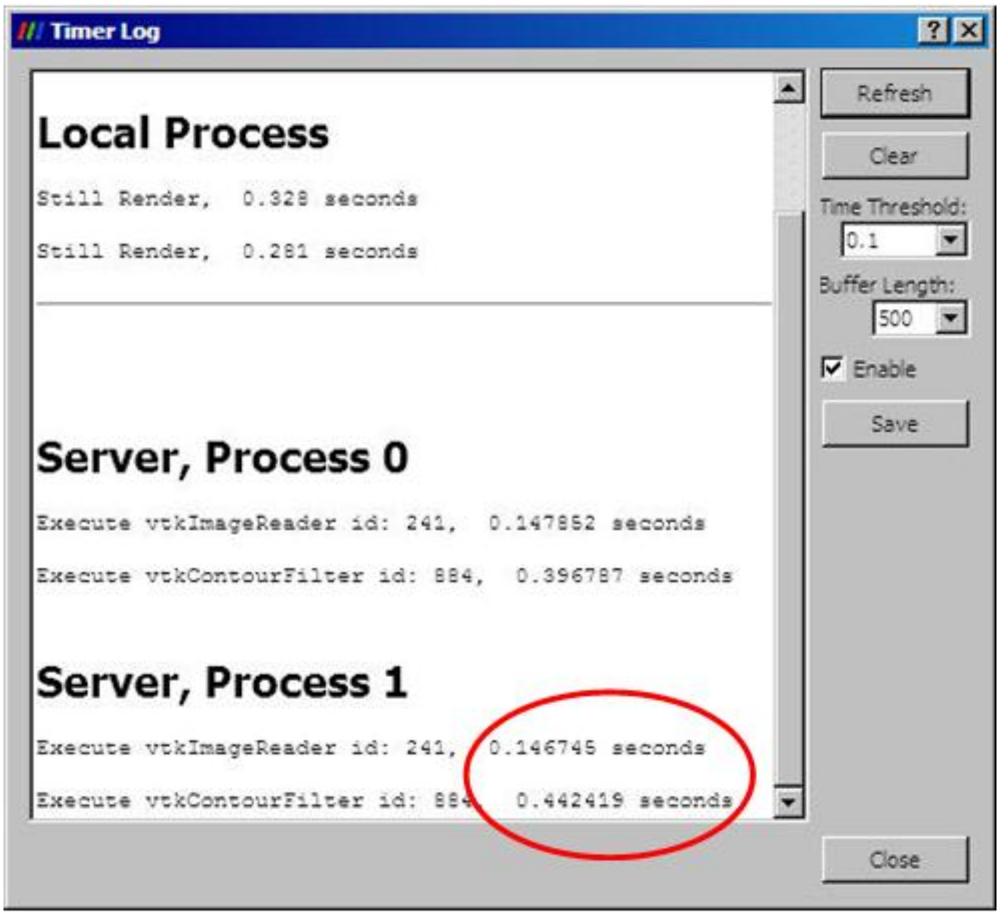
Server, Process 1

```
Execute vtkImageReader id: 1386,  1.55109 seconds
```

Figure 23: Timer Log for loading dataset.

5.2.3 Isosurface filtering times

An isosurface (contour) filter will be applied to the datasets. The ParaView built-in timer will be reset after loading the datasets. The isosurface filter is extremely computationally intensive which is suitable for performance comparisons between the desktop, Cloud VM on Lincoln Test Host, and Lincoln Hoiho Server. The processing time in the server and local processing times will be recorded from the ParaView client's Timer Log as shown in Figure 24.



```
Local Process
Still Render,   0.328 seconds
Still Render,   0.281 seconds

Server, Process 0
Execute vtkImageReader id: 241,   0.147852 seconds
Execute vtkContourFilter id: 884,   0.396787 seconds

Server, Process 1
Execute vtkImageReader id: 241,   0.146745 seconds
Execute vtkContourFilter id: 884,   0.442419 seconds
```

Time Threshold: 0.1

Buffer Length: 500

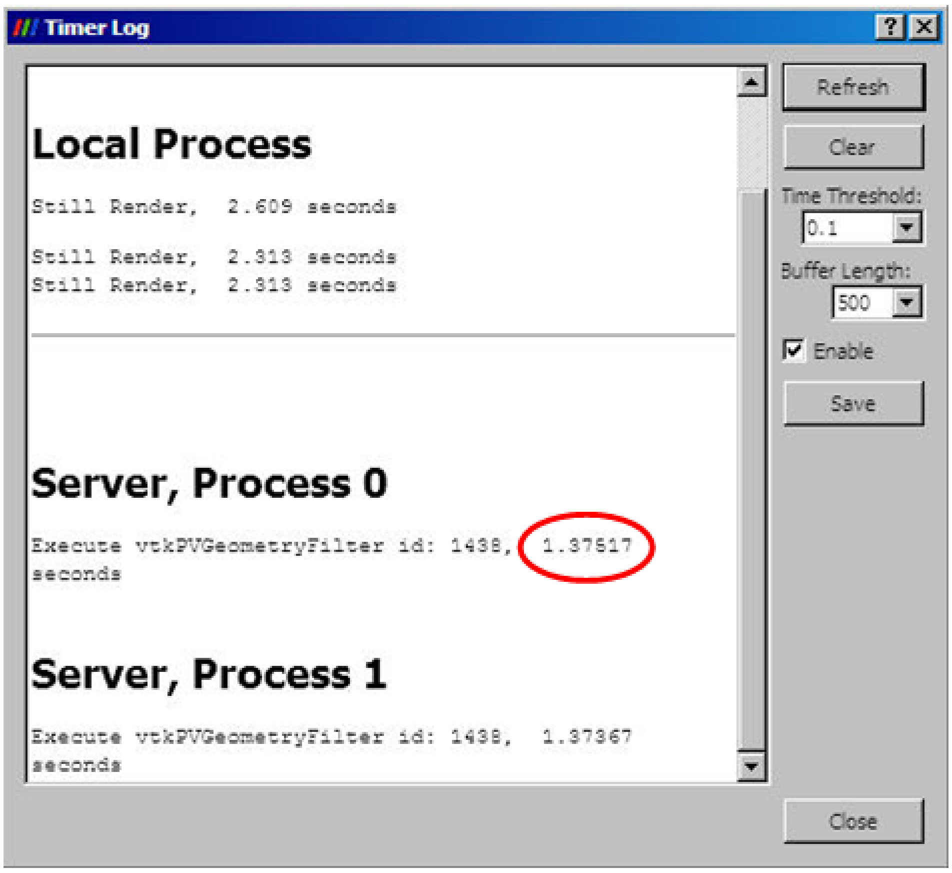
☑ Enable

Figure 24: Timer Log for isofiltering dataset.

