Virtualising Visualisation

A distributed service based approach to visualisation on the Grid

Stuart Muir Charters

A Thesis presented for the degree of
Doctor of Philosophy

Visualisation Research Group
Department of Computer Science
e-Science Research Institute
University of Durham
United Kingdom

April 2006

Dedicated to

Lizzie my wife, for her love and support throughout my PhD.

My parents, Ann and Graeme for providing me with the opportunities that I have had in life.

Virtualising Visualisation

A distributed service based approach to visualisation on the Grid

Stuart Muir Charters

Submitted for the degree of Doctor of Philosophy

Abstract

Context: Current visualisation systems are not designed to work with the large quantities of data produced by scientists today, they rely on the abilities of a single resource to perform all of the processing and visualisation of data which limits the problem size that they can investigate.

Objectives: The objectives of this research are to address the issues encountered by scientists with current visualisation systems and the deficiencies highlighted in current visualisation systems. The research then addresses the question:

"How do you design the ideal service oriented architecture for visualisation that meets the needs of scientists?"

Method: A new design for a visualisation system based upon a Service Oriented Architecture is proposed to address the issues identified, the architecture is implemented using Java and web service technology. The implementation of the architecture also realised several case study scenarios as demonstrators.

Evaluation: Evaluation was performed using case study scenarios of scientific problems and performance data was conducted through experimentation. The scenarios were assessed against the requirements for the architecture and the performance data against a base case simulating a single resource implementation.

Conclusion: The virtualised visualisation architecture shows promise for applications where visualisation can be performed in a highly parallel manner and where the problem can be easily sub-divided into chunks for distributed processing.

Declaration

The work in this thesis is based on research carried out in the Visualisation Resarch Group, the Department of Computer Science, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Publications

The following publications describe describing this work as a whole or in part have been made:

1. Visualisation for Informed Decision Making; From Code to Components
   Stuart M. Charters, Claire Knight, Nigel Thomas and Malcolm Munro
   Proceedings of the Workshop on Software Engineering Decision Support
   14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, July 15-19 2002. pp. 765-772

2. Visualisation in e-Demand: A Grid Service Architecture for Stereoscopic Visualisation
   Stuart M. Charters, Nicolas S. Holliman and Malcolm Munro
   Proceedings of the 2nd UK e-Science All Hands Meeting, Nottingham, UK, September 2003.

3. Distributing Stereoscopic Scientific Visualisation across the Grid
   Stuart M. Charters, Nicolas S. Holliman and Malcolm Munro
   Proceedings of Computer Graphics and Imaging 2004, Kaua'i, Hawai'i, August 2004.

4. Visualisation on the Grid: A Web Service Approach
   Stuart M. Charters, Nicolas S. Holliman and Malcolm Munro
   Proceedings of the 3rd UK e-Science All Hands Meeting, Nottingham, UK, September 2004.

5. The e-Demand Project: A Summary

   Paul Townend, Jie Xu, Erica Yang, Keith Bennett, Stuart Charters, Nicolas Holliman, Nicholas Looker and Malcolm Munro

   Proceedings of the 4th UK e-Science All Hands Meeting, Nottingham, UK, September 2005.

Acknowledgements

This work could not have been completed without the support, help and advice of so many people.

Professor Malcolm Munro for supervising my PhD, for endless hours of his time and for giving me the opportunity to undertake this research.

Dr Nicholas Holliman for introducing me to stereoscopic displays, and stereo graphics, for supervising my PhD and giving me the benefit of his industrial experience.

Dr Nigel Thomas for providing the opportunity to stay at Durham and undertake research that led to this PhD.

To my examiners Professor Ken Brodlie and Professor Nick Avis for their time in reading this thesis.

To all the members of the Visualisation Research Group, e-Science Research Institute and the Department of Computer Science for the discussions, socials and coffee room breaks!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visualisation is a communication mechanism allowing humans to gain insight into information. It has been steadily rising in impact over the course of its development, from early hand drawn diagrams to the latest computer generated interactive graphics. The size and range of problems to which it is applied is growing, as is the extent to which its use is pervading the sciences.

A new novel architecture, the focus of this thesis, is researched to address the issues that arise from the growth in size of problems and from the widespread use of visualisation.

The research fuses the areas of visualisation, stereoscopic display technology and grid technology together providing a solution capable of meeting the ever increasing demands of scientists and other users.

An overview of these areas with an outline of the objectives of the research and an introduction to the remainder of the thesis are presented.

1.1 Scenario

Imagine if you will the following situation:

A scientist working in the United Kingdom in his office is studying the creation of the universe and how it has developed since the big bang. In order to do this he has written a simulation, this simulation requires a large computational resource to compute; as he does not have such a resource available at his home institution he

runs the simulation on a resource at another institution, for example, in Europe.

The simulation is complex with a large number of entities, it takes a long time to complete, however at various points it outputs the current status of the simulation to files on disk. As the simulation contains a large number of entities it produces large files when the status is written to file. These files would take a large amount of time to transmit even across a high speed network, despite this the scientist wishes to monitor the simulation to ensure that it is progressing as it should. The scientist only needs to see a high level overview of the data to see how the simulation is progressing.

When the simulation is completed the scientist wishes to view the results of the simulation at a high level until an item of interest is found. The scientist then wishes to view that part of the result space in more detail. On finding a result that requires more investigation he wishes to view that result on a device that can display in stereo as the simulation is of a 3D phenomena and it is important for the scientist to have a clear view of the interactions occurring to cause the phenomena. On viewing the result in stereo the scientist decides he wishes to consult another scientist, a professor based in the United States. The difference in time zones is such that the scientists cannot effectively collaborate at the same time. The professor is currently very busy and can only fit in examining the result around her other commitments. The American professor has different stereo equipment to the UK scientist so a pre-generated video of the result could not work as it would not work correctly on the stereo equpiment. They cannot send the large amount of data so the professor can recalculate the visualisation, even if the data was small enough to send the professor does not have the time to recreate the visualisation.

A possible solution to this scenario is a visualisation system which will allow the extraction of high level data from a remote computer, allow visualisation at various levels of detail, dynamically generate images so as to cope with multiple types of stereo display equipment and support asynchronous collaboration.

1.2 Visualisation

Visualisation is concerned with providing insight into data, where that data is numerical and generated from experimentation or simulation it is termed scientific visualisation. Where different types of data, such as text, images and numbers, are brought into visualisation this is termed information visualisation.

Visualisation is a diverse area of research and the term itself is heavily overloaded. Visualisation as a term can be divided into three main areas:

Visualisations, visualisation technology and the visualisation process.

Visualisations relate to the visual representations and graphical output produced.

Visualisation technology relates to the software used to produce the visualisation.

The visualisation process is the procedures used to take raw data and convert it into a visual representation that a user can view and interact with.

The two main categories of visualisation, information visualisation and scientific visualisations, are examined as applied areas of research in chapter two. Visualisation technology encompasses the software solutions available to build visualisations for use by the end user. These systems include libraries, visualisation environments and distributed frameworks. These areas are all explored in greater detail in chapter two along with current and future trends in visualisation.

1.3 Stereoscopic Displays

We experience the world around us in stereo, we hear with two ears and see with two eyes. The world of computer displays has traditionally been mono, showing both 2D and 3D images with no perceived depth. Advances in display technologies now allow three dimensional images to be displayed stereoscopically.

A multitude of different stereo displays exist, each is different and requires custom content to get the best from them. Autostereoscopic displays are the main focus of this review, these displays allow the viewer to see stereoscopically without the need for special head wear such as shutter glasses. Autostereoscopic displays are available in a variety of formats including single user desktop displays and multi user presentation displays.

Images for display on stereoscopic displays must be generated specially, the mechanism for generating them involves creating two versions of the image, one tailored for the left eye and one for the right eye. Typically these two images are generated using multiple cameras in rendering a scene. Chapter Three discusses stereo display technology and image generation more fully.

1.4 Grid Computing

There is an increasing need and desire to leverage geographically disparate high performance computing facilities to allow larger problems to be solved and to increase the utilisation of these resources, one solution to this is Grid computing.

The Grid computing vision sees institutional boundaries being broken down and computing resources being freely traded to solve large scale problems.

To achieve this vision an underlying software infrastructure to manage security, storage, communication and computational resources needs to be in place. A batch processing solution and a service oriented solution to the infrastructure problem have been developed. Chapter Four discusses these solutions in more detail.

1.5 Objectives

The objective of this research is to provide a mechanism by which a visualisation can be viewed locally by a user, for example on their desktop computer, but that can harness remote resources to perform the resource intensive parts of the visualisation.

Increasing scientific study is carried out by both simulation, insilico experimentation and real world observation and experimentation. This scientific study is increasingly collecting greater quantities of data, often of the order of terabytes and beginning to tend towards petabytes. To analyse these quantities of data requires significant computational assistance. One such method, and perhaps the only method that is suitable for such large data sets is that of visualisation, the use of graphics and interaction, occasionally coupled with audio augmentation, to represent data and allow scientists to explore that data.

Despite a trend for large scale visualisation installations, such as CAVEs and RealityCenters in the past, the desire for local visualisation has increased as visualisation has become more of a 'normal scientific analysis activity'. This is reinforced by the views of users of visualisation services at insitutions with comments such as "prefer local visualization" and "I do visualization locally." [NERSC, 2002]

The computational ability of CPUs has increased greatly, in accordance with Moore's Law. However this increase has been insufficient to allow the processing of large amounts of data quickly. Supercomputing resources, combining many hundreds or thousands of processors provide one solution but the demand for these resources can be great leading to large lead times for computational jobs.

1.6 Criteria For Success

This section outlines criteria that the completed research can be evaluated against to determine how far it has gone to meeting its objectives. These criteria will be revisited and discussed in the final chapter of this thesis.

The criteria for success are an architecture that:

1. *Allows a visualisation to be displayed on a desktop display.*

   This criteria means that the architecture should allow the output, a visualisation, to be displayed on a desktop display attached to a PC.

2. *Runs across multiple computers/resources.*

   This criteria means that the architecture should be able to be deployed and executed on multiple different computers at the same time, this means that computers that are designed for a specific job can be used for that job.

3. *Performs visualisation of a data set*

   This criteria means that the architecture developed should allow a data set to be analysed and processed such that a visualisation is produced.

4. *Supports interaction within the visualisation.*

This criteria states that the architecture designed should be able to support interaction as part of the visualisation process.

5. *Supports collaborative visualisation.*

   This criteria means that the architecture should provide support for collaborative visualisation where participants are geographically distributed.

6. *Supports multiple display types, including autostereoscopic desktop displays.*

   This criteria means that the architecture should be capable of supporting multiple display types, of these display types that can be supported autostereoscopic displays should be explicitly supported.

7. *Supports multiple visualisation types*

   This criteria means that the architecture should not be specialised to support a single branch of visualisation, for example, scientific visualisation or information visualisation, but should as far as is possible be generalised to support multiple types of visualisation.

1.7 Major Contributions

The major achievements of the research have been:

- Definition of a Service Oriented Architecture for Visualisation on the Grid
- Implementation of the Service Oriented Architecture
- Real World Case Study Scenarios to demonstrate the architecture
- Performance Analysis of a Service Oriented Architecture
- Identification of areas for Future Work for both Visualisation and Visualisation Architectures

1.8 Thesis Overview

Chapter One contains an introduction to this thesis, outlining criteria for success and the objectives of this research. The next chapters look at three areas of research, visualisation, stereo and grid computing. Chapter Two focuses on Visualisation, looking at types of visualisation and the mechanisms through which they are achieved. Chapter Three looks at stereo graphics and display technologies. Chapter Four provides a look at Grid computing, what the vision is and how that has been realised.

Chapter Five presents the design of the virtualised visualisation architecture with Chapter Six describing the implementation of that design.

Chapter Seven presents an analysis of the performance of the architecture described in Chapter Six through experimentation and problem scenarios. Chapter Eight uses the performance information to evaluate the architecture and finally chapter Nine presents conclusions about the architecture.

1.9 Summary

This chapter has presented a brief introduction to three areas, visualisation, stereo and Grid computing along with the objectives for this research including the criteria for success.

Chapter 2

Visualisation

Visualisation is an exciting and expansive area of research, it combines the areas of computer graphics, problem solving and software engineering. There is both an art and a science behind good visualisation.

This chapter examines the art and science of visualisation and the technology behind the generation of visualisations.

2.1 Overview

Visualization has been a corner stone of computing from its earliest days [Brodlie et al., 2004b] and provides a mechanism for users to gain insight into data they are investigating.

A definition of visualisation (alternatively spelt visualization) is:

> Visualisation is "the ability to present information visually that is rapidly assimilated by human observers, and transformed into understanding or insight."
> [Bethel et al., 2003]

This definition is reinforced by the following quotations as to the purpose of visualisation.

> "The purpose of computing is insight not numbers" [Hamming, 1962]

For visualisation we can go as far to say that

"The purpose of visualisation is insight not pictures"

Ben Shneiderman quoted in Scientific American [Beardsley, 1999].

The purpose of visualisation therefore is insight, the mechanism to achieve that insight is pictures, those pictures are built from data, often numbers.

Therefore a high level definition of visualisation can be stated thus:

Visualisation is a high level interpretation mechanism for data.

This view is reinforced by Globus and Raible in their paper "Fourteen Ways to say nothing with Scientific Visualization" [Globus and Raible, 1994] who show how to produce meaningless visualisations by concentrating on pretty pictures rather than ones which convey scientific meaning and insight.

There are many types of visualisation, the most prevalent is scientific visualisation which is concerned with the visualisation of data from scientific experiments and simulations, this data is usually numeric in nature. The second main area of visualisation is information visualisation, the data visualised in information visualisation includes text, images as well as numerical data.

The majority of visualisations are generated through the use of the visualisation pipeline the conceptual model of which was proposed by Haber and McNabb [Haber and McNabb, 1990] and is shown in figure 2.1

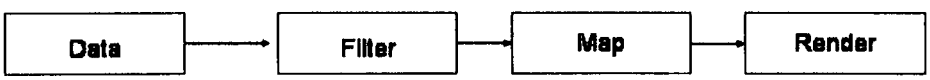


Figure 2.1: Conceptual Model of Visualisation Pipeline

The pipeline model is not always explicitly exposed in visualisations, however, each of the stages: data where the user data is loaded; filter where the data is manipulated; map where a representation of the data is generated and render where the final output is produced, are all used.

The pipeline model for visualisation is used as a basis for the design of many visualisation systems; the systems are discussed later in this chapter.

Ware [Ware, 2000] highlights five advantages of visualisation, these are listed below.

- Ability to comprehend huge amounts of data
- The perception of emergent properties
- Shows errors in the data easily
- Understanding large scale and small scale features.
- Facilitation of hypothesis formation

These advantages are shown through the following sections which discuss Information Visualisation and Scientific Visualisation as two broad categories in the visualisation topic.

2.2 Information Visualisation

> The use of interactive visual representations of abstract, nonphysically based data to amplify cognition. [Card et al., 1999]

Information Visualisation is concerned with the visualisation of data, including, text, images and numerical data. This area of visualisation subsumes many specialised areas of visualisation including software visualisation and medical visualisation.

The main focus of information visualisation is providing representations of abstract data sets. These data sets may be structured such as software source code or unstructured. Many of the data sets have no real world representation, therefore the aim of information visualisation is to provide a representation that shows the structure of the data and allows users to explore the data and gain insight into it.

Information Visualisation makes use of different types of representation. Many are abstract representations like that shown in Figure 2.2 whilst others map abstract data onto real world metaphors allowing users to make use of their existing knowledge to aid in the interpretation of the data.

The abstract visualisation shown in Figure 2.2 is designed to help users choose a new vehicle, it shows six variables relating to the vehicles. Each car model is represented by a glyph, this glyph is located by the values of each of the six variables. This results in similar vehicles being grouped together and the relationships between them being exposed. This grouping of similar vehicles is an example of the visualisation being used to show emergent properties in the data.

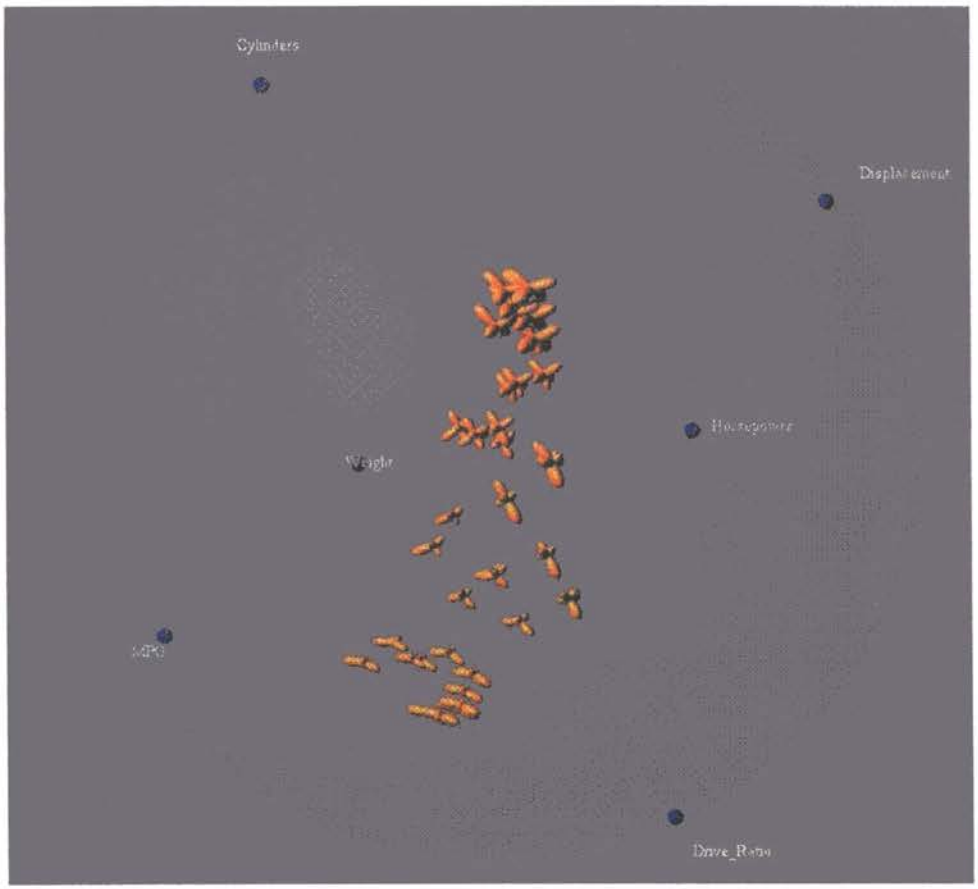


Figure 2.2: Abstract Visualisation for choosing a car [Theisel and Kreuseler, 1998]

A use of real world metaphors has been made by Charters et. al. [Charters et al., 2002] in mapping the properties of software components onto a cityscape metaphor as shown in Figure 2.3 this allows a user of the visualisation to use their pre existing knowledge of navigating a city to help them identify components that meet their requirements. The visualisation makes use of several sign post features to help users maintain their orientation within the visualisation and also contains a high level overview that allows users to select their initial area of interest.