Figure 24 shows the screenshot of the ParaView client's Timer Log and how the time will be calculated for filtering datasets using the isosurface filter. For the server processes, the longest time is for Process 1 (0.147 s + 0.442 s = 0.59 s) and this will be added to the local process time (0.328 s + 0.281 s = 0.61 s) for a total processing time of 1.20 s.

5.2.4 Data rendering times

By default when ParaView loads a 3D volume dataset, it shows an outline of the final image. Once the datasets have been loaded, the outlines will be rendered as a wireframe. This process shows all the edges of the cells in the datasets as lines. For the client/server tests, this method was performed on the server before the wireframe image was displayed on the client side. This test used less computation compared to isosurface filtering. The processing time in the server and local processing times will be recorded from the ParaView client's Timer Log as shown in Figure 25.


```
Local Process

Still Render,  2.609 seconds

Still Render,  2.313 seconds
Still Render,  2.313 seconds

Server, Process 0

Execute vtkPVGeometryFilter id: 1438,  1.37517
seconds

Server, Process 1

Execute vtkPVGeometryFilter id: 1438,  1.37367
seconds
```

Time Threshold: 0.1
Buffer Length: 500
☑ Enable

Figure 25: Timer Log for rendering dataset.

Figure 25 shows the screenshot of the ParaView client's Timer Log and how the time will be calculated for rendering datasets. For the server processes, the longest time is for Process 0 (1.38 s). This will be added to the local process time (2.609 s + 2.313 s + 2.313 s = 7.24 s) for a total processing time of 8.62 s.

5.2.5 Test environment

The performance tests will be carried out under the test environment as shown in Table 2. The resources used for the testing are the desktop PC, the Cloud VM (running on the Lincoln Test Host), and the Lincoln Hoiho Server. The machines will be linked with Lincoln University internal network with a speed of 100 Mb/s. For the client/server mode of ParaView, the ParaView client will be run from the desktop PC while the ParaView server is run in Cloud VM or Lincoln Hoiho Server.

Table 2: Test environment

	Desktop PC	**Cloud VM**	**Lincoln Hoiho Server**
Processor	Intel ® Core ™ 2 Duo CPU E8400 @ 3.00 GHz	Intel ® Core ™ i5 CPU 650 @ 3.20 GHz	Intel ® Xeon ™ CPU 2.66 GHz
Cores	2	4	8
Memory (GB)	4	3.7	8
Operating system	Windows XP Pro, SP 3 32 bits.	Ubuntu Server 10.04 64 bits	Ubuntu Server 9.04 64 bits

5.2.6 Summary

To assess the ease of use of Cloud visualisation from the point of view of a scientist, usability test will be carried out. For performance and scalability evaluation of the solution, multiple processors will be used and RAM will be increased during visualisation of different types of datasets. Finally, the performance of visualisation via client/server mode in a Cloud environment with stand-alone mode using a desktop will be compared.

Results may vary due to differences in hardware specifications, for example, CPU speed in the desktop PC, Cloud VM (on the Lincoln Test Host), and the Lincoln Hoiho Server. For all tests in client/server mode, the off-screen rendering will be used in the Cloud VM on the Lincoln Test Host and the Lincoln Hoiho Server, unless otherwise stated.

Chapter 6

Results and Discussions

6.1 Usability test

The usability test evaluated the ease of use of running ParaView, by comparing the process of running ParaView in stand-alone mode, and the process of the ParaView client connecting to the ParaView server on the Lincoln Hoiho Server with the Cloud environment.

ParaView stand-alone mode

To run ParaView in the standard desktop environment (i.e. not client/server mode), users just need to start the ParaView, load the dataset, and perform the visualisation. The steps are fairly easy and straightforward.

ParaView client/server mode

In ParaView client/server mode, users need to learn how to use SSH to connect to the ParaView server. The left column of Table 3 illustrates the additional steps required. Users also have to learn how to configure the ParaView client for client/server mode and how to start the server with the correct configuration (such as number of processes and off-screen rendering). ParaView client/server mode requires more steps compared to stand-alone mode; however, with simple instructions, most users will have no problem in carrying out the additional steps.

Once the scientists can run the ParaView in client/server mode, it would be easy for them to run it in a Cloud environment as the steps are similar (see the right column of Table 3). To run ParaView in a Cloud environment, users need to learn how to provision the VM and remotely login to the Cloud. The rest of the steps are the same for client/server operation once the VM is provisioned.

Table 3: Steps taken for ParaView client to connect to ParaView server

Lincoln Hoiho Server	Cloud VM in Aotea Cloud
1. SSH to Lincoln Hoiho Server.	1. SSH to Nimbus.
2. Start ParaView server.	2. Request to provision Cloud VM.
3. Start ParaView client.	3. SSH to the Cloud VM.
4. ParaView client connect to ParaView server.	4. Start ParaView server.
	5. Start ParaView client.
	6. ParaView client connect to ParaView server.

Although the Cloud VM requires more steps to deploy, it is clear that both the Lincoln Hoiho Server and the Cloud VM would be similarly easy to use. However, as the Cloud VM was not successfully run in the Aotea Cloud, networking issues (such as firewall settings and port forwarding) could not be tested.

6.2 Performance tests

Performance tests performed were:

1. time to load the datasets,
2. time to filter isosurface of the datasets, and
3. time to render wireframe of the datasets.

The time recorded was taken from the ParaView client's Timer Log. The local process shows the time to render datasets using a high level of detail at full resolution (still render), which includes time to transfer from the server to the client. The server process shows the process on the server side depending on how many processes were chosen. If more than one process was used, the longest time taken for any of the processes to run in parallel was recorded.

6.2.1 Data loading times

Data loading times in single-threaded and multi-processes operations are presented in Figures 26 and 27, respectively. The time was measured for reading the datasets and displaying the outline in the ParaView client. In client/server mode, the single-threaded and multi-processes operations differed from each other in their command. For single-threaded operation, the command is “pvserver” and it uses only 1 processor. For multi-processes operation, the command is “mpirun –np N pvserver” where N is the number of processors to run the processes. It may use 1 or more processors depending on the allocated resources.

Data loading times in single-threaded operation

The datasets were loaded on three machines with ParaView running in single- threaded operation. The time to load each dataset is shown in Figure 26. The Cloud VM was the fastest to load the datasets, followed by the desktop. This was because the Cloud VM had better CPU, at 3.20 GHz, compared to 3.00 GHz on the desktop and 2.66 GHz on the Lincoln Hoiho Server. A further test was carried out where the Cloud VM memory was reduced from 3.7 GB to 512 MB RAM. Time to load the medical datasets was the same, indicating that RAM did not affect loading time.

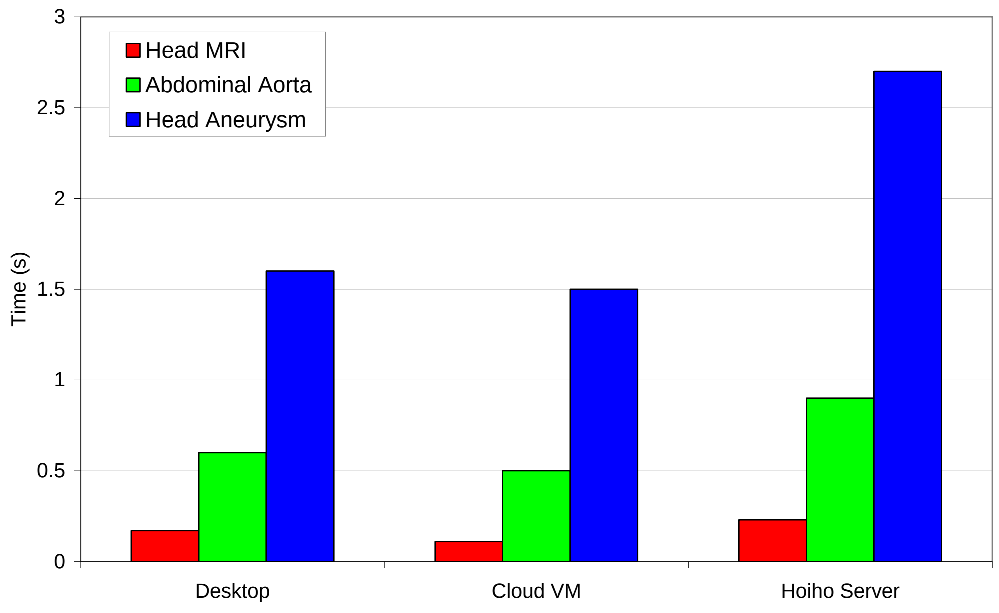


Figure 26: Time to load medical datasets on a single-threaded mode.

Data loading times in multi-processes operation

The datasets were loaded on the Cloud VM and the Lincoln Hoiho Server with ParaView server running in multi-process mode with off-screen rendering. The loading time of each dataset for a specific number of processes is shown in Figure 27.

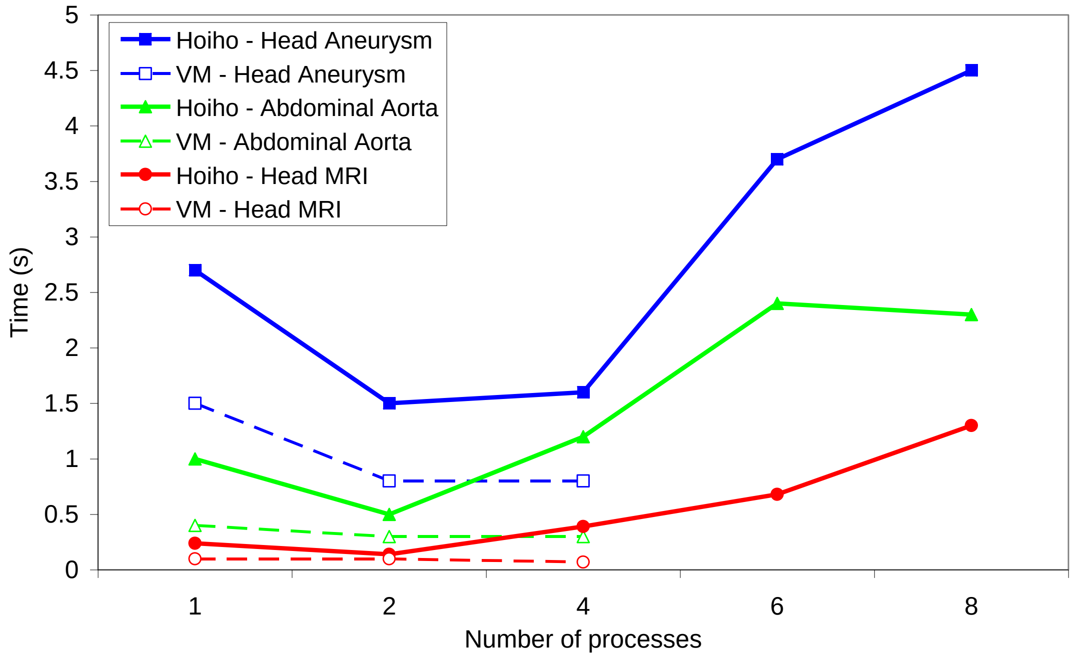


Figure 27: Time to load medical datasets using different number of processes.

Figure 27 shows an increase in loading time when more than two processes were used in the Lincoln Hoiho Server compared to a smaller number of processes. The performance was degraded probably because running in parallel added the overhead of inter-process communication for the datasets. Loading time using the Cloud VM was not different when using two or four processes.

6.2.2 Data filtering time

The desktop was able to perform the isosurface filtering only on the Head MRI. It crashed when trying to filter the Abdominal Aorta and Head Aneurysm datasets. The data filtering time of Head MRI for the desktop, Cloud VM, and Lincoln Hoiho Server is shown in Figure 28.