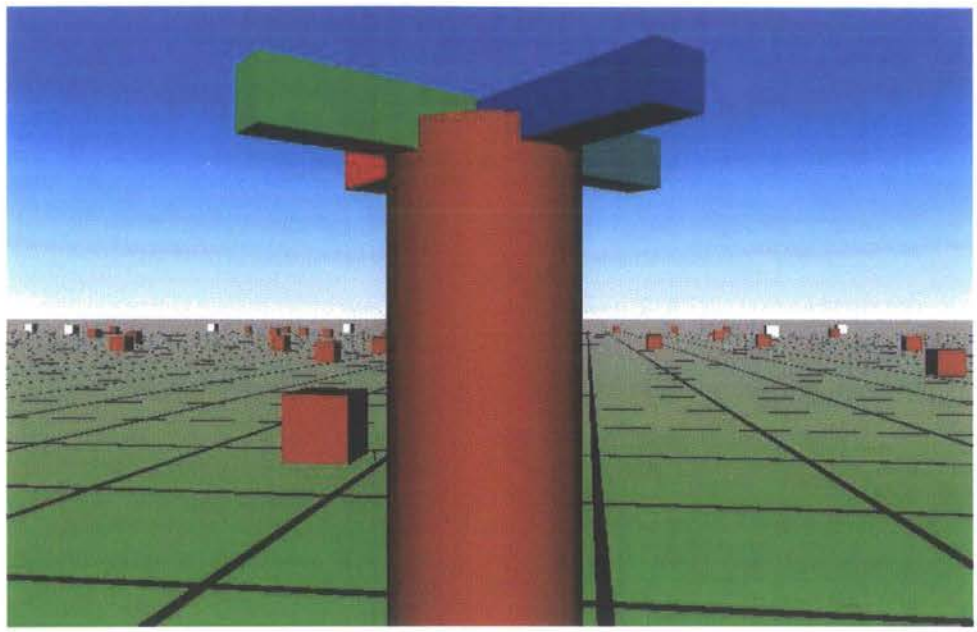


Figure 2.3: Visualisation of Software Components [Charters et al., 2002]

2.3 Scientific Visualisation

The use of interactive visual representations of scientific data, typically physically based, to amplify cognition. [Card et al., 1999]

This section examines the area of scientific visualisation. The area of scientific visualisation is the largest use of visualisation. It is driven by scientific data, usually numeric in nature, generated by scientific experiments and simulations.

Scientific Visualisation could be seen as simply a subset of information visualisation however the techniques used and the basis for the data used in scientific visualisation set it apart.

The data used in Scientific Visualisation is derived from scientific experiments and simulations. The visualisations generated are often deeply anchored to the physical nature of the phenomena being examined as this provides a base point of understanding to tie the visualisation too. The binding to the physical can however hinder the development of more informative, insightful and expressive visualisations that are abstract in nature. It has been argued that abstract representations are the future [Hanrahan, 2005], this may be the case but physical provides an ideal entry

point for investigations into and representations of phenomena.

The strong correlation between the physical and the visualisation provide a very good starting point for developing visualisations and choosing what techniques to use. Other areas of visualisation can suffer from the "blank canvas" problem where the data to be visualised has no identifiable physical form and as such an abstract form must be developed.

Visualisation to an extent has always been accepted by the scientific community, diagrams and pictures have long been used to convey information in scientific reporting. The challenge is to move from a desire to use visualisation as a pretty, artistic addition to the scientific investigative process, to as stated earlier, the use of visualisation as an integral part of the scientific investigative process to gain insight into the phenomena under examination.

Scientific visualisations that have been produced are examined to provide an overview of the nature of the resulting images from the visualisation process.

The visualisation of the asteroid Eros as shown in Figure 2.4 shows the physical shape of the asteroid with a false colour map superimposed across its surface. The false colour map ranges from red through yellow to green through blue highlighting the variations in the gravity slope across the asteroid. This visualisation shows the representation of an invisible phenomena on a physical representation of the object demonstrating how the abstract can be mapped onto the physical.

A technique known as splatting is shown in Figure 2.5 in this instance the wind direction across a land mass is being shown. The coloured arrows show the direction that the wind is travelling, the grey cloud showing the wind body. This visualisation shows another technique used in scientific visualisation exposing the range and diversity of techniques that can be applied.

2.4 Visualisation Systems

Visualisation Systems are the mechanism by which visualisations are produced. These systems range from libraries that can be used to build bespoke systems through to commercial offerings developed as general purpose visualisation tools.

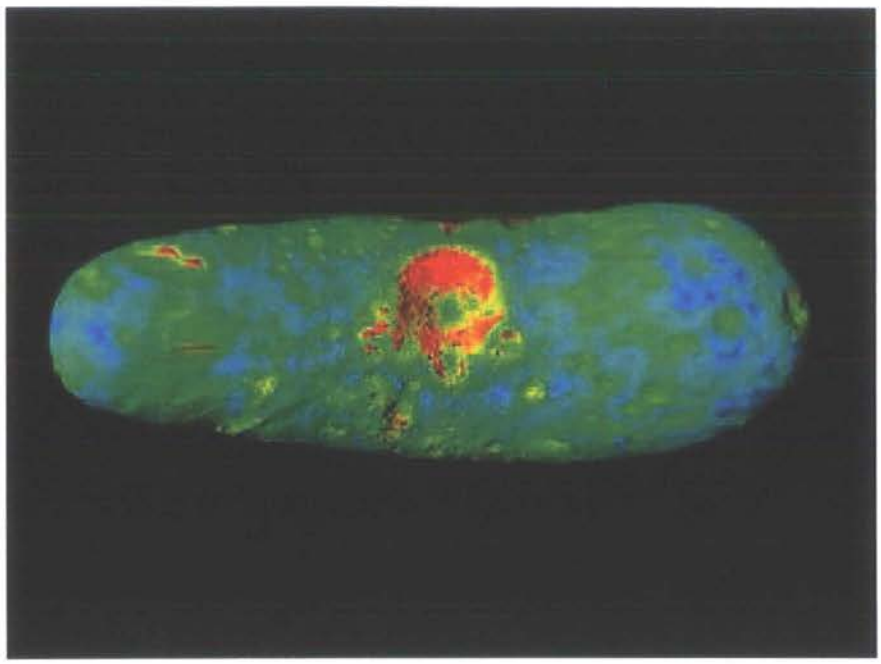


Figure 2.4: Gravity Slope on the asteroid Eros [NASA Goddard Space Flight Center]

Coupled with these are research developments pushing the boundaries and advancing the state of the art in visualisation. The visualisation systems surveyed are a representative sample of those available to visualisation researchers demonstrating the features that can be found in such systems such as:

- Composition
- Collaboration
- Remote Resource Use
- Cross Platform Operation

2.4.1 Visualisation Libraries

This section looks at some of the libraries that have been developed for the construction of bespoke visualisation applications, the Visualisation Toolkit (VTK) and VisAD are discussed.

Figure 2.5: Wind Direction shown through Splatting [Crawfis et al., 1993]

VTK

VTK [Schroeder et al., 2003] is an object oriented toolkit developed in C++ and designed for building visualisation applications. The visualisation pipeline approach is used, however in VTK the composition of pipelines is done programmatically rather than graphically.

The toolkit has modules that will run using technologies such as MPI allowing applications to be run in environments where multiple computers are combined within an institutional boundary, such as with a cluster computer.

Several front end solutions have been built using VTK, including ParaView from Kitware. These front end solutions allow the pipeline composition to be done graphically and take VTK into the realm of a modular visualisation environment.

The VTK libraries can support distributed applications with some modules allowing execution in parallel. This can help cope with large datasets but the memory on the host system is still a major limiting factor.

VTK does not support collaborative visualisation 'out of the box' as it is a collection of libraries but synchronously collaborative application could be developed.

VisAD

VisAD [Hibbard] is a Java component library used for the analysis and visualisation of numerical data. In the same way as VTK visualisations in VisAD are produced programatically but in VisAD the pipeline model is much less explicit. The VisAD library allows bespoke visualisation applications to be built, supporting distributed and collaborative visualisation features to be incorporated into those applications. VisAD concentrates its development on solutions to numerical visualisation problems and as such is less comprehensive than some other visualisation systems.

2.4.2 Environments

This section provides an examination of both Modular Visualisation Environments (MVEs) and of Problem Solving Environments (PSEs). MVEs and PSEs are software applications that allow users to construct visualisation pipelines from software modules. These applications typically run on a single computer, be it a desktop PC or a supercomputer.

Brodlie et al. present a examination of many historical and current MVEs and PSEs [Brodlie et al., 2004b]; an overview of one such MVE, IRIS Explorer is presented along with a look at SCIRun, a Problem Solving Environment.

IRIS Explorer

IRIS Explorer [Foulser, 1995] is an MVE that has benefited from both commercial development and academic research. The system was originally developed for the IRIX operating system before being spun out and developed for multiple platforms by NAG (http://www.nag.co.uk/). The system is based on a series of modules stored in a library, these modules can be combined together to form visualisation pipelines in a graphical user interface that allows a 'drag and drop' style of interaction.

The system can be extended by users developing their own modules for the system which can be imported to the library.

The system has been extended to allow collaborative visualisation through the COVISA project [Wood et al., 1997], now part of the standard distribution. Collab-

oration is achieved by one user exporting the output of one of their modules, and another user importing this into the pipeline running on their system. This requires that both users have a copy of IRIS Explorer running and that they are performing, different place, same time collaboration as defined by Applegate [Applegate, 1991].

IRIS Explorer is usually executed on a single resource, as such its capabilities to load large datasets and to cope with large problems is severely limited. Essentially the bigger, in complexity or data size, the problem that is to be visualised the bigger, in terms of memory and processor capability, the computer that is required. Modules in the IRIS Explorer environment can be run on a remote computer using rsync but each module is limited by the capabilities of the resource that it is run on.

The COVISA extensions to IRIS Explorer allows synchronous collaboration to take place but can duplicate the processing done by modules, by placing a copy locally to each participant. The data interchange between collaborators is not optimised for efficiency and no synchronisation of the visualisation views is provided.

The gViz project [gViz] is looking to 'grid enable' IRIS Explorer to allow the application to connect to and to steer simulations running on the Grid. The project has developed a XML web based visualisation pipeline designer, SVG Map Editor, which produces pipelines described in skML a custom markup langugage. These pipelines can then be read by an IRIS Explorer module that creates a map inside IRIS Explorer and allows modules to be launched on specified hosts. gViz is limited by the fact that it is grafting grid technology onto an existing product that has not been designed with that mode of operation in mind. Therefore the whole pipeline only exists whilst the client is running, whilst the application can attach, detach and re-attach to simulations that are left running it cannot be left processing data produced by the simulation. The gViz model also only allows a forward flow of data through the pipeline which makes advanced interaction pipelines difficult to construct.

IRIS Explorer is hampered in its ability to meet the demands of grid computing through being tied to an application perspective where all modules run as part of the main application and nothing can exist or persist outside of that main application.

SCIRun

SCIRun [Parker and Johnson, 1995] is a PSE developed at the University of Utah, Scientific Computing and Imaging Institute (SCI). SCIRun differs from other MVEs such as IRIS Explorer in that it has been developed in a non-modular manner, for users to extend the functionality of SCIRun they must compile their additions into the system source code. SCIRun is targeted at shared memory parallel machines, however it now has extensions that allow it to run in other distributed environments.

SCIRun does not contain any collaborative visualisation elements directly but can be used in a shared desktop manner through use of applications such as VNC.

Figure 2.6 shows SCIRun being used to perform a medical visualisation, one of the areas that it has been developed to provide extensive support for.

SCIRun does not support multiple display types directly, support for this must be developed separately and as with other modules compiled into the system. Scalability in SCIRun is achieved by the core software being designed for a shared memory parallel architecture enhanced by extensions to support distributed processing.

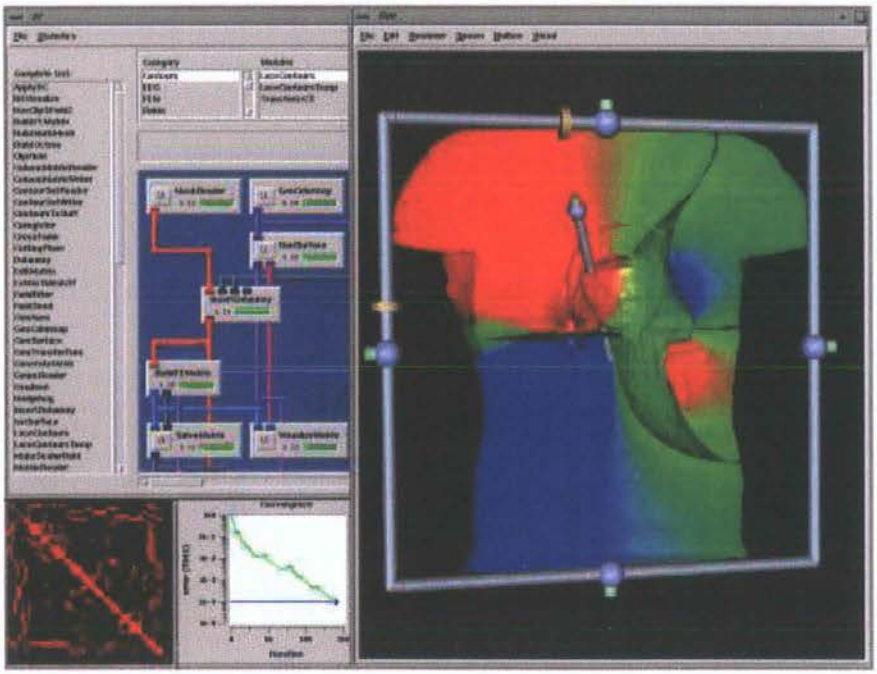


Figure 2.6: SCIRun [Parker and Johnson, 1995]

Triana

Triana [Taylor et al., 2003] is a Peer-to-Peer (P2P) problems solving environment that allows a large number of resources to be used to distributed computation across. These resources run a Triana Service which communicates with a Triana Controller Service. The services communicate using the JXTA protocol. Each Triana Service is a general purpose service and can be re-tooled to provide different functionality.

The Triana System is written in Java and uses the Java Sandbox security model to ensure that host machines are not compromised by rogue code being executed.

GAPtk

The Grid aware Applications Portal toolkit (GAPtk) [Sastry and Craig, 2003] is a toolkit that aims to provide a web-service interface onto existing legacy problem solving environments to allow them to interface with Grid middleware to allow the processing and visualisation of large amounts of data.

The toolkit communicates between services using the Simple Object Access Protocol (SOAP). The toolkit contains two implementations of the client backend for maximum flexibility, a Java based Axis implementation and a custom C library.

The toolkit offers two main types of service, one for data extraction and another for data analysis and visualisation. The data service allows querying of simple metadata from known data sources.

ParaView

The ParaView [Kitware] visualisation system is built upon the VTK library using a mix of Tcl/Tk and C++, the application grew from a joint US development project involving Kitware and Los Alamos National Laboratory and currently involves several US research organisations. ParaView like other visualisation environments allows visualisation pipelines to be built as shown in Figure 2.7. In ParaView this is done as a set of operations layered on each other, similar to the paradigm used in 3D modelling packages for graphics manipulation. The system allows users to select operations from a menu system and to alter their properties once they are added to the operation stack. There is no facility for adding external modules to the system

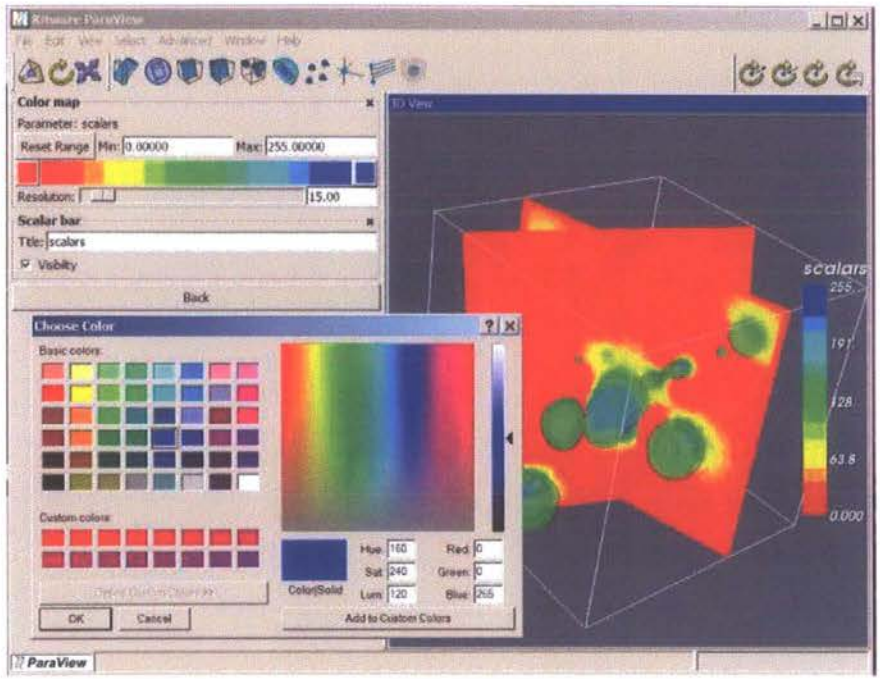


Figure 2.7: ParaView [Kitware]

and currently not all of the VTK operations are supported in ParaView. ParaView has been designed to allow scalability of the application through using MPI to allow parallel computation and parallel I/O.

The ParaView system is open source which allows other developers and researchers to develop and contribute new features to the system and to examine the source code to determine how operations are achieved.

2.4.3 Distributed Visualisation Systems

This section examines research in the area of visualisation systems with a distributed nature. Visualisation systems that have a distributed component can be classified into several subsections.

- Client/Server
- Web Based
- Applets

The area of distributed visualisation systems is growing, due in part to the demand for the use of non-traditional technologies, very large data sets and instant availability of visualisations; these are demands that cannot be served by infrastructure based on desktop computers [Brodlie et al., 2004a].

Brodlie et al. present a good overview of Distributed and Collaborative visualisation systems [Brodlie et al., 2004b], these are discussed in this section.

Distributed Visualisation Models

Several architectures and models for distributed visualisation have been developed, these are examined in this section.

MANICORAL

The MANICORAL model was developed as a reference model for distributed cooperative visualisation as part of the Multimedia And Network In COoperative Research And Learning project [Duce et al., 1998]. The model is shown in figure 2.8.

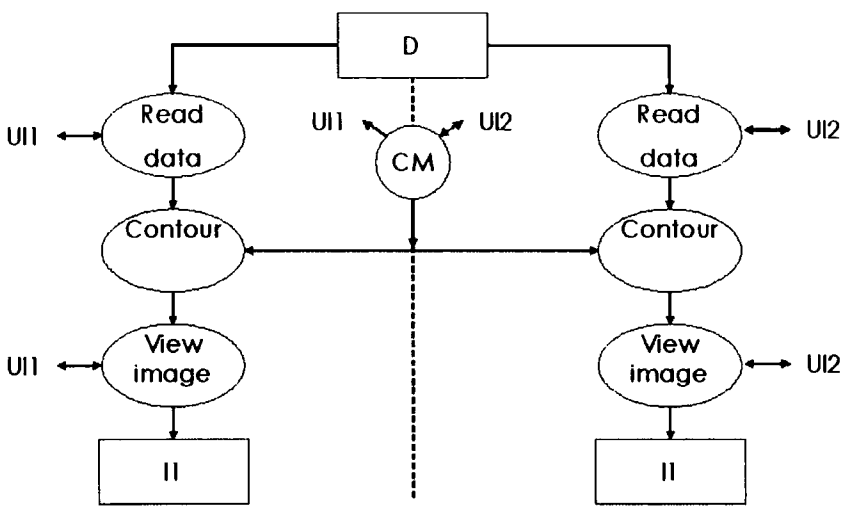


Figure 2.8: MANICORAL Reference Model [Duce et al., 1998]

The model contains input data, modules and output image. The modules perform the computation. In many visualisation pipelines such as the Haber

McNabb [Haber and McNabb, 1990] each one of these modules is specialised to perform a specific task; filter, map or render.

The MANICORAL model involves a large duplication of tasks, in Figure 2.8 the read data, contour and view image tasks are all duplicated. There is communication between the contour modules which allows both pipelines to be loosely synchronized.

A model which involves a duplication of the pipeline results in a wasted computation, where the resources running the duplicate pipelines are dissimilar it becomes difficult to maintain synchronisation between the pipelines, unless a lowest common denominator factor is used to maintain balance.

COVISA

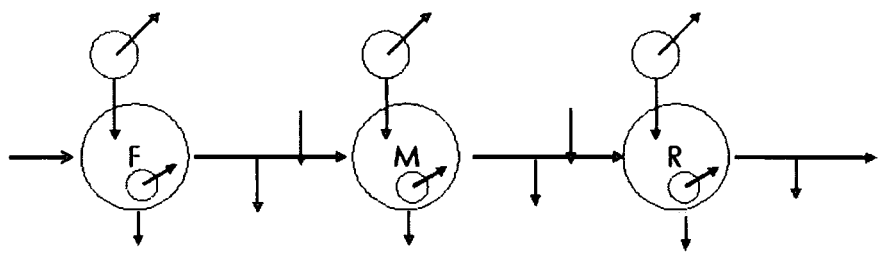


Figure 2.9: Woods Model for Distributed Cooperative Visualisation

Integrated into IRIS Explorer, COVISA is a central server collaboration system which distributes information to connected users.

The COVISA model as shown in 2.9 allows modules in the visualisation pipelines to be shared between multiple users. This is achieved by adding a remote output to a module and the remote user adding a remote input. The remote input feeds into a copy of the next module in the pipeline. In this way any changes in the host pipeline up to the point of the remote output are reflected in all pipelines. The central server approach allows users to specify if updates from their modules, below the remote output/input, are communicated to others in the collaboration. No floor control policy is implemented meaning a free for all approach to collaboration is taken.

This type of model requires a central server, or a master user, this user must

remain in the visualisation at all times making asynchronous collaboration difficult to achieve. The exporting of modules to other users across a network introduces delay in the propagation of data through the pipeline as changes must be fed to all receivers of the data.

COVISE

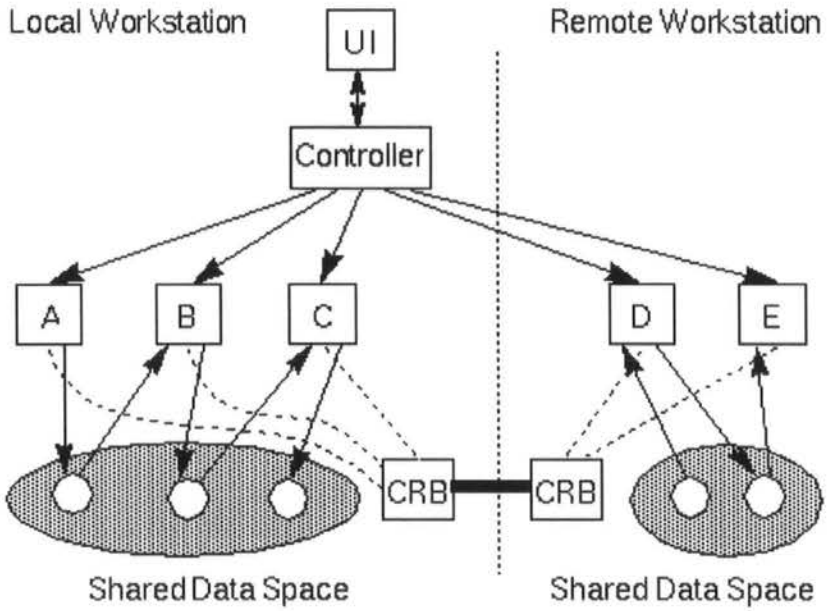


Figure 2.10: COVISE model

COVISE [Wierse, 1995] is a collaborative visualisation and simulation environment designed for supercomputer based working, allowing modules to be spread across different machines to make best use of their characteristics.

The COVISE model shown in Figure 2.10 uses a shared data space to distribute the original data to each client, the clients then process the data to produce the graphics for the visualisation. The clients are managed by a single central controller and the only data that is passed between users once the original data has been distributed is viewpoint synchronisation data.

This model is useful where a static scene is generated at the start of the visualisation process with only small positional updates to be sent after that time and the visualisation is running on comparable resources that can handle

the graphics and update load. As the bulk of the data transfer occurs at the start, this can be done using a high speed data transmission mechanism. For scenes where data is generated throughout the lifetime of the visualisation this model is less useful as the data must be replicated to multiple locations increasing the amount of data. Synchronisation of the scenes across multiple clients is an issue that needs to be carefully considered when this type of model is used.

Client/Server

This section looks at visualisation systems that split the visualisation process into two parts, one of which is a server and runs remotely and the other is a client that runs locally to the user.

The most basic Client/Server architecture is that where the raw data is stored remotely, on a web server for example, and the client loads the data for visualisation from a URL.

At the opposite end of the spectrum is the situation where the server performs all of the visualisation and just sends the final picture to the client, this type of architecture is used in the RAVE project [RAVE] which is concerned with visualisation to a PDA.

There is a trade off with this type of system, when is it best to perform graphical calculations on the display device and when is it best to perform these on a remote device and send the completed frame to the display device. Often the solution is a hybrid, one that combines the sending of some completed graphics with some on device calculations. This allows a level of local interactivity without requiring a large amount of data transfer whilst not overloading the local device with the graphics processing requested of it.

Web Based

This section looks at visualisation systems that make use of Web and Internet technologies.

A web-based front end to Amira, termed WEB-IS2 [Wang et al., 2003], has been

presented in which a web page containing a Java applet is loaded by the user and used to interact with a remotely running version of Amira, a MVE.

The Java applet provides a GUI that allows the user to establish visualisation pipelines on the remote server which performs the computation and can then display the result to the user. The applet is customised to support a variety of platforms including PDA devices with a limited display area. In this case the number of options presented to the user is reduced, leaving only the most used and most useful features.

The system is limited by the computational power of the service resource and the amount of memory available for a user session, if one user is performing a particularly memory intensive or processor intensive visualisation all users of the system would be impacted.

2.4.4 Applets

This section looks at visualisation systems that used Java Applets to provide visualisation services to users via the web.

Applet based systems such as VizWiz [Michaels and Bailey, 1997] provide a platform independent environment that can be run from a Web Browser by a user on the internet. Such systems use the processing capabilities of the local machine to perform the visualisation. A limitation with Java Applets is the sandbox security model which prevents them from accessing data from the local machine. To work around this problem Applet systems are either coupled to a limited number of datasets made available or provide a mechanism via the a Web Browser to upload a data file to the remote server which can then read back and used by the Applet. The need for the Applet to download a remote dataset can result in a large delay before visualisation can begin especially if a large dataset is used. The performance capabilities of the local machine also greatly impact the effectiveness of the visualisation system to perform the necessary computations on the dataset.

Feature/System	IRIS Explorer	SCIRun	ParaView
Graphical Composition	X	X	X
Synchronous Collab.	X		
ASynchronous Collab.			
Remote Resource Use	X		X
Scalable	X		X
Reusable Modules	X	X	
Accessible via Internet	gViz		
Modules easily ported			
Multiple Architectures	X	X	
Parallel Implementation		X	X

Table 2.1: Comparison of Features in Visualisation Systems

2.4.5 System Features

The range of features of a variety of visualisation systems support is shown in Table 2.4.5. This list is helpful in identifying the features that are lacking from current visualisation systems and provides a comparison for use in evaluating a grid visualisation solution. These features are derived from the criteria for success and the scenario outlined in the introduction.

2.4.6 Current and Future trends in Visualisation Systems

This section looks at the directions that visualisation systems are going in with the advent of Grid Computing and the developments that can be expected in the future.

Visual Supercomputing

The term visual computing was introduced by Brodlie et al. [Brodlie et al., 2004a] and is defined as:

"Visual Supercomputing is concerned with the infrastructural technology for supporting visual and interactive computing in general, and visualization in particular, in complex networked computing environments"

They set out a vision for and the challenges to developing the 'next generation' of visualisation systems, including the use of autonomic computing technologies to build self-healing visualisation pipelines and the desire for a 'visualization secretary' to automatically find visualisation services and compose them into a correct pipeline for the non-expert user.

The e-Viz project [e-Viz] has been established to begin work on developing such an infrastructure. The e-Viz project uses an XML based knowledge base to discover an appropriate visualisation process which is executed by the e-Viz server, either interactively or in batch mode, before the results are returned to the e-Viz client and the visualisation customer. The project will develop prototype software systems, conceptual models and interaction protocols which will be evaluated in a simulation environment simuVS.

The Impact of the Grid

Shalf and Bethel present a vision of the short to medium term future with their DiVA project which looks at a framework, the Distributed Visualization Architecture, for distributed, component based, Grid-enabled visualisation [Shalf and Bethel].

A list of properties of this framework is presented:

- Distributed, Heterogeneous Components
- Fundamental Graphics and Visualization Algorithms
- Distributed Execution and Dynamic Scheduling
- End-to-end Performance

The conclusion that this type of development is a "daunting challenge" is made, but also that "Such a vision serves to promote growth of visualization technology into Grid environments, and to promote unity within the visualization and computational science communities."

2.5 Summary

Visualisation, both as an art and a science, has an important role to play in the analysis of large data sets.

Numerous techniques for visualisation are available these are embodied in visualisation systems, either bespoke or general purpose, closed source or open source, stand alone or distributed. Each system has a different range of features but none provides a suitable environment to allow asynchronous and synchronous collaborative visualisation using federated resources to scale in the analysis of very large data sets.

It is this opportunity for research that must be addressed by new visualisation systems. The future impact of visualisation has the potential to be great as it is increasingly used for the analysis of scientific data.

Chapter 3

Seeing Double, the wonderful world of Stereo

The world around us is three dimensional, through having two eyes and seeing stereoscopically we can make judgments about the world more easily.

Typically a computer monitor does not afford us the luxury of allowing images to be seen stereoscopically. However recent advances have allowed the creation of displays that allow the the viewing of images in stereo.

The creation of this type of display presents an opportunity to leverage this technology to display three dimensional visualisations in stereo, providing greater insight into the data being visualised [Ware, 2000].

The use of stereoscopic displays requires special images to be generated for each display. This chapter examines the types of stereoscopic display and the techniques for generating the correct stereo images for them.

The proliferation of stereo capable devices at an attractive price point has led to it being difficult to ignore such devices and their capabilities. As takeup of devices increases so does the demand for applications to support them. The PC revolution putting a computer on every desk has altered working patterns and concepts of users. They expect to be able to work at their desk, on their computer, with their display. Therefore the demand for personal stereo capable devices is likely to be high.

3.1 Overview

Stereo graphics have increasingly been used by the scientific community as the technology to generate high quality interactive stereo has become available. This chapter examines how humans see stereo and the technology that is available works to produce the illusion of a three dimensional world on a two dimensional screen. The technology allows people to "suspend disbelief" and feel as though the things they are seeing actually do sit in front of or fall behind the screen they are viewing them on.

3.2 How we see stereo

One of the major driving factors behind the creation of stereo capable devices is human biology. It is only with an understanding of the human visual system and how it works that attempts to develop devices to fool our brains into thinking they are seeing a 3D object have been made.

Figure 3.1 shows how the eye connects into the brain to form the human visual system [Leigh]. Through having two eyes and with their forward facing, horizontally displaced position in the human skull each eye sees a slightly different view of the world. These two views fused back together by the brain are what allows the world to be seen in stereo.

Whilst the physical make up of the human body provides a large proportion of the information our brain requires for stereoscopic vision, extra information to aid the judging of depth can be found within the scene being viewed. A 2D image such as Figure 3.2 contains many cues that help determine depth. These can be used as 'standalone' features or can augment the information gained from two views of the same scene.

- Interposition
- Linear Perspective
- Light and Shade

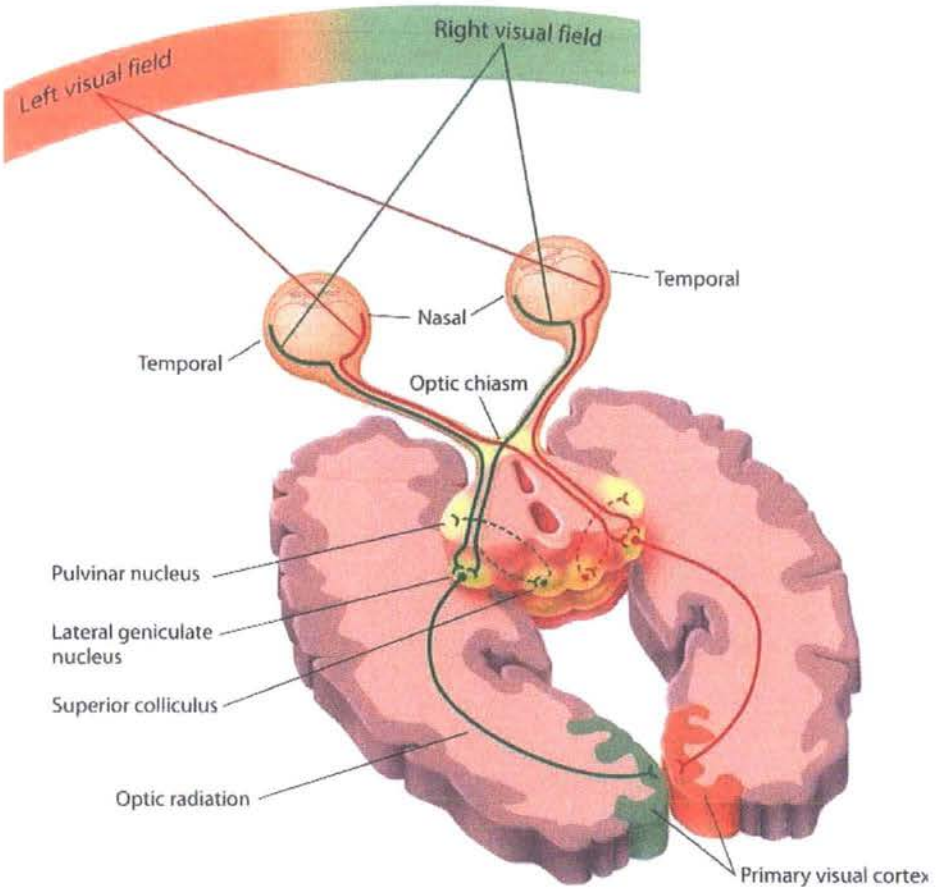


Figure 3.1: The Human Visual System [Leigh]

- Relative Size
- Texture Gradient
- Aerial Perspective

In addition to the 2D depth cues presented above two more cues exist, these are:

Motion parallax seen when the object under observation moves or the observer moves.

Oculmotor cues from the physical working of the eye as it focuses on the scene being observed.