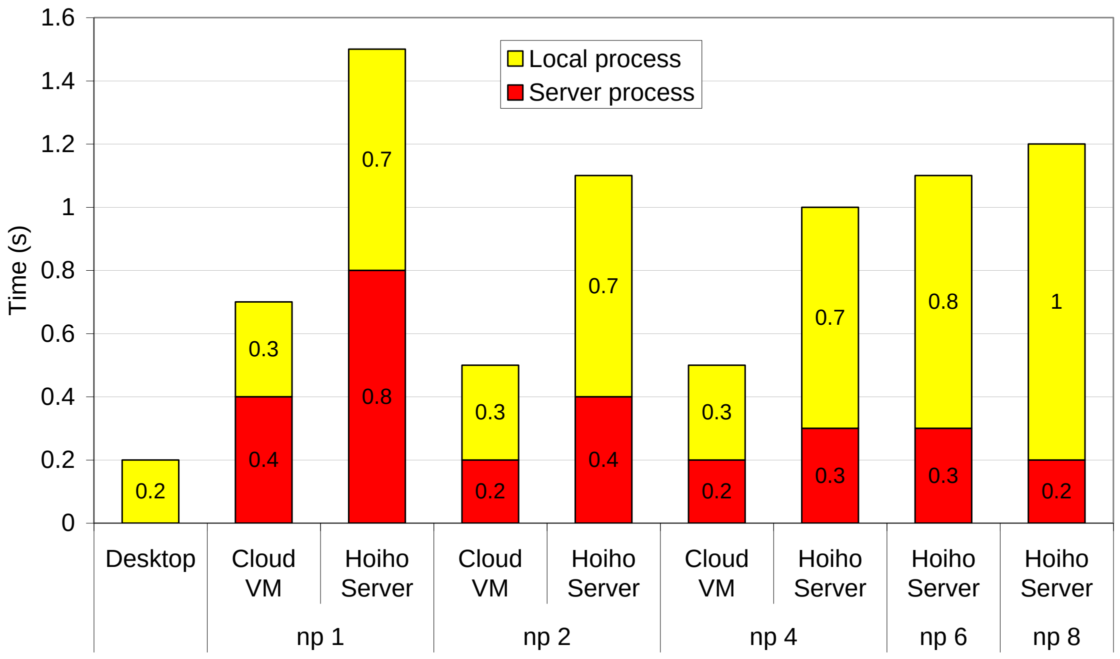


Figure 28: Isosurface filtering time of Head MRI using Desktop, Cloud VM, and Lincoln Hoiho Server. np indicates number of processes.

Figure 28 shows the improved performance for both Cloud VM and the Lincoln Hoiho Server when the processes were increased from one to two and four processes. However, there was a decline in performance when the number of processes (np) was increased further. From np 1 to np 2, the time for the server processes in the Cloud VM and the Lincoln Hoiho Server were reduced by half due to efficient distribution of processes. For smaller datasets, it maybe best to perform the visualisation locally i.e. stand-alone mode as findings from this study showed that increasing the np would not increase the performance. Based on this result, it is faster to perform isosurface filter function on desktop than client/server mode.

A further test was performed using 1, 2, and 4 processes with different memory allocations at 512 MB, 1-, 2-, and 3.7-GB. The server processing time decreased significantly by half from np 1 to np 2 for all memory allocations. However, compared to np 2, there was not much decrease in server processing time at np 4. The dataset did not require much RAM because of its small size; therefore, there were no differences when either 512 MB or 3.7 GB RAM were used.

The isosurface filtering for the Head Aneurysm on the desktop and the Cloud VM could not be performed as there was insufficient memory to process the data. Similarly, it could not be performed efficiently on the Lincoln Hoiho Server as it took more than 382 s to produce the final image when one processor was used, but the computer then froze. The process produced an additional 114 million cells and used up to 5200 MB RAM. When 4 or 8 processes were used, it took 351 s and 174 s, respectively, to visualise as shown in Figure 29.

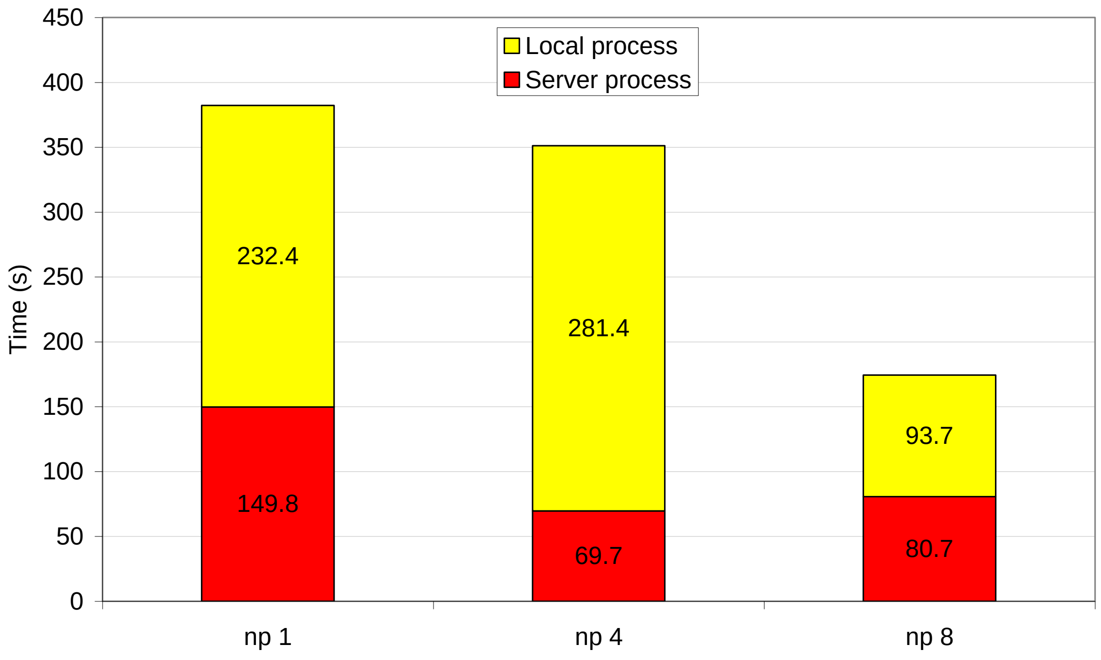


Figure 29: Isosurface filtering time of Head Aneurysm using Lincoln Hoiho Server. np indicates number of processes.

The time taken to perform the isosurface filter was reduced as the number of processes increased. The time taken for processing on the server increased going from np 4 to np 8 most probably because of insufficient shared memory. However, overall time for the np 8 was still fastest as less time was needed to transfer the image from the server to the client and display it. For np 8, 8 CPU cores processed the datasets and the same number of processes sends the image to the client concurrently, hence faster time for local process.

The data filtering time for the Abdominal Aorta using different number of processes in Cloud VM is shown in Figure 30.

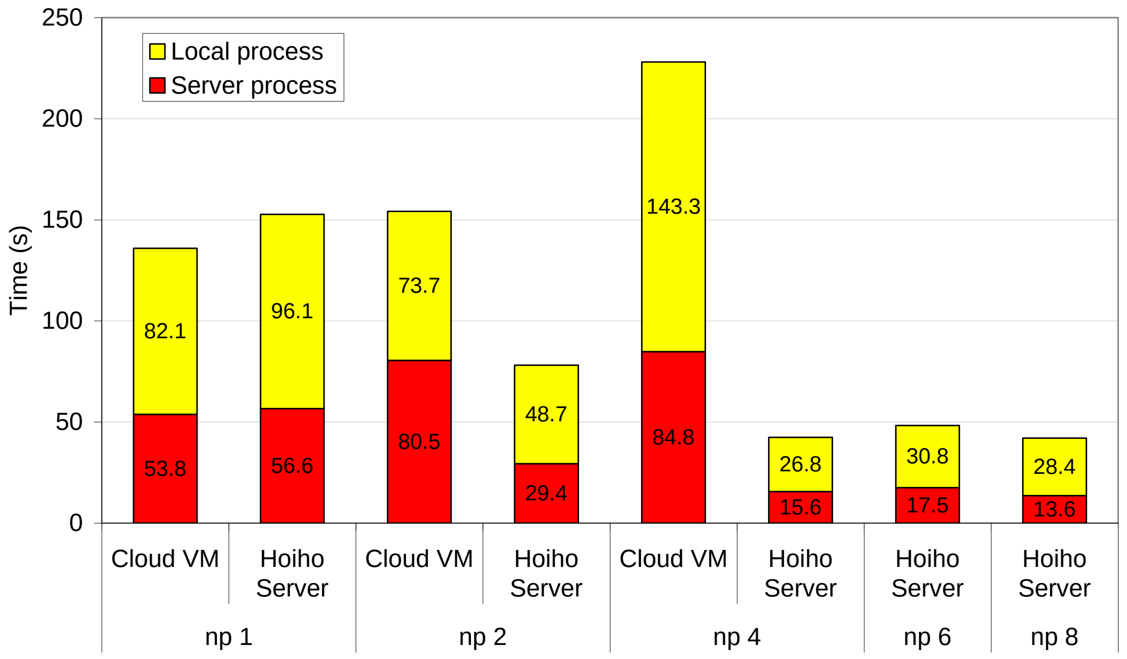


Figure 30: Isosurface filtering time of the Abdominal Aorta using different number of processes on Cloud VM. np indicates number of processes.

The isosurface filtering time for the Cloud VM was increased when the number of process was increased. This is due to the limited shared memory of 3.7 GB among the processes, which slows the filtering when the number of processes is increased. The initial Abdominal Aorta dataset has 45 million cells and used up 45 MB of RAM. After the isosurface operation, it produced a further 48 million cells and used up another 2200 MB of RAM.

On the other hand, the isosurface filtering time at the Lincoln Hoiho server generally decreased with an increasing number of processes. This is due to the larger memory in the Lincoln Hoiho Server compared to the Cloud VM (8 GB verses 3.7 GB).

6.2.3 Data rendering time

Data rendering time on single-threaded operation for the desktop, Cloud VM, and the Lincoln Hoiho Server is shown in Figure 31.

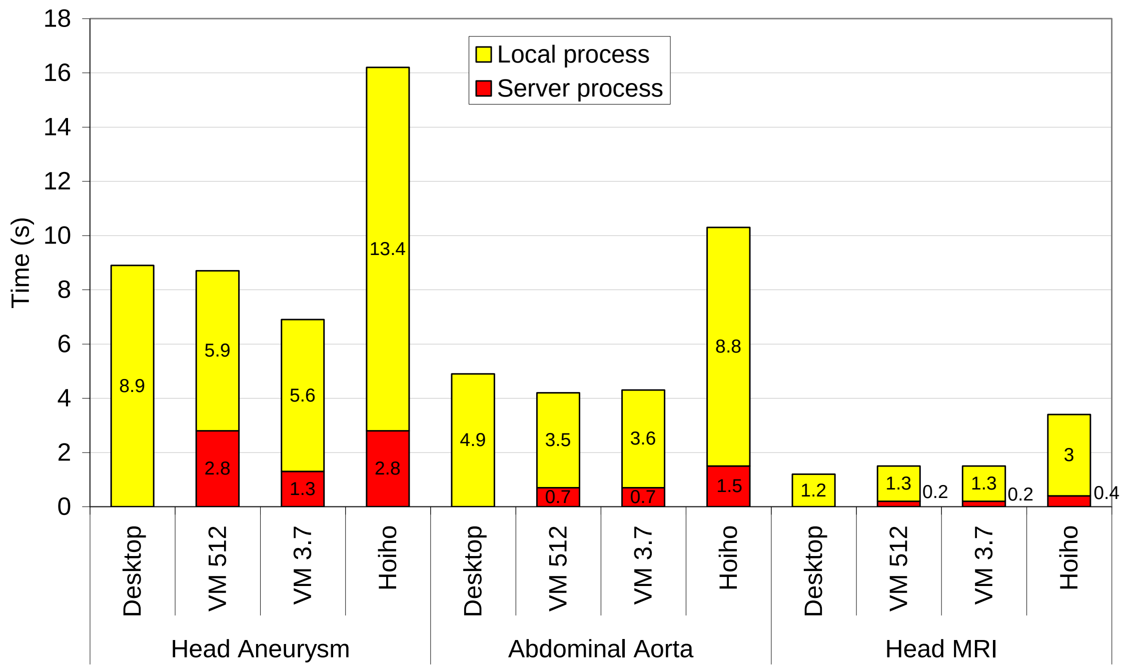


Figure 31: Data rendering time on single-threaded operation.

The Lincoln Hoiho Server took longer to render under single-threaded operation compared to the desktop and the Cloud VM. The rendering time was also longer using the Lincoln Hoiho Server when compared with the Cloud VM running with the RAM of 512 MB, indicating that RAM has little influence on data rendering time. The local processing time using the Lincoln Hoiho Server was also longer compared to the desktop and the Cloud VM.