All of the cues together with stereo vision provide an excess of information to the visual system, this excess means that the visual system can still determine depth

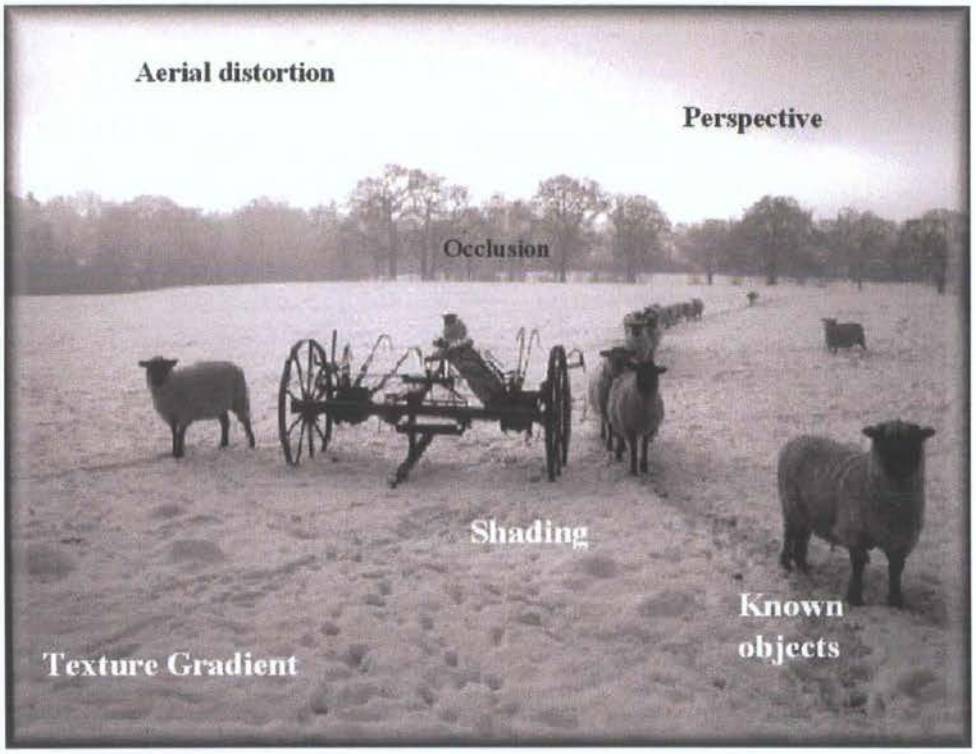


Figure 3.2: 2D Depth Cues [Holliman, 2004] (Photographer: David Burder)

without all information being present. This is demonstrated by the fact that those people who do not possess stereo vision are able to function well in a 3D world.

Binocular Vision provides several benefits over monocular vision; these are listed below:

- Relative Depth Judgment
- Spatial Location
- Breaking Camouflage
- Surface Material Perception
- Judgment of Surface Curvature

These benefits can be useful in visualisation where complex 3D phenomena are being studied and where depth judgment is of importance in observing the complex interactions between parts of the phenomena in order to understand what is going on [Ware, 2000].

3.3 The Basics of Stereo

This section looks at the area of stereo graphics covering methods for generating stereoscopic images and the factors that impact on the generation of correct stereo images.

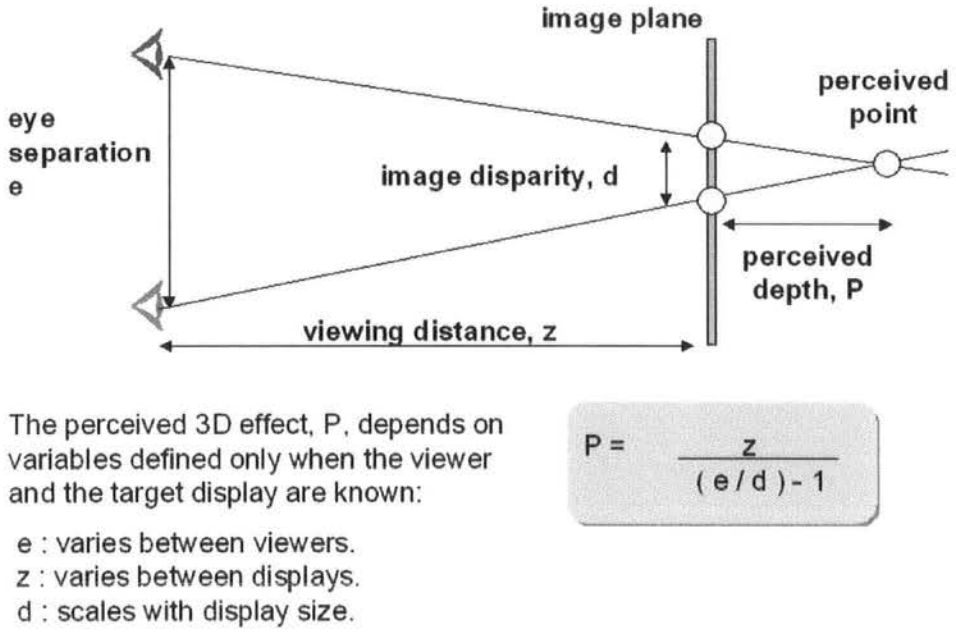


Figure 3.3: Equation governing the perception of Stereo Images [Holliman, 2004]

Stereo images are created using two images, one of these is created for the left eye and the other for the right eye. In generating these images the factors and equation shown in Figure 3.3 need to be considered.

The eye separation of the user, e, cannot with most displays be altered for each user so an approximation of the 'average population' is used when displays are designed and manufactured. For different displays the parameters Z and d are known, but differ as they depend on the physical properties of the display. By using these parameters the location that the cameras should be positioned at to create the desired image can be found.

As these factors differ for each display type, the camera positions must be tailored for each screen that is to be used in order to generate the correct image for that display.

The generation of stereo images can occur physically or virtually. With physical generation a traditional camera is used to take the two images of the (physical) subject, these are then combined digitally. Alternatively two virtual cameras can be used to generate stereo images of virtual scenes using rendering software.

The generation of correct stereo images is dependent on the device being used to display those images and the viewer of the images themselves. Figure 3.3 shows the controlling equation for stereo image generation. If the values for the equation are unknown then the quality and correctness of the stereo images is impacted on for the particular screen. It is not normally possible for a viewer's eye separation to be known when generating stereo images, as such a viewer with an eye separation outside of the 'average population' may have difficulty viewing the generated images. Therefore the display device that will be used to display the generated image must be known before the image generation which reduces the portability of generated images across displays.

3.4 Stereoscopic Displays

There are a wide range of stereoscopic displays available, these can be split into two broad categories, those requiring the user to wear eyeglasses and those that do not. For an indepth examination of display types readers are directed to Holliman [Holliman, 2004].

In the category of display requiring users to wear eyeglasses we look at shutter glass displays and polarised glass displays, also known as active and passive stereo displays.

3.4.1 Eyeglass Displays

This section looks at two types of stereoscopic display system requiring eyeglasses, those that require shutter glasses, often called active stereo displays and those that require polarising glasses, often called passive stereo displays.

Shutter Glass Displays

Shutter Glass displays, often known as active stereo displays, make use of frame sequential stereo, where a left image is displayed in one frame and the right image is displayed in the next frame. To correctly display these images to the left and right eye of the user they wear special glasses, shutter glasses, which contain filters that darken alternately synchronised with the display so that each frame is shown to the correct eye. These glasses tend to be heavy and cumbersome which affects the users experience of using and interacting with the display.

Polarised Glass Displays

Polarised Glass displays, often known as passive stereo displays, project the left and right stereo images at the same time. The two images are polarised differently, for the user to see the correct image in each eye they must wear glasses that only allow the correctly polarised image to reach each eye. These glasses are lightweight and less cumbersome than shutter glasses.

A project polarised type of display is often used for large screen display and large group collaborations.

3.4.2 Autostereoscopic Displays

Autostereoscopic displays are stereoscopic displays that do not require the user to wear any special eye or headwear in order to see stereoscopic images. Three types of autostereoscopic display are examined, a standard, two view, fixed viewing position display, a multiview display and a tracked autostereoscopic display.

Autostereoscopic displays are typically manufactured from Liquid Crystal Displays (LCD), a standard LCD panel is used with an additional parallax [Holliman, 2004] barrier placed in front or behind the panel. The parallax barrier can either be fixed (always on) or switchable depending on the design chosen. A switchable barrier allows the LCD to be used as a traditional 2D monitor in addition to an autostereoscopic display but results in a slight reduction in stereo image quality due to an increase in cross talk between channels.

Two view Display

This autostereoscopic display is more correctly described as a two view, fixed viewing position display. In this type of display a static parallax barrier is used, and a single stereo pair is shown on the display. The display has a fixed viewing position.

An autostereoscopic display has been incorporated into a laptop computer by Sharp using a switchable screen, it also features in a mobile and is available in a desktop model. This type of display is typically used by a single user.

Multi View Displays

A multi view display has a wider viewing angle and multiple viewing positions when compared with a standard two view display. Where a standard display uses a single stereo pair to display the stereoscopic image a multi view displays use a number of stereo pairs. This means that viewers of the display can see a stereo pair from whichever angle they view the display.

This type of display is often used in a collaborative manner with multiple users gathered around the display.

Head Tracked Displays

Head tracked displays are a specialised version of autostereoscopic displays, they are often variations of standard two view displays, but instead of having a fixed viewing position have optics that allow a user's view to be tracked.

This type of display has a mechanism which can detect the location of the user's head and/or eyes and alter the properties of the parallax barrier so that the user can see a correct stereo image from their location.

This type of display allows the viewer a greater freedom of movement whilst using the display.

3.5 The science behind the magic

There are a wide range of stereo devices available in the market place each with different features and target markets.

The technology behind the displays, allowing them to display stereo images varies from device to device but the main principals of the technology fall into several main categories.

- Wheatstone mirror stereoscopes
- Polarised glasses with polarising display
- Shutter Glasses with a view switching display
- Analglyph glasses (red/blue stereo is an example)
- Brewster Stereoscopes (Head Mounted Displays for example)
- Parallax Barriers
- Lenticular Optics
- Micropolorisers
- Holographic Elements

Some of these technologies require the user to wear glasses or a head mounted unit to see stereo, others allow the user to see stereoscopically without the need for any special technology beyond the screen. It is this type of display, autostereoscopic displays, that we consider in detail below.

3.5.1 Parallax Barriers

Displays based on parallax barriers are constructed as shown in Figure 3.4. A layer containing strips of a light blocking material is made into a sandwich with a TFT display and a light source.

The striped layer can be made of an electronically switchable material and therefore allows a display to be a switchable between 2D and 3D. This type of display is restricted to a usually fixed two view display.

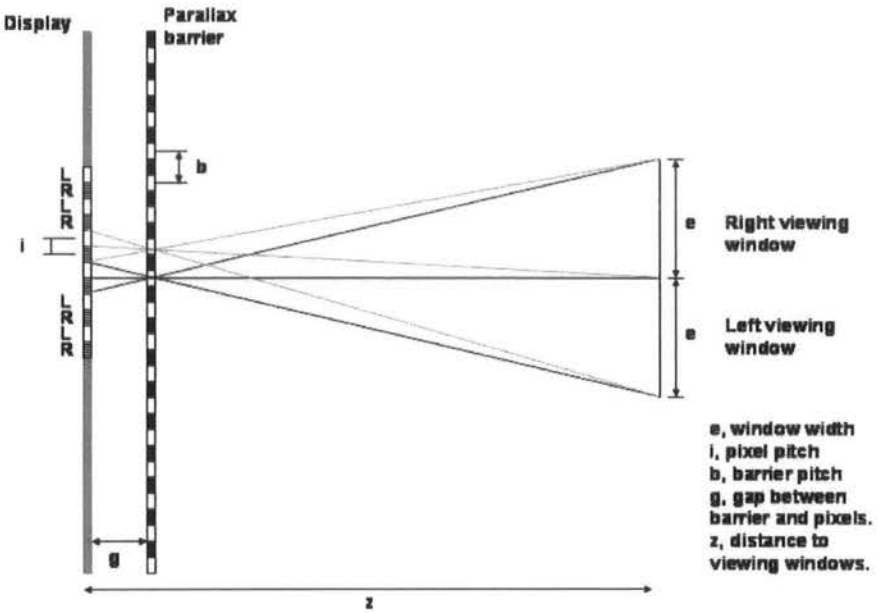


Figure 3.4: Parallax Barriers [Holliman, 2004]

3.5.2 Lenticular Optics

Initially a non-switchable technology, lenticular based displays as shown in Figure 3.5 made use of a series of cylindrical lenses across the surface of the display which are aligned to direct the light correctly to each eye. Recent developments by Ocuity [Ocuity, 2005] and Philips [Royal Phillips Electronics, 2005] have produced switchable lenticular displays which can operate at full brightness compared with parallax barrier displays.

Some lenticular displays have been motorised to work with eye tracking systems allowing a greater freedom of movement by the viewer.

3.5.3 Micropolarisers

A micropolariser autostereoscopic uses polarised light from an LCD and a patterned retarder which acts like a parallax barrier. In front of these as shown in Figure 3.6 is an analysising polariser, if this front polariser is removable the display can be a mechanically switchable 2D/3D display.

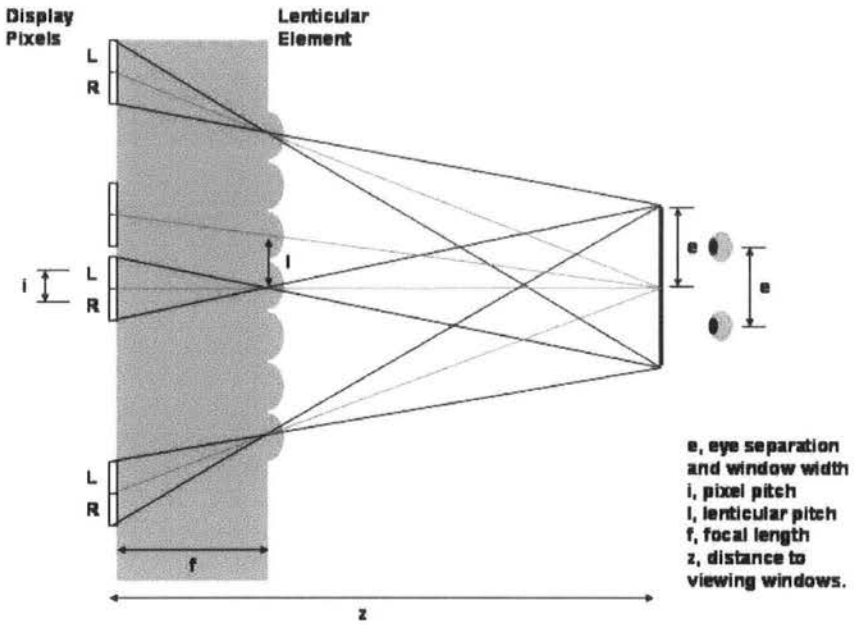


Figure 3.5: Lenticular Optics [Holliman, 2004]

3.5.4 Holographic Elements

Holographic Optical Elements have been used in the creation of 3D displays [Trayner and Orr, 1996]. The optical elements are arranged in horizontal strips so that alternate strips form left and right viewing windows.

A few practical problems remain with the approach, one of the most significant is colour fringing due to the diffractive nature of the elements. The system can track users by moving the light source and also made switchable between 2D and 3D with a modified light source.

3.6 Software Support for Stereo Devices

Given the proliferation of stereo devices the quantity of software that can drive them is increasing, particularly software that will support them "out of the box". The special considerations required when generating stereo images have already been discussed. These considerations have generally been put to one side by software

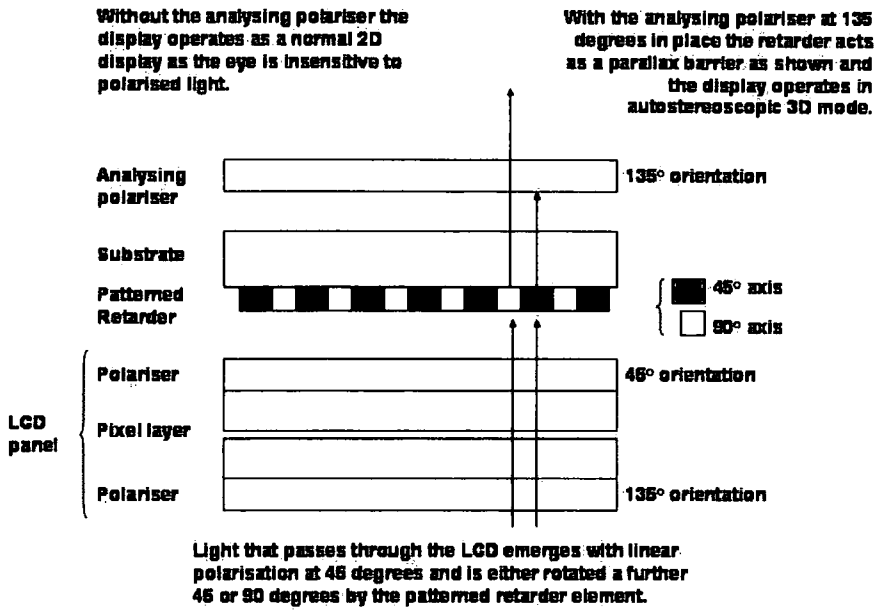


Figure 3.6: Micropolarisers [Holliman, 2004]

developers who have used a "one size fits all" approach to stereo enabling their applications.

This section looks at the stereo support offered by some visualisation packages and the ability or otherwise to include or alter stereo algorithms.

3.6.1 The Visualisation Toolkit

The Visualisation Toolkit (VTK) has been developed as a library of visualisation components that can can be joined together programmatically to form a visualisation pipeline, for more detailed information refer back to Chapter Two.

The display component of the toolkit contains several stereo modes including, red/blue, frame sequential and Interlaced. However the stereo is generated from a fixed camera position within the scene. When the scene is interacted with, particularly when zooming into a region of interest occurs the disparity shown is greater than the display or the human visual system can accommodate. As such at best artefacts within the image appear and at worst the stereo effect is lost completely.