It was not clear as to what caused these unanticipated results. A possible factor for the slower local process with the Lincoln Hoiho Server is the slower processing power of 2.66 GHz compared to the 3.20 GHz at the Cloud VM. Another possible contributing factor could be the network speed. To investigate the causes for the increased local processing time using the Lincoln Hoiho Server, further tests were performed. A network speed performance test between the Lincoln Hoiho Server and the desktop (as client) was conducted using Secure Copy (SCP) protocol to transfer 256 MB of data from the Lincoln Hoiho Server to the desktop. Five tests were conducted and the transfer speeds for the tests were 8.0, 10.7, 10.7, 10.7, and 10.7 MB/s. Next, a similar test was conducted from Test Host VM to the desktop. The transfer speeds were 11.1, 11.1, 10.7, 11.1, and 11.1 MB/s. The upper limit of transfer speed for Lincoln internal network is 12.5 MB/s (100 Mb/s). These findings suggest that the increased local processing time may be due to the delay in data transmission from the Lincoln Hoiho Server to the desktop. Local processing time was faster with the Cloud VM, possibly due to a faster transfer speed.

Figure 32 illustrates the rendering time for Head Aneurysm using different memory allocation and number of processes for the Cloud VM.

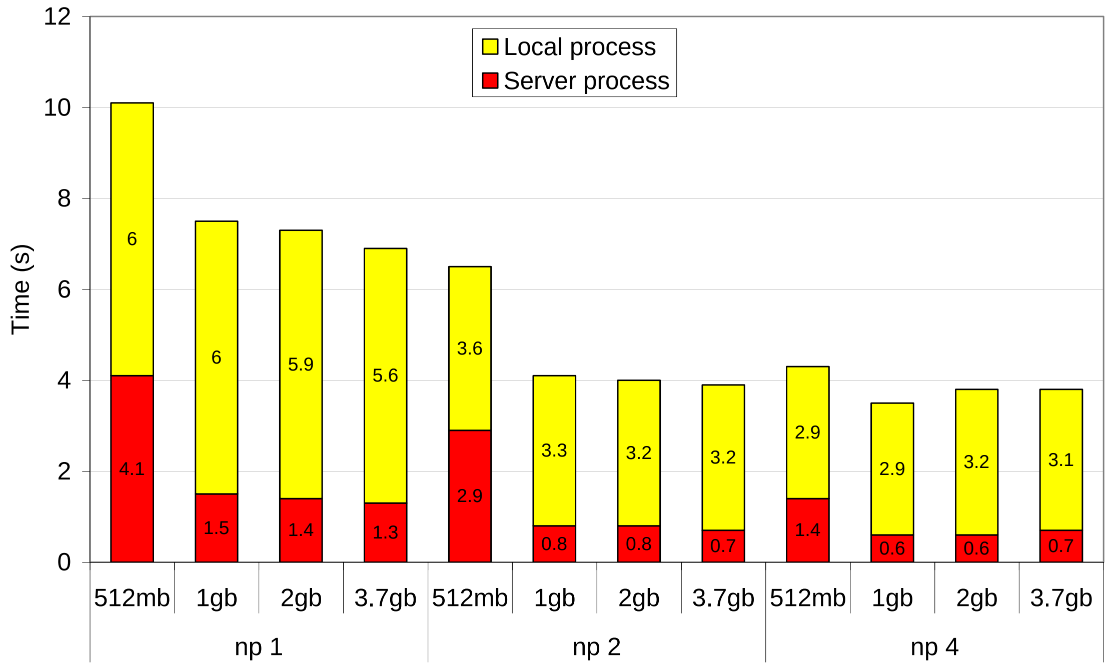


Figure 32: Time to render Head Aneurysm using different memory allocation and number of processes for the Cloud VM. np indicates number of processes.

Results showed that increased RAM and number of processes can improve the rendering time on the server, as shown by the red column in Figure 32. When the number of processors was increased from np 2 to np 4, the total processing time was not significantly reduced, as compared to the differences seen when np 1 was increased to np 2. This may be due to the shared memory in parallel rendering. Where there is a lot of communication between processors, limited shared memory will cause a slowdown. Hence, adding more processes onto a shared memory will eventually have a diminishing return.

6.3 Summary

The usability tests demonstrated the ease of use of ParaView for visualisation on a desktop PC. Running ParaView remotely (in the client/server mode) would require users to learn a few additional steps. Although using the Cloud VM required more steps than the Lincoln Hoiho Server, results show that both implementations would be similarly easy to use.

Findings from the performance tests showed that a stand-alone desktop can manage a small dataset efficiently compared to client/server mode. However, larger datasets require more memory and numbers of processes to perform the visualisation. The scalability of a virtual machine where the allocation of memory and processors can be modified accordingly would enhance the ability to visualise large datasets. Increased RAM and number of processes may influence performance depending on the type and size of datasets and the ParaView operations (filtering, mapping, and rendering).

Chapter 7

Conclusions

Visualisation of large datasets is computationally intensive in terms of filtering, mapping, and rendering the datasets. An underpowered desktop PC may be unable to process the data or would take a lot of time to perform the visualisation. This would not be a satisfactory interactive visualisation experience for scientists exploring their data.

The goal of this project was to investigate if interactive visualisation was feasible in a Cloud computing environment. Therefore, a Cloud VM was created and deployed in a Cloud environment to visualise large datasets.

The objectives of this project were:

1. To investigate the ability to deploy a visualisation system in a Cloud environment.

2. To assess the ease of use of Cloud visualisation from the point of view of a scientist.

3. To assess the performance and scalability of the solution when multiple processors are used and RAM is increased.

4. To compare the performance of visualisation via client/server mode in a Cloud environment with stand-alone mode using a desktop.

Results from this study showed that the objectives of this project have largely been met. The Cloud VM could be deployed in a Cloud environment which offers scalable memory and processors for the VM to be utilised. The process of provisioning the Cloud VM via Nimbus was relatively straightforward and would be feasible for a scientist to carry out. Once provisioned, the ParaView server would be started and the ParaView client connected for an interactive visualisation session.

However, due to the failure of the Aotea Cloud networking, the idea of "on-demand" visualisation could not be tested. Despite the problems with Aotea Cloud, the approach has merit and should be tested in other Cloud environments.

Compared to the desktop PC, the Cloud VM could handle much larger datasets and performed visualisations which otherwise could not be done. The differences between running ParaView on the desktop and in the Cloud required just adding a few extra steps. Running the visualisation in the Cloud could greatly enhance the ability of the scientists to interactively explore large datasets. However, it would be important to use an appropriate cloud (e.g. a Science cloud) that provides the necessary HPC resources and scability.

Findings from the performance tests showed larger datasets require more memory and number of processes to perform the visualisation. However, the increases in number of processes and memory size do not always improve performance. Increased RAM and number of processes influence performance depending on the type and size of datasets and the ParaView operations performed, such as filtering, mapping, and rendering.

This study used datasets of up to 256 MB. Datasets of 256 MB are considered large by some literature, but not for some scientific problems. Therefore, the capability of the Cloud VM to render larger data, such as in the Gigabyte or Terabyte range, could be investigated in future work. Additionally, a better user interface (such as a web interface) could be developed for deploying and starting the ParaView server to improve the ease of use for scientists. Performance tests could also be conducted using a Cloud VM running on hardware with available graphics processors (without off-screen rendering) to compare with the existing performance tests (with off-screen rendering).

In conclusion, the Cloud computing environment can offer scientists the ability to carry out interactive visualisations on large datasets. A Science cloud environment can provide on-demand visualisation on HPC facilities which could hugely assist scientists to analyse their data. This option is also likely to be cheaper and more scalable than continually purchasing additional hardware (e.g. faster desktops or private clusters). Therefore, further investigation of the potential of visualisation in the Cloud is warranted.

References

Ahmed, A. A., Latiff, M. S. A., Abu Bakar, K., & Rajion, Z. A. (2007). Visualization Pipeline for Medical Datasets on Grid Computing Environment. *Proceedings of the International Conference on Computational Science and its Applications (ICCSA), 26-29 August 2007* (pp. 567-576). Kuala Lumpur, Malaysia.

Akioka, S., & Muraoka, Y. (2010). HPC Benchmarks on Amazon EC2. *Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 20-23 April 2010* (pp. 1029-1034). Perth, WA, Australia.

Benchmarks STAR-CD V3.20 (2008). Retrieved November 22, 2010 from: http://www.cd-adapco.com/products/STAR-CD/performance/320/

Bestgrid (2010, 4 February). *Running Jobs via Globus*. Retrieved from the BesTGRID website: http://technical.bestgrid.org/index.php/Running_Jobs_via_Globus

BlueFern (2007). *Technical Overview - University of Canterbury "Blue Fern".* Retrieved from the University of Canterbury website: http://www.bluefern.canterbury.ac.nz/UCSCuserdocs/UniversityofCanterburyBlueFernTechnicalOverview(2).pdf

Cedilnik, A., Geveci, B., Moreland, K., Ahrens, J., & Favre, J. (2006). Remote Large Data Visualization in the ParaView Framework. *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization, 8-10 May 2006* (pp. 163-170). Lisbon, Portugal.

Chengtong, L., Qing, L., Zhou, L., Junjie, P., Wu, Z., & Tingting, W. (2010). PaaS: A revolution for information technology platforms. *Proceedings of the International Conference on Educational and Network Technology (ICENT), 25-27 June 2010* (pp. 346-349). Qinhuangdao, China.

Condor (2010). *High Throughput Computing.* Retrieved November 28, 2010, from http://www.cs.wisc.edu/condor/

Dawoud, W., Takouna, I., & Meinel, C. (2010). Infrastructure as a service security: Challenges and solutions. *Proceedings of the The 7th International Conference on Informatics and Systems (INFOS), 28-30 March 2010* (pp. 1-8). Cairo, Egypt.

Dunigan, T. H., Jr., Vetter, J. S., White, J. B., III, & Worley, P. H. (2005). Performance evaluation of the Cray X1 distributed shared-memory architecture. *Micro, IEEE, 25*(1), 30-40.

Faraj, A., Kumar, S., Smith, B., Mamidala, A., & Gunnels, J. (2009). MPI Collective Communications on The Blue Gene/P Supercomputer: Algorithms and Optimizations. *Proceedings of the 17th IEEE Symposium on High Performance Interconnects, 25-27 August 2009* (pp. 63-72). New York, NY.

Fenn, M., Murphy, M., Martin, J., & Goasguen, S. (2008). An Evaluation of KVM for Use in Cloud Computing. *Proceedings of the 2nd International Conference on the Virtual Computing Initiative (ICVCI), May 2008.* RTP, NC.

Foster, I., Yong, Z., Raicu, I., & Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree Compared. *Proceedings of the Grid Computing Environments Workshop (GCE), 12-16 November 2010* (pp. 1-10). Austin, TX.

Garg, S. K., Buyya, R., & Siegel, H. J. (2010). Time and cost trade-off management for scheduling parallel applications on Utility Grids. *Future Generation Computer Systems, 26*(8), 1344-1355.

Geelan, J. (2009). *Twenty-One Experts Define Cloud Computing.* Retrieved November 22, 2010, from: http://cloudcomputing.sys-con.com/node/612375

Haber, R. B., & McNabb, D. A. (1990). Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. *Visualization in Scientific Computing, IEEE,* 74-93.

Hosking, R. (2010, 15 August). *Aotea - Auckland Research Cloud.* Retrieved November 28, 2010 from: https://wiki.auckland.ac.nz/display/CERES/Aotea+-+Auckland+Research+Cloud

Igable (2009, 27 November). *How Cloud Scheduler Fits into the Cloud / HPC Ecosystem*. Retrieved November 22, 2010, from: http://cloudscheduler.org/

Keahey, K., Figueiredo, R., Fortes, J., Freeman, T., & Tsugawa, M. (2008). Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. *Proceedings of the Cloud Computing and Its Applications (CCA),* Chicago, IL.

Keahey, K., Freeman, T., Lauret, J., & Olson, D. (2007). Virtual workspaces for scientific applications. *Journal of Physics: Conference Series, 78,* 012038. doi:10.1088/1742-6596/78/1/012038

KitwarePublicWiki (2010a, 10 August). *Data formats.* Retrieved November 22, 2010, from: http://www.paraview.org/Wiki/Data_formats

KitwarePublicWiki (2010b, 18 October). *ParaView: Build And Install.* Retrieved November 22, 2010, from: http://www.paraview.org/Wiki/ParaView:Build_And_Install.