For alternative stereo generation algorithms to be incorporated into the VTK viewer this would have to be done as a rewrite of or a modification to the viewer module. The source code of the software is available to allow this and the suppliers of VTK, Kitware accept contributions to the development of the package. It is perhaps a cumbersome and time consuming way of introducing updates. The system is also tightly coupled so any package that wanted to make use of the system would have to be recompiled with it. If it was not incorporated into the main body of the VTK library distribution the task of maintaining and distributing an alternative version of the viewer with each new release of VTK would have to be managed.

3.6.2 IRIS Explorer

IRIS Explorer is a modular visualisation environment, described in detail in chapter two, it has a library of modules which are joined together in a visualisation map to form a complete visualisation. The output of the visualisation is displayed within a viewer module which is contained within the visualisation environment.

The module has limited stereo support modes but as it is contained within the visualisation environment the whole of the visualisation map must be shown on the stereo screen, not just the viewer module. This makes interacting with the map difficult and the amount of non-stereo content being displayed on the stereo screen makes for an uncomfortable experience.

Much of the criticism levelled at VTK about the nature of the stereo also applies to IRIS Explorer. With IRIS Explorer standalone modules that can be added to the library can be developed this would allow a suitable viewer for each stereo screen to be developed and for some control over the way the visual content was produced. This would not negate the fact that the produced viewer would have to exist inside the environment, however it may be possible to link some external viewer to the system to counter this problem.

3.6.3 VisAD

The VisAD toolkit described previously is developed in Java and as such makes use of the Java3D libraries for three dimensional visualisation and stereoscopic visualisation. The toolkit therefore is limited by the support for different stereoscopic displays to that contained within Java3D. This support is generic and cannot be easily tailored to different stereo displays. It would be possible using Java3D to develop a custom stereo display that could be configured for individual displays but to work effectively it would require a user editable configuration file to allow the parameters of each screen to be set.

3.7 Summary

The technological advances allowing the creation of stereoscopic displays have opened new doors for three dimensional visualisations to be displayed. Displaying these visualisations stereoscopically can provide a good insight into the data being visualised.

To work effectively with autostereoscopic displays visualisations need to be designed so that images for the display are correctly generated. This need has an impact on the software used to generate the visualisations. This chapter has examined a range of stereoscopic display devices and the techniques to generate the correct images for the displays so as to better inform the development of the visualisation architecture.

Chapter 4

The Grid

Grid Computing, an emerging area of research, the next generation computing technology, breaks down barriers between institutions allowing resources anywhere on the network to be utilised. This is the vision, a vision of the right computer for the job, computing power on demand allowing even the most intensive of computations to be performed with ease. Interoperability of billing, security and management systems reducing the burden of administration. Currently two implementation paradigms are being used in an attempt to make that vision a reality. This chapter discusses the vision in depth and examines the implementations underway to realise the vision.

4.1 The Grid Vision

Foster and Kesselman [Foster and Kesselman, 1998] are two leading grid visionaries, building on the area of meta-computing they presented the grid as a utility, such as the electricity distribution network, providing computing power and resources that can be used on demand. This vision has since been refined to mean, multiple institutions collaborating together in a virtual organisation, federating heterogeneous resources using a meta-computing infrastructure in order to achieve a goal.

A range of computing applications are immediately suited to the grid. They are those that certain decomposition techniques can be applied to such as:

- Pipeline Decomposition.

- Functional Decomposition.
- Data Parallel Decomposition.

Other applications may need to be refactored to take account of the implications of grid computing.

4.2 The Batch Processing Grid

The High Performance Computing (HPC) community were the first adopters and drivers of the grid computing vision, consequently the batch processing grid was the first attempt to build a common infrastructure, or middleware, to allow users at differing institutions make use of resources in a standardised way. These users were used to submitting jobs and waiting for hours, days, weeks or even months for their job to run and return a result. To these users the idea of a shared job queue so that their jobs could run on a machine anywhere in the world potentially, making use of any available suitable resource was a huge leap forward. It would mean that a job that might sit in a queue for weeks might get serviced in a matter of hours or days. Worldwide computing load would be balanced, overworked local resources would have their loads reduced and spare computing capacity remotely would be soaked up. This vision of the "metacomputer" was presented by Smarr and Catlett in 1992 [Smarr and Catlett, 1992] having a five year horizon to realisation, over ten years later despite the progress made the dream is still that. Whilst this vision was utopia for traditional HPC users the reality was somewhat different, the importance of human factors in establishing trust and acceptance was forgotten. Institutions are reluctant to allow external users on their resources, and users are very good at crafting programs suited to their institution's HPC resource and not suitable for running in a different enviroment. The issues of scheduling across institutional boundaries, of security, of predicting performance and of specifying required resources are all complex and challenging ones, some have been resolved, some have been partially resolved and many remain as open problems.

The Globus Toolkit 2 [Alliance, 2004], was developed to allow this type of computing to occur. The toolkit was developed by the Globus Project (now Globus Al-

liance) as a standard set of tools to provide access to heterogeneous resources. The toolkit included allocation managers, high performance data transfer (GridFTP) and monitoring and discovery services.

This toolkit has been widely deployed in high performance computing centres around the world and has been used in many projects to provide a mechanism to distribute computation. However this has usually been within an existing HPC environment or using a national resource that has been tightly controlled and that has a common software set across the environment.

4.3 The Service Oriented Grid

The proliferation of the the grid vision throughout first the computer science community and then the scientific community as a whole brought a demand and a desire to perform computing that was not done in a batch processing manner and a need for more user friendly tools. The grid community in response made the move to a service oriented architecture as the next iteration in the development of grid software allowing resources to be interacted with through a mechanism based on current web technologies.

The first attempt to produce a service oriented grid middleware was Globus Toolkit 3, which introduced Grid Services, a proprietary extension of Web Services, that introduced the notion of state and wrapped the tools in Globus Toolkit 2 with a new interface.

This first attempt is being superceded with a non-proprietary implementation that retains compatibility with Web Services, and contributed to the Web Service development process, this implementation will be released as Globus Toolkit 4.

Due to the flux that has plagued the Service Oriented Grid implementations the majority of current deployments are restricted to testbed grid infrastructure being used for specific projects. No national infrastructure has yet been deployed which may hamper attempts to migrate from the batch oriented grid software with projects preferring to remain with that which they know and which has infrastructure in place.

4.3.1 Grid Services

Grid services in the Globus Toolkit 3 were developed by the Globus Alliance as an implementation of the Open Grid Service Infrastructure (OGSI) version 1 [Banks et al., 2004], part of the the Open Grid Service Architecture (OGSA) framework [Foster et al., 2004] developed through community processes at the Global Grid Forum (GGF).

Grid services in Globus Toolkit 3 built upon the Globus Toolkit 2, a toolkit designed for batch processing grids. The toolkit wrapped the functionality exposed by version 2 of the Globus Toolkit as services and allowed these to be exposed on the grid.

These grid services are stateful in nature, to make use of a service, an instance has to be created by using a service factory. The created service is then used to perform the operations that are made available. Grid services are stateful as an instance is created for each user that wishes to use the service, each instance of the service can store data, known as Grid Service Instance Data, about the state of the service, the user and transactions being performed. Grid services have a lifetime associated with them, this lifetime may be infinite or finite. This means that grid services can be transient and stateful.

4.3.2 Web Service

As adoption of grid services began to increase the web service community examined the solution and those making use of version 3 of the toolkit provided feedback. They found the proprietary extensions hard to use and the web service community objected to the way that state had been introduced to services. This led to a re-examination of the service oriented architecture for the grid.

Whilst new proposals were being formulated it became apparent that whatever solution was presented, it would remain compatibility with web services and as such we examine the area of web services below.

Web services are defined [Gudgin and Ewald] as as an endpoint for communication that:

- Uses standard high-level Internet communication protocols like HTTP or SMTP
- Transfers data using XML messages
- Describes its message types using a portable type system which is both language and platform neutral
- Provides a way to access metadata describing the messages it accepts

Web services are a stateless technology, in that they are designed to have one persistent instance per resource and that the same operations are called by all clients of the service. The web service must therefore be stateless otherwise operations called by one user may affect the computation being performed by other users. Web services are persistent and stateless, this is a fundamental difference to grid services which are transient and stateful.

To address the move towards the use of web services and the fact that web service technology itself is still maturing, a white paper defining the WS-I+ profile that establishes a common set of technologies that are safe to use to try and ensure a level of compatability across web service based implementations has been published [Atkinson et al., 2004]. This is seen as an intermediate stage until the service oriented architecture for the grid is re-established in a new form.

4.3.3 The Future of Services on the Grid

Grid services are being re factored by several organisations including the Globus Alliance to build them with standard web service technologies, allowing them to interoperate with web services, and other services deployed on the grid.

The Globus Alliance offering is based on the Web Service Resource Framework (WSRF) [Czajkowski et al., 2004] and will be called Globus Toolkit 4.

The UK Open Middleware Infrastructure Institute (OMII) is also producing an implementation that will be WS-I+ compliant and will be called OMII_1 (omi-one)

4.4 Globus Revisited

There are multiple incarnations of the Globus toolkit, a short review of these versions is conducted below.

- Globus Toolkit 2

  The first major release of the toolkit, designed for use in a batch processing and HPC environment. It lincludes tools for data transport, discovery, monitoring and resource allocation.

- Globus Toolkit 3

  A complete revision of the toolkit based on the OGSI specification, introducing grid services and providing service wrappers of the version 2 tools.

- Globus Toolkit 4

  A revised implementation of version 3 inline with the WSRF specification. The development of version 4 is still underway and no final public release has been made to date.

4.5 The Grid now and to come

The vision of grid computing is a utopia for computer scientists, limitless CPU, storage and other resources accessed from anywhere in the world, via high speed network interconnects, through a compatible set of tools, protocols and services. The reality is somewhat harsher, human factors, immature technology and perhaps a desire to run before walking means that grid computing is currently floundering, trying to gain a stable foothold with which to lay down some firm foundations. Despite this the vision still burns brightly, drawing in researchers like moths to a flame, stirring passions and stimulating academic discourse and debate. The current implementation arena is strewn with dead, dying and mortally wounded challengers, the old guard fights its corner whilst fresh young challengers enter to do battle, each generation learning from the mistakes of the previous one. A successor to the batch computing guard may come along but it is more likely that arms will be laid down

and a spirit of mutual cooperation will prevail and that a symbiotic relationship will emerge.

4.6 Beyond the Grid

The term 'Interactive Supercomputing' is used to describe an ideal for high performance interactive visualisation. It is related to the concept of 'on-demand computing' where capacity is available as and when users require it on a computer. Interactive Supercomputing would mean that a visualisation pipeline could execute in an interactive manner, without advance reservation of computing cycles, without estimating the required computing power nor the wall clock time required. It is felt that this use of computing resources would follow the CPU graph of a typical desktop computer, large idle periods with peaks of high usage. To achieve this vision of interactive supercomputing, it is felt that the mindset of supercomputer operating system designers has to change from one of batch computing to one of mixed mode operation. Interactive applications would run in a state where they are idle, when they are required to perform computation they would signal the operating system which would stop any more batch jobs being executed, and swap out any existing batch jobs, to free up the processing power required to complete the interactive job. Batch jobs can then be restarted and the system can return to a traditional batch operating system mode. This concept of operating is similar to the cycle scavenging operated by applications such as climateprediction.net [climateprediction.net] where an application runs in the background until the user requires interactive performance.

This need for Interactive Supercomputing is highlighted by several current and future users of visualisation in the NERSC Visualization Greenbook Report [Hamann et al., 2003].

4.7 Summary

The Grid provides a suitable platform for building high performance, scalable, evolvable and secure software solutions. To do this the following features are required from the grid middleware

- Bulk Data Transfer Mechanism
- Streaming Data Transfer Mechanism
- Support for Service Oriented Software
- Directory Services
- Resource Management
- Accounting Infrastructure
- Security Infrastructure

The provision of these features means that application developers can concentrate on the application logic that makes their application unique, avoiding having to re-implement house keeping features. This development model promotes standardization and interoperability between grid systems allowing applications and services to be deployed widely.

Whilst the concept of Interactive Supercomputing is presented as a future development from Grid Computing the present development stage of Grid Computing makes it the only current candidate for building high performance scalable systems.

Chapter 5

Virtualised Visualisation Architecture

The research challenge that the virtualised visualisation architecture addresses can be stated:

"How do you design the ideal service oriented architecture for visualisation that meets the needs of scientists?"

Working from the scenario described in chapter one coupled with the analysis in chapters two, three and four of current visualisation systems and their deficiencies also the technologies of stereo capable devices and the grid this chapter presents a definition of a virtualised visualisation architecture that takes these factors and considerations into account in order to answer the research question posed.

5.1 Definition

This section outlines the requirements for the design of the virtualised visualisation architecture and the architecture generated from those requirements.

A virtualised visualisation architecture in its design is driven by the requirements for that architecture. These requirements are generated from the uses that the architecture would be put to and the deficiencies found in current models for visualisation systems and their implementations.

5.1.1 Requirements

The uses that the architecture would be put to are listed below, these uses are derived from analysis of the scenario outlined in chapter one and the way in which visualisation systems are currently used. The uses of such an architecture include:

I Analysis of Large Data Sets, which may not be local to the scientists.

II Interactive visualisation.

III Collaborative Visualisation

IV Visualisation of large data sets on computers with insufficient processing power to manage them.

V Visualisations distributed across multiple institutions

VI Visualisation at the desk

VII Visualisations displayed using multiple display technologies

The requirements for the virtualised visualisation architecture will drive its design, these requirements are built from the uses that the architecture will be put to and the issues discussed in Chapter Two with regard to current visualisation systems and the specific requirements of technologies such as stereoscopic displays as discussed in Chapter Three.

1. That the virtualised visualisation architecture can run on a variety of machine architectures.

   Grids are often heterogeneous in nature, this is particularly the case when resources from different institutions are used as such any architecture should be capable of running on multiple platforms.

2. That the visualisation can be displayed on a variety of different display platforms, including stereoscopic devices.

   The range of devices available to scientists to display their visualisations on is increasing. The devices include stereoscopically capable devices such as

autostereoscopic displays. For a visualisation architecture to be of use to the scientists who make use of such devices it must support them.

3. That the visualisation should be available on the desktop.

   Many scientists work at a computer in their office, whilst they may perform experiments in a lab or undertake fieldwork away from their institution the analysis of their data is primarily performed at their workstation. For a visualisation architecture to gain acceptance it must integrate into a scientist's normal working routine.

4. That the architecture will operate across multiple institutional boundaries, accounting for security issues.

   In an increasingly connected world, institutional boundaries have been hardened against the proliferation of worms, viruses and other undesirable and unauthorised network traffic. Firewalls with a limited range of open network ports sit at network boundaries, grid resources are located behind these firewalls. It is important therefore that to reduce the work in deploying services and to reduce the security risks associated with network communication that the virtualised visualisation architecture operates using standard network protocols.

5. That the architecture will allow multiple scientists to collaborate from different locations, either synchronously or asynchronously.

   Science is often performed by distributed multi disciplinary teams; these teams may be spread across time zones which would make regular synchronous collaboration difficult. As such the virtualised visualisation architecture should support both synchronous and asynchronous collaboration.

6. The architecture should cope with very large remote data.

   One of the key driving factors behind the concept of grid computing is highly distributed working. In a team engaged in scientific endeavour this can mean that one part of a team is generating data and another is performing the analysis. For this to be achieved scientists need to access non-local data sets,

these could be large, tens of terabytes or more, and in some cases composed of multiple smaller data sets distributed across different locations.

7. The architecture should allow visualisation to be performed interactively.

   One of the powers of visualisation is the ability to ask "What If?". To perform 'what iffing' scientists need to be able to interact with their visualisations. Another form of interactivity that is required is the ability to receive updates from a changing data source, simulations or sensor, and reflect that in the visualisation. This would allow simulations monitoring tasks to be performed.

5.2 Architecture

The architecture is based on a service oriented architecture where services provide the functionality. This type of architecture is eminently suited to a distributed grid based system that fulfills the requirements enumerated above.

5.2.1 Overview

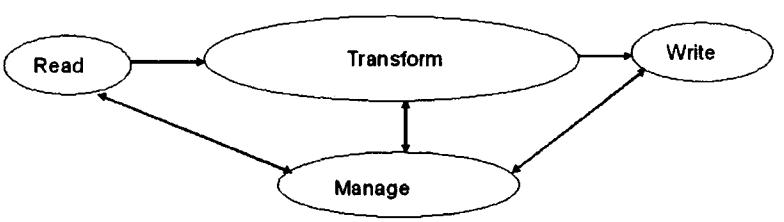


Figure 5.1: Initial Design

Figure 5.1 shows a traditional three stage visualisation pipeline, an input (read), an operation (transform), and an output (write). These stages are connected by streams, this is the starting point for the definition of the virtualised visualisation architecture.

The new architecture is shown in Figure 5.2. The links between each service are changed so that they operate across a grid.

The grid is added between these services to signify that:

Figure 5.2: Introduction of the Grid

- Different services can be located on different resources, within different institutions and at different geographic locations.
- It also signifies that services can be connected using grid technologies.
- The use of grid technologies for connections aids in achieving interoperability with other services.