KitwarePublicWiki (2010c, 28 May). *Setting up a ParaView Server*. Retrieved November 22, 2010, from: http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server

Lentz, D. (2010, 29 July). *32-bit and 64-bit.* Retrieved November 26, 2010, from https://help.ubuntu.com/community/32bit_and_64bit

Lizhe, W., Jie, T., Kunze, M., Castellanos, A. C., Kramer, D., & Karl, W. (2008). Scientific Cloud Computing: Early Definition and Experience. *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC), 25-27 September, 2008* (pp. 825-830). Dalian, China.

Marshall, P., Keahey, K., & Freeman, T. (2010). Elastic Site: Using Clouds to Elastically Extend Site Resources. *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 17-20 May, 2010* (pp. 43-52). Melbourne, Australia.

McReynolds, T., & Blythe, D. (2005). *Advanced graphics programming using openGL.* San Fracisco, CA: Morgan Kaufmann.

Mesa (n.d.). *The Mesa 3D Graphics Library*. Retrieved November 20, 2010, from http://www.mesa3d.org/download.html.

Moreland, K., Lepage, D., Koller, D., & Humphreys, G. (2008). Remote rendering for ultrascale data. *Journal of Physics: Conference Series 125 012096, 125*(1). doi:10.1088/1742-6596/125/1/012096.

Nam, S., Jeong, B., Renambot, L., Johnson, A., Gaither, K., & Leigh, J. (2009). Remote visualization of large scale data for ultra-high resolution display environments. *Proceedings of the 2009 Workshop on Ultrascale Visualization, 14-20 November, 2009* (pp. 42-44). Portland, Oregon.

Nimbus (2010). *Nimbus 2.6 - Bird's Eye View.* Retrieved November 22, 2010, from: http://www.nimbusproject.org/docs/2.6/summary.html

Nimbus (n.d.). *Introduction to Nimbus @ UC*. Retrieved November 22, 2010, from: http://www.nimbusproject.org/nimbus_cloud

Ogrizovic, D., Svilicic, B., & Tijan, E. (2010). Open source science clouds. *Proceedings of the 33rd International Convention, MIPRO, 24-28 May, 2010* (pp. 1189-1192). Opatija, Croatia.

OpenMPI (n.d.). *Open MPI: Version 1.4.3.* Retrieved November 22, 2010, from: http://www.open-mpi.org/software/ompi/v1.4/.

OpenNebula (2010). *Most Flexible and Innovative Enterprise-class IaaS Cloud Software.* Retrieved November 22, 2010 from: http://www.opennebula.org/

Razak, S. F. A. (2009). Cloud computing in Malaysia Universities. *Proceedings of the Innovative Technologies in Intelligent Systems and Industrial Applications, CITISIA, 25-26 July, 2009* (pp. 101-106). Kuala Lumpur, Malaysia.

SalesForce (2010). Retrieved November 22, 2010 from: http://www.salesforce.com/platform/cloud-infrastructure/

Schröder, H. (2009). High Performance Computing for Visualisation and Image Analysis. In H. Badioze Zaman, P. Robinson, M. Petrou, P. Olivier, H. Schröder & T. Shih (Eds.), *Visual Informatics: Bridging Research and Practice* (Vol. 5857, pp. 12-21): Springer Berlin / Heidelberg.

Shuai, Z., Xuebin, C., Shufen, Z., & Xiuzhen, H. (2010). The comparison between cloud computing and grid computing. *Proceedings of the International Conference on Computer Application and System Modeling (ICCASM), 22-24 October, 2010* (pp. V11-72 - V11-75). Taiyuan, China.

Squillacote, A. H. (2007). *The ParaView Guide: A Parallel Visualization Application*. Colombia: Kitware

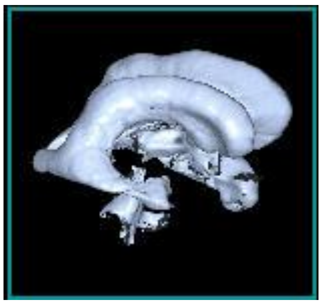
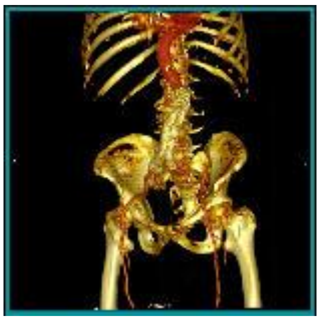
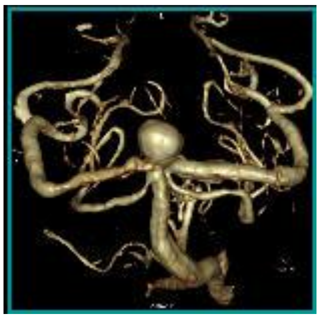
Tan, A. (2010). *Virtualization with Ubuntu 10.04 - Lucid Lynx*. Retrieved August 5, 2010 from: http://www.ideyatech.com/2010/05/virtualization-with-ubuntu-1004-lucid-lynx/.

Wernet, E. A. (2010). *Scalable and Distributed Visualisation using ParaView.* Retrieved November 20, 2010, from http://salsahpc.indiana.edu/tutorial/slides/0726/ParaView_Tutorial.pdf

Wong, K. (2009). Pictures in the Cloud. *Computer Graphics World, 32,* 42-47.

Appendix 1: Medical datasets used in this study

All datasets are binary, storing 8bit/16bit voxels for all slices, for all scanlines, and for all voxels. If 16bit voxels are used, they are stored in Least-Significant-Bit format (LSB), which is commonly used on PC platforms. Note that most medical datasets only use up to 12bits per voxel, so the upper four bits will be zero.

Dataset	Name Size, Bits per Voxel Spacing (mm)	Description
	Head MRI CISS 8Bits (8 bits set) 256 x 256 x 124 0.9, 0.9, 0.9	1.5T MRT 3D CISS dataset of a human head that highlights the CSF (Cerebro-Spinal-Fluid) filled cavities of the head.
	Stented Abdominal Aorta 16Bits (12 bits set) 512 x 512 x 174 0.8398, 0.8398, 3.2	CT Scan of the abdomen and pelvis. The dataset contains also a stent in the abdominal aorta. No contrast agent was used to enhance blood vessels.
	Head Aneurysm 16Bits (12 bits set) 512 x 512 x 512 0.1953, 0.1953, 0.1953	Rotational angiography scan of a head with an Aneurysm. Only contrasted blood vessels are visible.