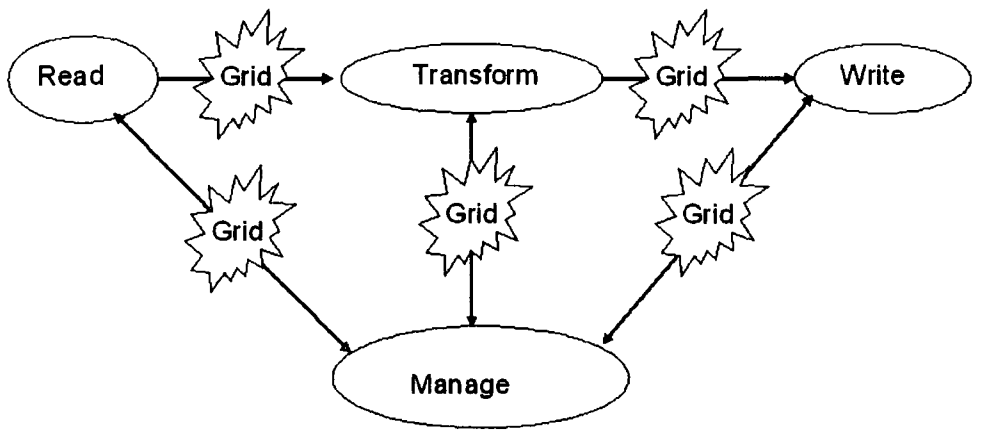


Figure 5.3: Introduction of the Manage Service

Introduced into the design as shown in Figure 5.3 is the manage service. The manage service has multiple roles in managing the visualisation pipeline.

A feature which has been incorporated into the definition to assist with interactivity within the visualisation is the bi-directional links between the manage service and each individual service. These links allow a service to make a request for more data or a change to a parameter. The request is sent to the manage service which ensures that it is completed by re-directing it to the correct service. The protocol used to change parameters and send control statements is the steering protocol.

Each service in the pipeline produces an output, this output must be directed to the input of other services to complete the pipeline. These outputs are known as data streams, they are an active data transport mechanism in that they allow the one service to be writing data at the same time as the next service is reading the data. The data streams that connect each pipeline are such that as a service has finished processing a data item it can be sent to the next service in the pipeline. In this manner very large data sets can be processed in less time as each service in the pipeline is operating in parallel rather than in a batch manner where each service in the pipeline has to finish its entire computation before the next service can process the output data.

5.2.2 Services

Four types of service in the architecture have been shown so far. Read services, which have the responsibility of making data accessible to the pipeline. These could read from a flat file, a database, a simulation or even a sensor. Transform services come next in the pipeline and perform an operation on the data, typically converting it to one form or another. Write services are at the end of the pipeline and output the visualisation in some form, often to screen but equally validly to file or other storage mechanism. The Manage service exists to manage the pipeline.

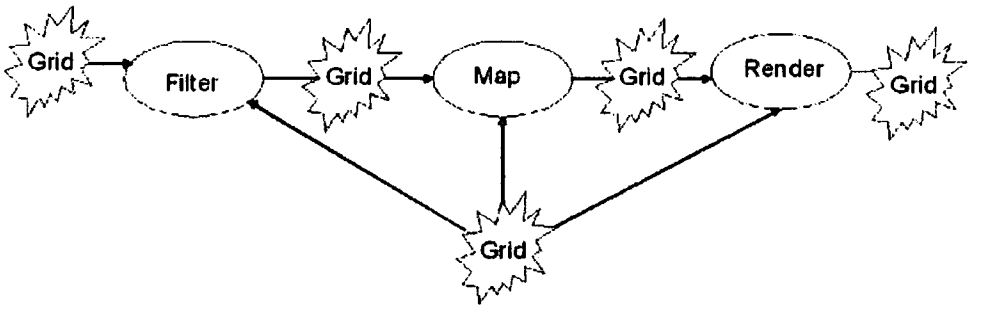


Figure 5.4: Expanding Transform Services

The transform service can be divided into three sub types, this is shown in Figure 5.4 where filter, map and render services are represented.

The resource requirements of each of the three transform stages, can differ and

the execution of them as individual units on different resources means that those requirements can be more readily met. As an example a map service may require a large amount of CPU and memory to perform a conversion to a visual representation whilst a render service may require a high performance graphics sub system.

The sub division of the transform stage also brings the added benefit of increasing the resilience of the pipeline, for example, if the transform stage was executed on a single resource and that resource failed all the computational effort of that stage would be lost. In the revised transform stage with multiple resources the effect of a single failure is reduced as is the chance of a complete failure. The complete pipeline with the three transform services, filter, map and render in operation is shown in Figure 5.5.

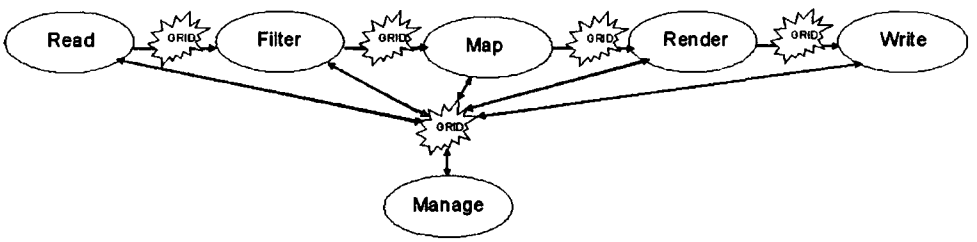


Figure 5.5: Final Pipeline Definition

Each service produces a single output stream that can be directed into another service (except the Write service which does not produce an output stream)

A pipeline may consist of multiple branches, these branches may stem from one or more reader services but will at some point in the pipeline join together. Each service therefore can accept multiple input streams from other services (except the Read Service which does not accept any input streams).

A stream is a transport mechanism between services, it is an abstract concept representing a typed flow of data from one service to another. A single stream may have multiple destinations but will always carry the same information to each.

The generic properties that each service exposes are:

- An output stream (except Write Service)
- Multiple Input Streams (except Read Service)

- Ability to retrieve configurable parameters of a service and their parameters
- Ability to change configurable parameters of the service

The services within the architecture are read, transform, write and manage, together with a set of tools for composition and the user interface these are now discussed in more detail.

5.2.3 Read Service

Read services are the services which provide the input to visualisation pipelines, they do not take input from other services, but they provide an output stream of raw data. The raw data can be held in any form, flat file, database, simulation or even a sensor. The reader service converts the format into an output stream that can be used by other services.

5.2.4 Transform Service

Transform services perform an operation on their input data and produce an output stream, there are three types of transform service:

Filter Service

Filter services are used to separate data from a data set, it may be that a data set contains a large number of variables of which only a few are required for the visualisation. The filter service would separate the required variables from the data set reducing the amount of data that required transmitting to the next service.

Filter services would also be used for performing calculations on data, for example, if a visualisation required data derived from variables in the data set a filter service could be used to perform those calculations.

Map Service

Map services are used to create a visual representation of the data being visualised. In many cases this visual representation is a geometric representation of the visual-

isation but it may be a different representation. The map service will generally use the output of one or more filter services as input.

Render Service

Render Services are used to convert a visual representation to a displayable image. The render service is typically configured to the parameters of the output device the image is destined for. For stereoscopic devices the render service may render two images itself or may use other services to reduce the load.

The scenario outlining the need for a distributed visualisation architecture in Chapter One and the study of stereo and auto-stereoscopic displays in Chapter Three highlighted the need to design the render service into a service that was easily configurable and interchangeable to support a wide variety of devices each of which would have their own unique configuration. A render service may need to have the ability to take parameters from a device such as an auto stereoscopic display which describe how the service should produce its output. An example of such a parameter set is shown in Figure 5.6:

```
e = eye separation (float)
d = image disparity (float)
z= viewing distance (float)
s = stereo type (side by side | interlaced | frame sequential )
v = number of views
```

Figure 5.6: Example configuration parameters for a stereo render service

Render services may also convert data to a non-visual output format such as sound or for a haptic interface.

5.2.5 Write Service

Write services are the terminus of the visualisation pipeline, typically they are a service which presents the image delivered from the render service onto a display device. However they may be a service that writes to a file or a service that creates a video clip of the visualisation.

A Write service may also connect to an alternative output device such as a haptic interface.

5.2.6 Manage Service

The manage service acts as the intermediary between user tools and the visualisation pipeline and also between services in the pipeline and other services in the pipeline. The manage service allows users to create a visualisation pipeline through it and can perform housekeeping operations to do with state management and security delegation.

The roles the manage service has are:

- Pipeline Co-ordination
- Context Management
- Steering
- Collaboration
- Security Delegation

Pipeline Co-ordination

The manage service is responsible for instantiation of pipelines submitted via the composition tool. Once a pipeline has been instantiated the management service directs requests from services in the pipeline to the correct destination.

Context Management

Where stateful services are in use, identifiers for a session or user may be required, the manage service is responsible for maintaining a record of identifiers and the services to which they belong. The manage service can then present a single identifier for a pipeline to users.

Steering

The manage service is responsible for directing steering requests from other services, the pipeline and users of the pipeline to the correct destination. In managing steering requests in this way individual services do not need to be aware of all the other services in the pipeline. This allows greater flexibility to reconfigure pipelines.

Collaboration

The manage service allows pipelines to be used in a collaborative manner. By managing steering requests the manage service can provide mechanisms to allow or prevent users from steering a pipeline. The manage service allows asynchronous collaboration by allowing a pipeline to persist even when no users are connected to it. This allows one user to establish the pipeline and for other users to make use of the pipeline at a later time.

Security Delegation

Using services distributed across the grid and therefore across multiple institution boundaries means that steering and accounting policies may be in effect to use each service. The use of a manage service allows a user to delegate the credentials to that manage service as a proxy; this allows all administration to be done centrally and transparently. The manage service can also implement an access control policy for collaborative pipelines , this can be used to restrict access to the pipeline.

Manage Service Design

The manage service is designed to hide much of the complexity of managing visualisation pipelines and as such it provides a lightweight interface to other services. Much of the complex management is done in the internal logic of the service. The class diagram in Figure 5.7 shows an exemplar of the interface and some of the internal data structures that a manage service would provide for use by a composition tools a user and other services in the visualisation pipeline.

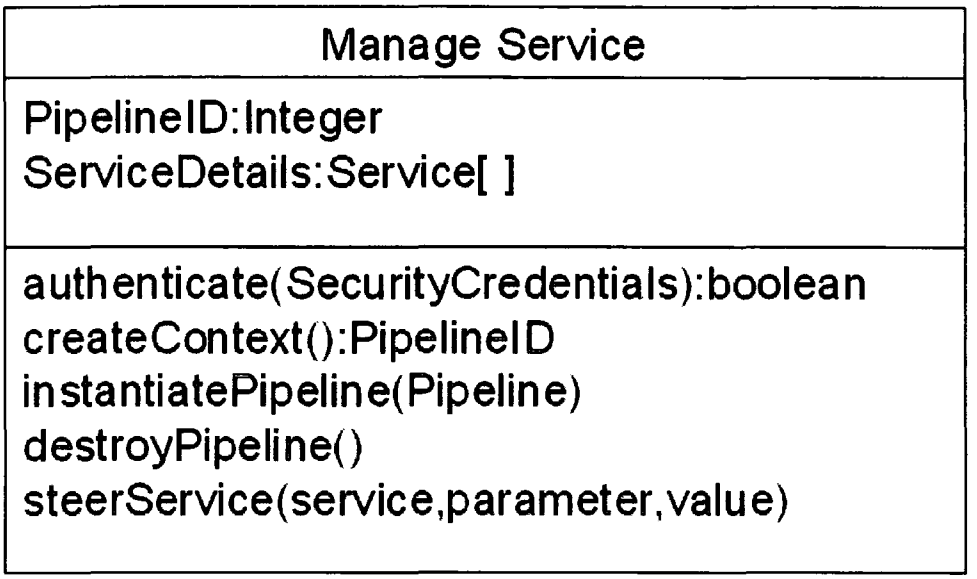


Figure 5.7: Manage Service Class Definition

5.3 Tools

In creating and using visualisation pipelines additional user tools are required. These tools are an important part of the design and creation process and the use of the visualisation pipeline once it is operational but do not impact on the architectural definition of the virtualised visualisation architecture. This section examines the Composition Tool and the User Interface.

5.3.1 Composition Tool

The composition tool is used by the creator of a visualisation pipeline to discover available services, using service discovery services on the grid, and to compose them together to form a visualisation pipeline. The user can then enact the visualisation pipeline using a management service. Once a pipeline has been enacted using the management service it can be left to execute without the composition tool running.

The flowchart in Figure 5.8 shows the operational order of composing a visualisation pipeline before submitting it to the manage service for instantiation.

5.3.2 User Interface

The user interface is the mechanism through which users of the visualisation pipeline can alter the configuration of the pipeline. The user interface is designed to be run on an as needed basis and connects via the management service. The user interface makes use of the steering protocol to issue changes. The user interface can be an entirely stand alone tool or it can be incorporated into the composition tool or into a writer service. Multiple user interface tools can be used with a visualisation pipeline for collaborative sessions. The manage service is responsible for managing these multiple interfaces in this instance.

5.4 Composition

In addition to the services within the pipeline other tools are required to support the development of visualisations. The composition tool shown in Figure 5.9 is used to construct visualisation pipelines. This is achieved by using grid infrastructure services that allow the visualisation services to be discovered, then composed together to form a visualisation pipeline using the composition tool. This then instantiates the pipeline by communicating with the management service which manages the pipeline. The composition tool is no longer required once the pipeline has been created through the manage service for the pipeline to run, however it can be used to make adjustments to the pipeline composition during execution.

The services, filter, map and render have no enforced ordering in the pipeline, however they are typically ordered as filter, map, render.

To describe the composition of pipelines a simple notation based on regular expressions can be used where:

R	*is a read service*
T	*is a transform service*
F	*is a filter service*
M	*is a map service*
P	*is a render service*
W	*is a write service*
.	*denotes a service join*
||	*denotes branches in parallel*
()	*encloses parts of branches for readability*

The notation excludes the manage service as it is assumed that a manage service is present for each pipeline.

The minimum pipeline that can be created is a Read Service and a Write Service. However typically a pipeline would include one or more transform services. This can be written using regular expression notation:

$$\mathbf{R}^{+}\ \mathbf{T}^{*}\ \mathbf{W}^{+}$$

As an example, a pipeline with one of each service in the conventional order as in Figure 5.5 would be represented as:

$$\mathbf{R.F.M.P.W}$$

A pipeline with two branches from the reader service that rejoined at the render service would be described in the notation as:

$$\mathbf{R.((F_1.M_1)||(F_2.M_2)).P.W}$$

The services within the pipeline can be self contained functional units or they can provide an interface to the grid, hiding a highly scalable service, as for example shown in Figure 5.10. This type of service would be required to deal with very large data sets. The figure also shows multiple services collaborating to provide the desired performance. In this manner time critical computation can be done using the local resources with less time critical computation performed at other locations. If a model for grid computation that includes a cost to use a resource is in operation this type of strategy can be used to reduce the cost of computation.

5.5 State in Services

The architectural definition presented thus far is built with stateful services in mind. The use of stateful services mean that each operation called in a service may not be side effect free.

The use of stateful services allows persistent pipelines to be created and for them to be steered. As a minimum each service would hold the following stateful information.

- Location of the input data
- Location of the output data
- Location and identifier of the manage service

The class diagram in Figure 5.11 shows the minimal set of state information that a service would hold.

In addition many services would also contain a set of configurable parameters, for example a service that is providing isosurfacing may contain a parameter that sets the value at which to draw the surface. It may contain another parameter which sets the colour that the surface should be.

For the example above the state information definition may look like.

Standard Stateful Data.

input_locations: URL[]
output_location: URL
manage_service: URL
session_identifier: String

Service Specific Data

surface_value: float min 0.0 max 1.0
colour_r: int min 0 max 255
colour_g: int min 0 max 255

colour_b: int min 0 max 255

The class diagram for a stateful isosurface service is shown in Figure 5.12 this shows how the service builds upon a common stateful service with specific data for the computation that the service will undertake.

The use of state allows a service to relate the results of all computations performed for the same pipeline, this can be important if computation reduction and bandwidth reduction strategies are employed. In a similar manner the use of state allows predictive computations to be undertaken and related to the correct pipeline, allowing low cost and idle cycles to be used on resources.

5.6 Requirements Review

This section demonstrates how the requirements for the architecture have been met by the definition of the architecture outlined.

The requirements of the architecture are restated for completeness.

1. That the virtualised visualisation architecture can run on a variety of machine architectures.

   The first requirement can only be met by the implementation. However by using a service oriented architecture with a connection across a network connecting each service means that the communication mechanism is platform independent and therefore the machine architecture that each service is run on does not affect the visualisation architecture. Grid middleware provides a homogeneous set of tools and services across heterogeneous architectures which simplifies development for different platforms and allows migration of code more easily.

2. That the visualisation can be displayed on a variety of different display platforms, including stereoscopic devices.

   The separation of the write service from the transform services, allows a specialisation of the write service as a display component and the render sub

division of the transform service. This split allows a variety of display services to be written for different display platforms without having to re-implement the render service each time.

3. That the visualisation should be available on the desktop.

   Following from the separation of render and write services, by decoupling the processor and memory intensive task of rendering from the local machine if required a low powered machine can be used to display the visualisation, such as a standard PC. The grid makes the decoupling of the final visualisation from the process to generate the visualisation possible and allows the scalability required to provide the required performance.

4. That the architecture will operate across multiple institutional boundaries, accounting for security issues.

   The use of a standard for communications between services and the use of standard protocols for data transfer will allow the architecture to operate across multiple institutional boundaries.

5. That the architecture will allow multiple scientists to collaborate from different locations, either synchronously or asynchronously.

   To support collaboration the architecture has many features, the decoupling of the composition tool and the user interface means that the pipeline can be manipulated by any user at any time. The manage service allows visualisation sessions to exist and run without any user connected to them. This means that asynchronous collaboration is possible as one user can establish a pipeline and other can connect and view it at a later time. The grid allows this type of pipeline to exist by allowing services to be deployed on remote computers and to persist.

6. The architecture should cope with very large remote data set.

   The scalability model shown in Figure 5.10 allows services to increase the processing power they have available to perform their computations. The

subdivision of transform services to filter, map and render also assists in coping with large data sets in distributing specialised computation to different resources on the grid.

7. The architecture should allow visualisation to be performed interactively.

   The scalability and division of services allows interactivity to be achieved more easily as the services can employ techniques to increase the rate at which they process data. These techniques can be a simple linear scaling of processing power or predictive techniques which preempt the user's needs.

The main features of the architecture that allow it to meet the requirements include the separation of all the stages in the pipeline and that these services can be built in a scalable way. The use of tools for composition and the user interface provides an important degree of separation which the manage service controls.

5.7 Summary

The definition section has outlined the requirements behind the development of the virtualised visualisation architecture, the way in which these requirements have been taken and incorporated into the design. Issues affecting the design of the virtualised visualisation architecture have been explored and explained. The conventional visualisation pipeline has been extended by:

- Adding the grid between services and allowing services to scale using the grid.
- Adding the Manage Service allowing persistent pipelines and asynchronous collaboration.
- Allowing Steering of all services in the pipeline to enable the visualisation to be controlled by both services and users.
- Decoupling the composition from the pipeline to remove the need for the composition tool to remain active during the pipeline's life.

- Decoupling the user interface from the pipeline to allow multiple users to exercise control over the pipeline and allow the development of user interfaces suitable for the devices being used to control the pipeline.

- Allowing specialist render services to cope with tasks such as rendering for autostereoscopic displays.

The development of a virtualised visualisation architecture is driven by requirements as outlined at the beginning of this chapter, the realisation of those requirements is achieved through the implementation of the design. This implementation and the issues encountered in creating that implementation to meet the requirements and the design are discussed in the next chapter.

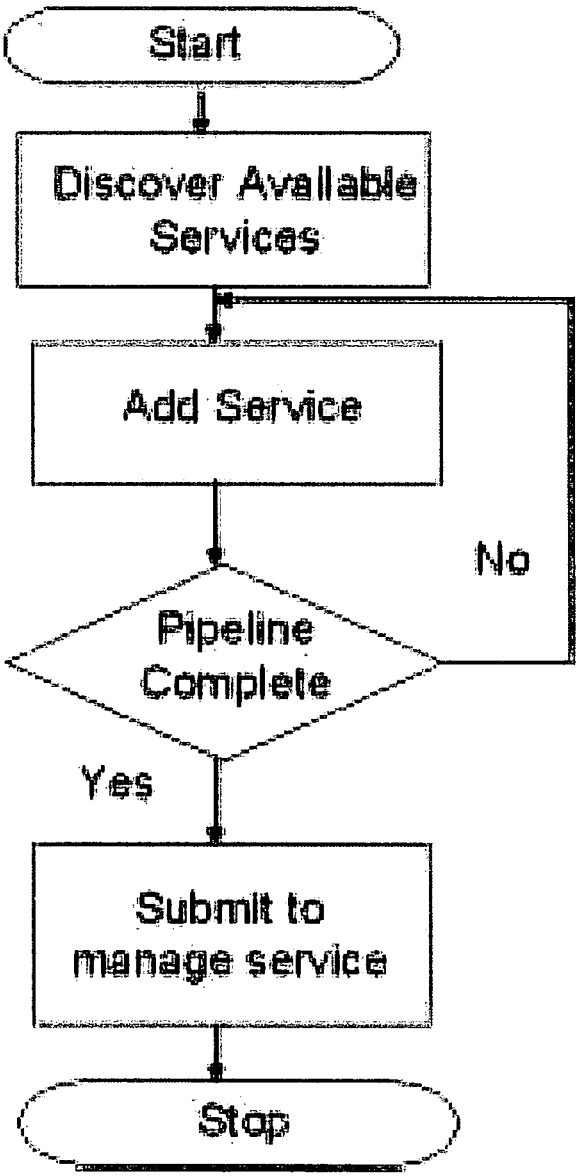


Figure 5.8: Composition Tool Operation Flowchart

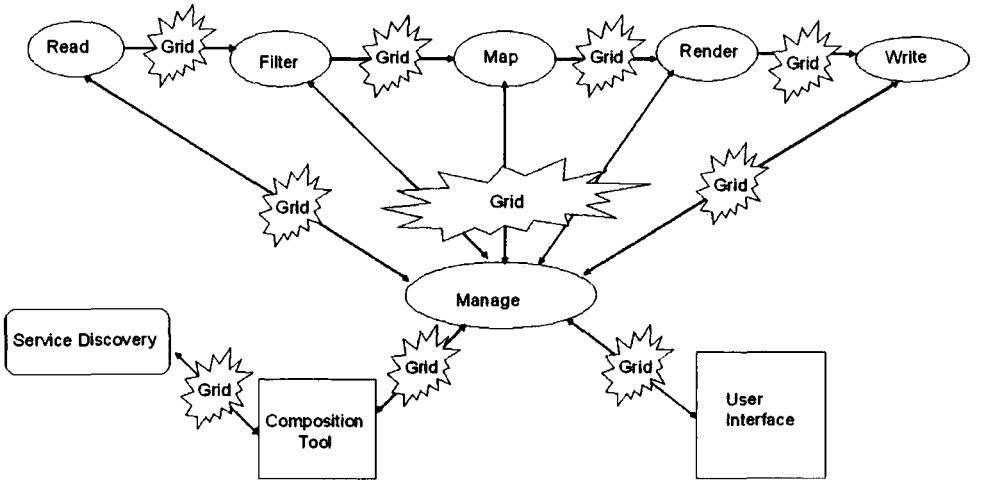


Figure 5.9: Final Pipeline Definition with Support Tools

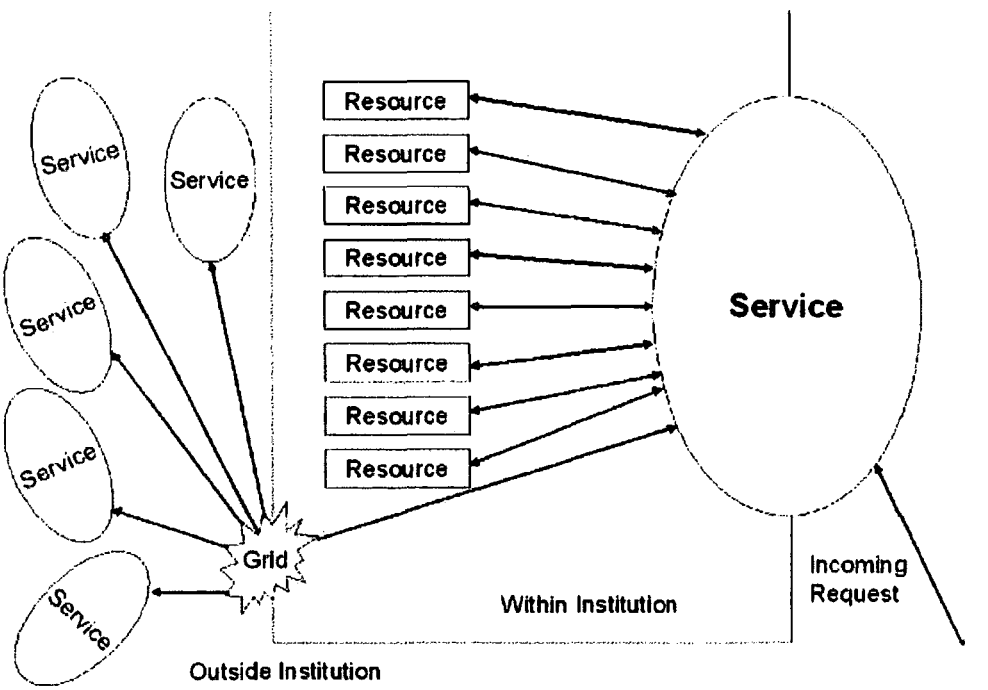


Figure 5.10: How a service can scale

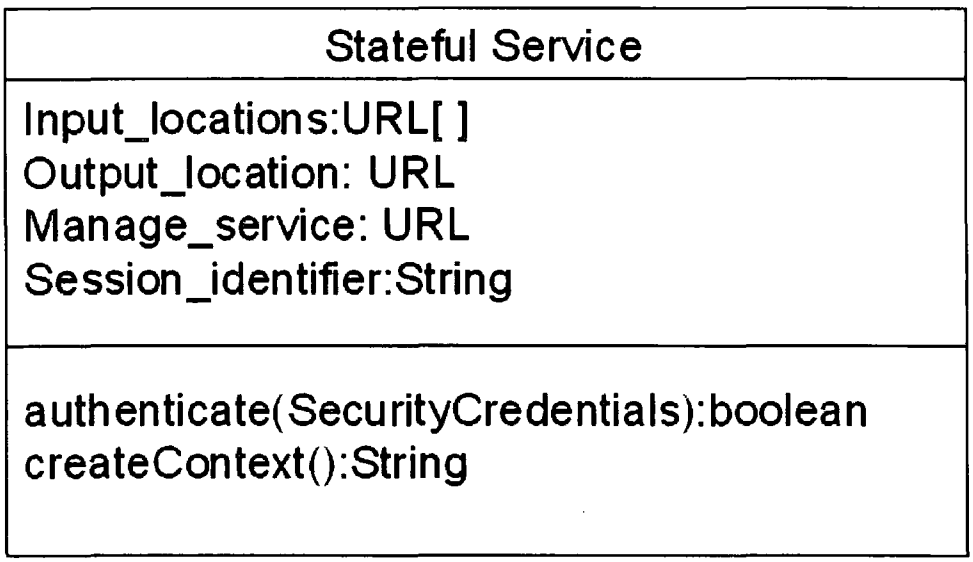


Figure 5.11: Stateful Service Class Diagram

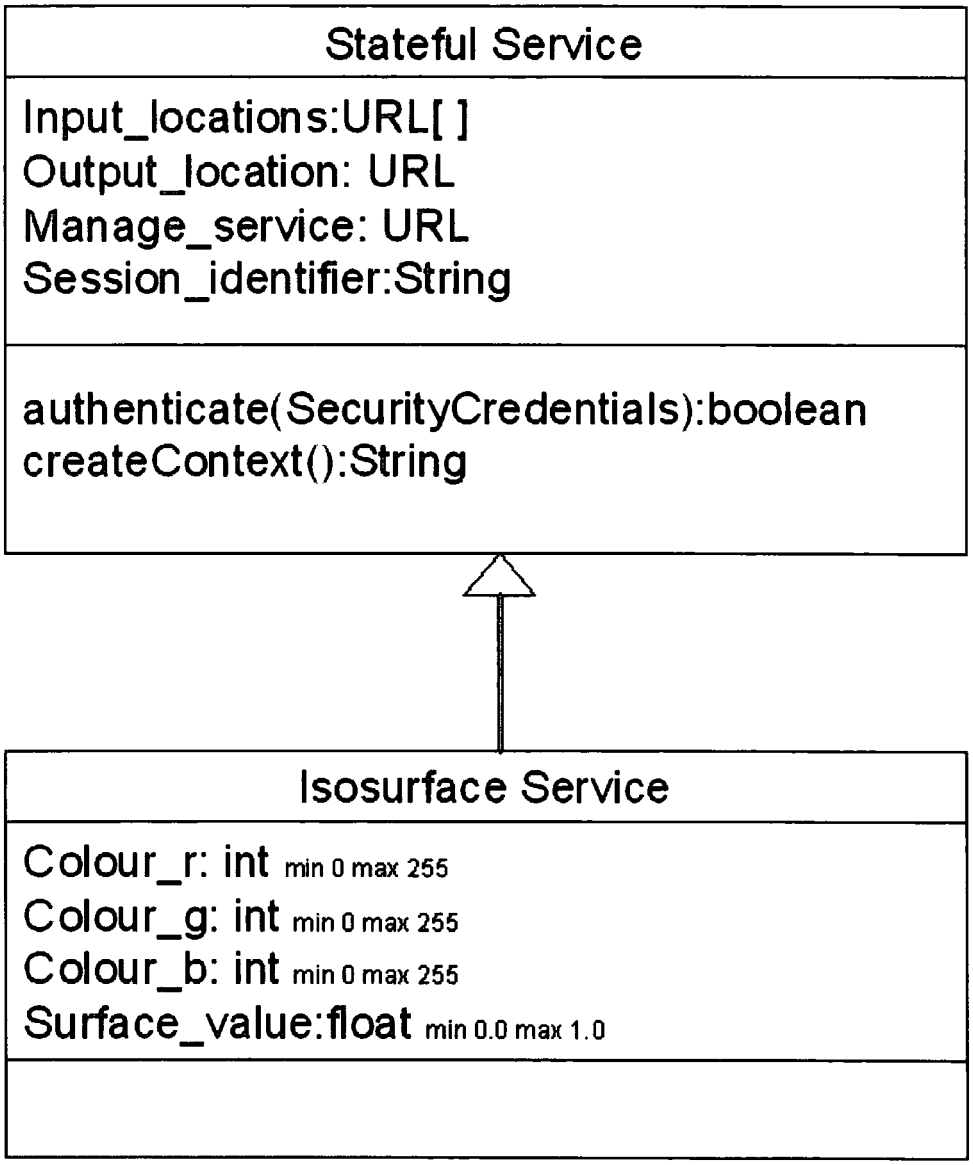


Figure 5.12: Stateful Isosurface Service Class Diagram

Chapter 6

Implementation

This chapter describes three implementations of a virtualised visualisation architecture as defined in chapter five. These implementations were developed as the technology of the service oriented grid was developed and underwent several fundamental changes. Each of the implementations is described here to highlight the issues encountered in developing the virtualised visualisation architecture.

The first and second implementations were done using versions of the Globus Toolkit 3 from the Globus Alliance. The Globus software provides a middleware layer for computing on the grid and has been discussed in more detail in chapter four.

The first implementation was using the version 3 Alpha release of the service based grid software from the Globus Alliance. This software then had a major update before it was released as version 3.0.2 which involved a repackaging of many components in the toolkit and an alteration in the implemention of some functionality; this version of the Globus toolkit was used for the second implementation.

Following the release of the 3.0.2 version of the software a discussion within the grid community took place about the philosophy and development approach to the grid service software, this led to a fundamental shift from stateful grid services to stateless web services. This fundamental shift in the statefulness of services had a big impact on the realisation of the design of the virtualised visualisation architecture and the third implementation.

The final section of this chapter discusses the implementation of the architecture

using stateless web services and the changes that this required to the design of the architecture. An examination of how to introduce state to web services to allow control to be exercised over the pipeline through steering and to use of the visualisation architecture for interactive visualisation and for collaborative visualisation has also been undertaken.

6.1 Globus Toolkit 3 Alpha 2 Implementation

The implementation of the virtualised visualisation architecture using the Alpha 2 release of the Globus Toolkit 3 from the Globus Alliance was the first implementation of the architecture to be attempted. The architectural definition was used for the development of interfaces in Grid Service Description Language (gWSDL), an extension of Web Services Description Language (WSDL) that introduced support for inheritance from multiple gWSDL documents.

This inheritance allowed a base interface common across all services in the architecture to be developed and for this interface to be specialised for the different types of service in the architecture, and again specialised by service developers if they wished to implement any extra functionality in their services. The multiple inheritance also allowed interfaces, such as a steering interface, to be developed. The advantage of this approach is that services can be developed in an iterative manner and new functionality incorporated by inheriting an interface and then adding the functions to the service. The use of these multiple interfaces allows some typing of services which can be of use when constructing visualisation pipelines.

This implementation was started as a prototype proof of concept as it was understood that the release version of the Globus Toolkit 3 would be made available shortly and it was felt that an implementation using the release version of the toolkit would be preferable to one using an Alpha release. For this reason the main focus of the implementation was the construction of services, other issues, including data transport, security, composition and discovery were put to one side during the prototype implementation.

The data transport mechanism chosen for ease and simplicity was the Simple

Object Access Protocol (SOAP), this is the mechanism used by services for message passing and for calling functions in services. The mechanism is inefficient for large data transfers as it encodes messages in an XML wrapper and sends all data as UTF-8 text.

To create a visualisation pipeline a user would for each type of service required call a factory service which would create an instance of the service that the user wanted, the factory would then return to the user an endpoint reference to the service instance created. The user would then be the owner of the instance and can manipulate that service and its properties to alter the behaviour of the service. The user can also destroy the service when they have finished using it.

Each service instance can have Grid Service Data Items, these can be thought of as variables exposed publicly, these can be manipulated and could provide an mechanism for steering the services.

6.2 Globus Toolkit 3.0.2 Implementation

The implementation using the Globus Toolkit 3.0.2 was begun but never completed due to the move to web services as discussed later. However during this implementation the Globus Toolkit 3 Alpha 2 implementation of the architecture was examined and areas for improvement were highlighted.

The main area to be examined was the area of data transport, it was recognised that for the architecture to be suitable for processing large data sets that an alternative data transport mechanism would be required. The type of data transport required was that of streaming, whereby as data is produced it is sent across the network to the next service in the chain. This type of data transport mechanism would allow the start of a large data set to be processed before the end of the data set had arrived. It would also be appropriate for devices that produced a theoretically infinite data set, such as environmental sensors.

Due to the batch processing origins of grid software and in particular the Globus Toolkit, such a data transport mechanism has not yet been implemented for use in a service based architecture.

A compromise solution to the problem was decided upon for this implementation, that of HTTP Streaming, a mechanism that allows a file, referenced by a URL to be read across the network. To use this solution each service has to write its output locally to a file and the service must return a link to that file. One approach to the returning of the URL would be for the service to wait until it had started processing data and writing to disk and then return the URL, that way the service next in the pipeline would know that the file existed and was populated with data. However due to the nature of distributed communications, services have a limited time in which to issue a response before the calling service times out believing that the service it is calling did not receive the message or has ceased to be in existence and therefore is unable to process the request. It is not determinable as to when a service will begin processing data as each service is dependent on the services higher in the chain from it, as such simply extending the timeout level for a service would not solve the problem. Extending the timeout value to solve the issue would also have the effect of increasing the time before a true service failure was noticed. It is felt that this would be an undesirable side effect and one that should be avoided.

The solution adopted was for a service to return a URL to the calling service before it began processing the data from the previous service in the pipeline. The URL returned was a system generated one, consisting of the local machine name and directory where data was stored combined with a millisecond time stamp from the local system, it is recognised that this URL generation technique is not guaranteed to be unique, however it is felt that the case of two or more identical URLs being generated on the same host was sufficiently remote to make it highly unlikely. An extension to the URL generation mechanism would be the use of a random number in addition to the time stamp which would reduce the probability of identical URLs further.

An implementation of some services using the Globus Toolkit 3.0.2 was completed to test the HTTP streaming as a data transport mechanism, it was found to be less than ideal but an improvement upon the use of SOAP as a data transport mechanism. One of the issues with the HTTP streaming mechanism was the incomplete transfer of data, this was observed on several occasions, it is thought

that incomplete data transfers occurred when a service was reading data faster than the providing service was outputting the data, in this case the HTTP mechanism reached the end of the file written to disk and terminated the data transfer whilst the providing service continued to write data after this point. A solution to this issue has not been discovered as yet.

The implementation of the virtualised visualisation architecture was not completed as the grid community entered a discussion about the future direction of grid services, the outcome of these discussions was that the future grid services would be based upon web services. The details of changes and the timescale for these changes were still uncertain at this stage. It was decided therefore that a move to web services, would be a technology change that would be better to make sooner rather than later due to the fundamental difference between grid services and web services over state.

6.3 Web Service Implementation

The Web service implementation brought together development from both of the previous implementations of the virtualised architecture as the implementation using Globus Toolkit 3.0.2 had not been completed.

The web service implementation also required re-examination of the original design for the implementation of the architecture due to the stateless nature of web services. The other area that required re-examination was the interfaces written in gWSDL as the WSDL that would be used with web services does not support inheritance.

The web service implementation was able to reuse some Java code from previous implementations of individual services to perform the filter, map and render parts of the pipeline.

6.3.1 Stateless Services

An examination of the interfaces developed for the initial implementation of the architecture in light of removing the requirement for inheritance showed that these

would need to be rewritten. It was decided to split the five types of service, read, filter, map, render and write into three groups, read services, transform services and write services.

The read service group contains just read services and the interface for this type of service was written with a focus on serving data, it did not need to provide a mechanism for the service to take in data.

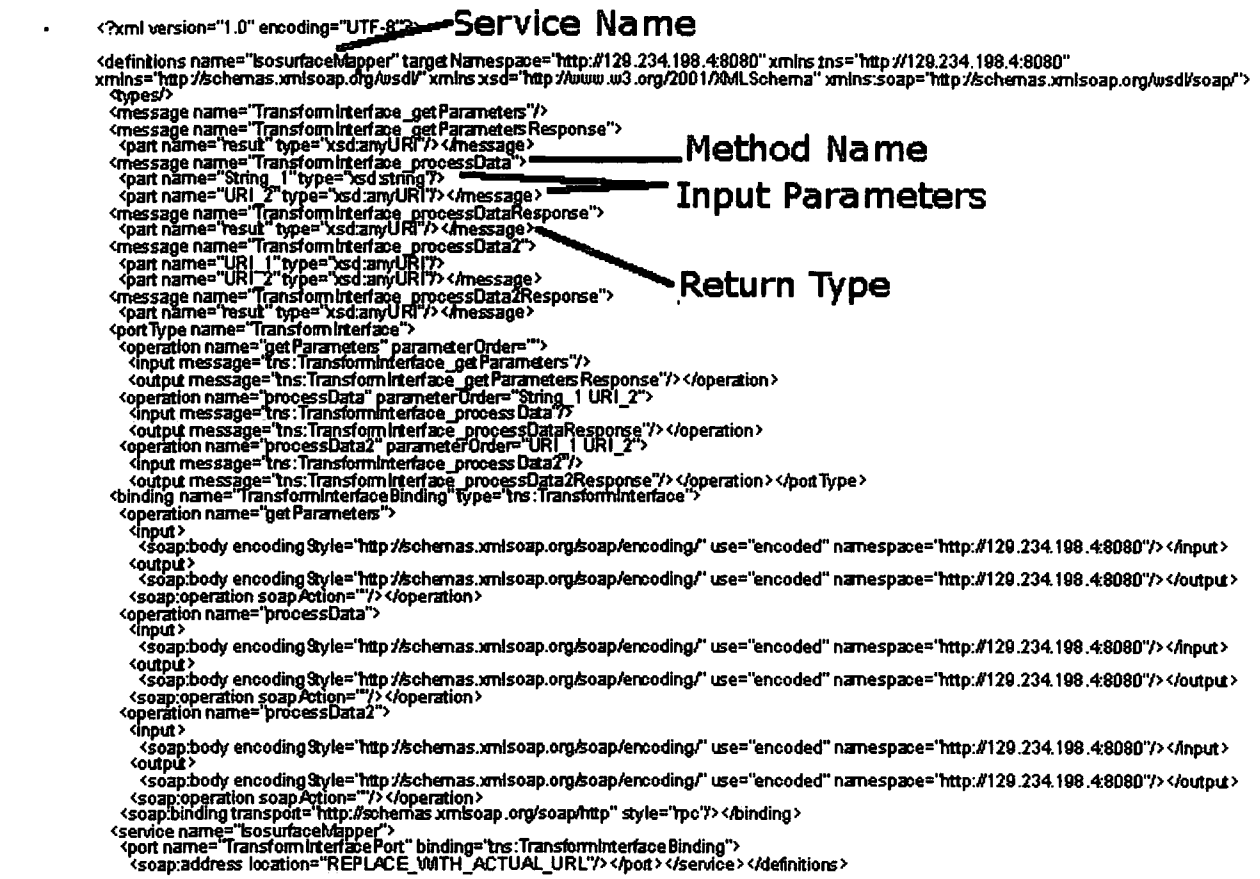
The transform services group contained the filter, map and render services, these services were grouped together as they each take in data and produce an output. These services were termed the transform group as their function is to change, or transform, the data they take in to produce a new output.

The final group is the write service group containing the write services, this group is simply a consumer of data and does not need to provide a mechanism for the service to return data.

With these three WSDL interfaces defined, five service interfaces were defined, one for each of the service types, the read service interface using the read group WSDL interface. The filter, map and render services each having their own java interface using the transform group WSDL interface and the write service having a java interface using the write group WSDL.

The decision to use different Java interfaces for the services, especially in the transform group was made to maintain the functional differential between the services. Each service in the group does the same general task, in the transform group this is read in data, perform some operation on that data and return an output; but they have different specialties within that generalisation. Filter services, reduce the data set size and/or change the data format; map services, map raw data to a visual representation; render services, take a visual representation and produce a geometric output or an image output.

Web service instances are not created in the same way as instances of grid services, web services run as a single instance in a web service container, there are no web service factories, this means that all users interact with the same instance of a web service. A user therefore has no ownership over the web service, this is why web services are stateless. A web service also has no grid service data items, nor




```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="IsosurfaceMapper" targetNamespace="http://129.234.198.4:8080" xmlns:tns="http://129.234.198.4:8080"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
 <types/>
 <message name="TransformInterface_getParameters"/>
 <message name="TransformInterface_getParametersResponse">
  <part name="result" type="xsd:anyURI"/></message>
 <message name="TransformInterface_processData">
  <part name="String_1" type="xsd:string"/>
  <part name="URI_2" type="xsd:anyURI"/></message>
 <message name="TransformInterface_processDataResponse">
  <part name="result" type="xsd:anyURI"/></message>
 <message name="TransformInterface_processData2">
  <part name="URI_1" type="xsd:anyURI"/>
  <part name="URI_2" type="xsd:anyURI"/></message>
 <message name="TransformInterface_processData2Response">
  <part name="result" type="xsd:anyURI"/></message>
 <portType name="TransformInterface">
  <operation name="getParameters" parameterOrder="">
   <input message="tns:TransformInterface_getParameters"/>
   <output message="tns:TransformInterface_getParametersResponse"/></operation>
  <operation name="processData" parameterOrder="String_1 URI_2">
   <input message="tns:TransformInterface_processData"/>
   <output message="tns:TransformInterface_processDataResponse"/></operation>
  <operation name="processData2" parameterOrder="URI_1 URI_2">
   <input message="tns:TransformInterface_processData2"/>
   <output message="tns:TransformInterface_processData2Response"/></operation></portType>
 <binding name="TransformInterfaceBinding" type="tns:TransformInterface">
  <operation name="getParameters">
   <input>
    <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="http://129.234.198.4:8080"/></input>
   <output>
    <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="http://129.234.198.4:8080"/></output>
   <soap:operation soapAction=""/></operation>
  <operation name="processData">
   <input>
    <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="http://129.234.198.4:8080"/></input>
   <output>
    <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="http://129.234.198.4:8080"/></output>
   <soap:operation soapAction=""/></operation>
  <operation name="processData2">
   <input>
    <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="http://129.234.198.4:8080"/></input>
   <output>
    <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="http://129.234.198.4:8080"/></output>
   <soap:operation soapAction=""/></operation>
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/></binding>
 <service name="IsosurfaceMapper">
  <port name="TransformInterfacePort" binding="tns:TransformInterfaceBinding">
   <soap:address location="REPLACE_WITH_ACTUAL_URL"/></port></service></definitions>
```

Figure 6.1: WSDL for a transform service

any similar concept. Each operation within a web service must be self-contained and have no impact or bearing on any past or future operations.

As web services are always running within the container, this means that the overhead associated with connecting to a service is reduced as a service instance does not have to be created for each user; the first call to the service when a container is started may have this instantiation overhead associated with it but all subsequent calls, by any user, are instantiation overhead free.

The choice of technology used to deploy the web service architecture was made after surveying the available technologies. The Tomcat container from the Apache Foundation was chosen as it had been successfully used in the Globus Toolkit as the basis of the grid service container, it had a wide community user group and many tools and tutorials designed to work with it. The Tomcat container is also

open source and free of a license fee and runs on many operating systems a vital requirement of a container for this type of architecture.

The Sun Java Web Services Developers Kit, which also contains Tomcat as a container environment, was also chosen to help with the development of Java web services. The kit provides tooling to aid the conversion from Java to WSDL and from WSDL to Java when creating services. The kit also contains assistance with service packaging.

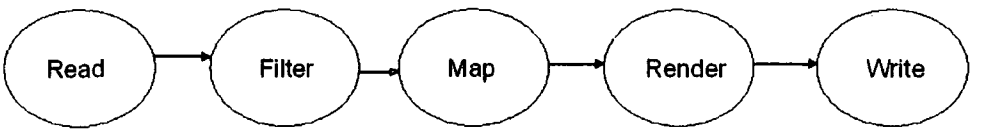


Figure 6.2: Overview of Implemented Services

In the initial web service implementation of the architecture as shown in Figure 6.2 the issue of state was ignored and a 'one shot' pipeline approach was followed. That is, a user establishes a pipeline and uses it once, if the user wishes to reuse the pipeline they must start over and reconstruct a pipeline and all data must be reprocessed. This is an inefficient method of pipeline use but successfully avoids the complications required by state management. An example of a pipeline that was implemented is shown in Figure 6.3

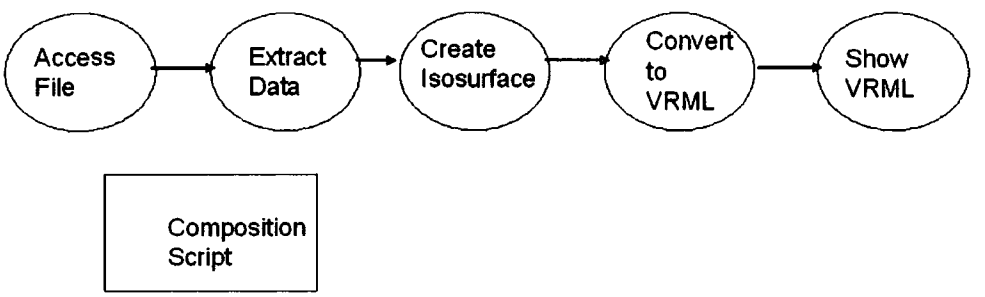


Figure 6.3: Example of an implemented pipeline

In this implementation external libraries such as the Visualisation Toolkit (VTK) were used to provide functionality for some services, this use of external libraries whilst allowing reuse caused complications of its own. The design of these external libraries is based on reading data from file and outputting to screen. In the web

service model, the preferred approach was to read from a data stream and output to file. This design model for the external libraries meant that they did not operate within the web service container, due to the low level calls that the libraries make and the sandbox security model of the web service container. A work around to this was to read all the data that was to be processed to a file on disk, to pass a local file path via a socket connection to a daemon running which calls the library as is shown in Figure 6.4.

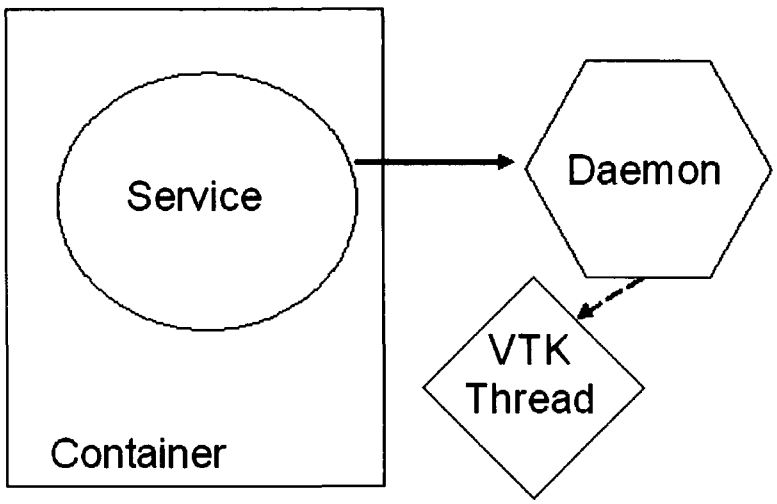


Figure 6.4: Using External Libraries with Services

This method is used in a map service where, for example, isosurfacing is performed. The data is read from the filter service to a file on disk, it is then processed using the VTK library whose output is also written to disk. This data can then be streamed to the render service. To achieve this the URL returned by the filter service is passed as a parameter to the map service which polls the file, waiting until it is written out by the filter service. When the file has been written by the filter service it is read by the map service to local disk. The location of the local file is then passed into a daemon that interfaces to the VTK library via a socket connection. The VTK library is then used to process the data file and produce an output file. It is the URL of the output file that is returned by the map service method call and used by the render service to obtain the data that it requires.

The transfer of data, as with the previous implementation, is achieved using the

Java COG kit from Globus to provide HTTP streaming of data to a service.

To use stateful services within a web service based architecture change are required to the design and implementation of the individual services, these changes are discussed in the next section.

6.3.2 Introducing State

If state is required in the virtualised visualisation architecture then changes to the design of the web services must be made. The services must be designed to explicitly manage state and the steering of parameters.

A decision must be made as to how state should be managed within the architecture, should each service manage state itself or should state management be centralised for the creating of a shared context between services.

The approach outlined allows each service to be concerned with the parts of state management that directly affect it without having to manage all the services that are using it. The addition of context allows services to be steered as web services are able to be reused without having to resubmit all parameters.

The addition of context also allows the use of intelligent data management, if a set of steering parameters has been used before and the data is still cached then that data can be reused rather than a service having to re process it. Services can also be checkpointed allowing interesting points in the visualisation to be saved and restored at a later date. The diagram in Figure 6.5 shows the logic for dealing with messages that have a state identifier when they are processed by a web service.

Local vs. Global State

The addition of state to services can happen at different levels of abstraction. Each service can maintain the global state of the pipeline of which it is part, alternatively each service can just maintain state information relating to the computation that is is undertaking.

If global state is implemented each service must contain information about users, security policies and the connections between all services in the pipeline. When a change happens in one part of the pipeline this change must be reflected at all

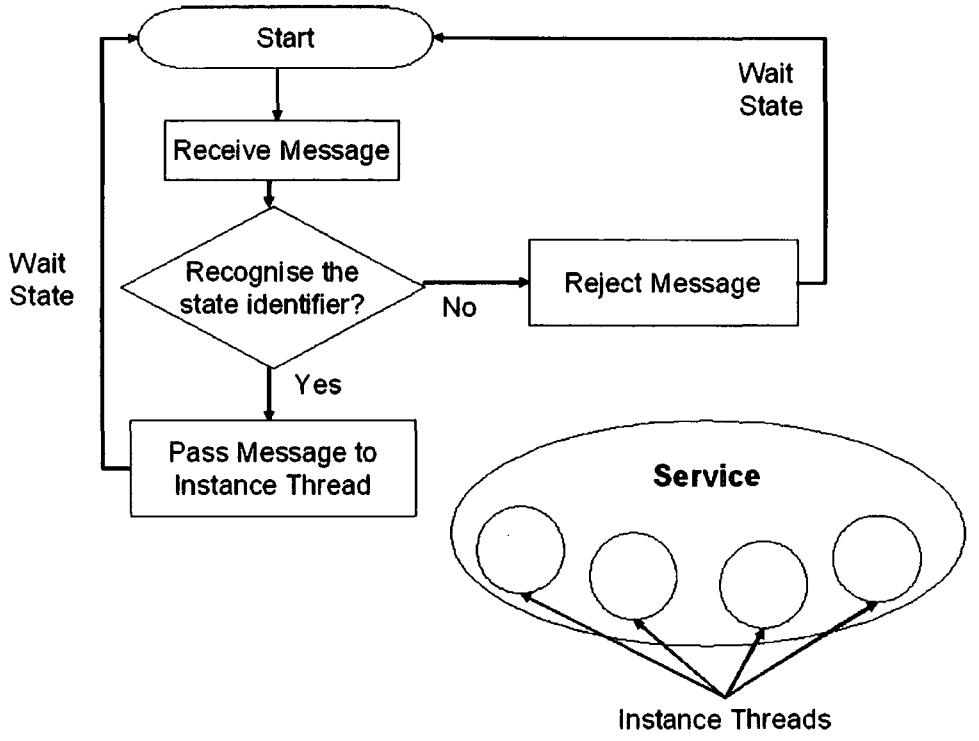


Figure 6.5: Stateful Message Logic

other services. This type of state management has the advantage that each service does not need to rely on any third party if and when it requires access to state information however it introduces a significant overhead in maintaining the state and synchronizing across all services in the pipeline.

Local state at each service reduces the overhead of maintaining state across the pipeline but introduces a requirement for an external maintainer of state. In the virtualised visualisation architecture this role is taken on by the Manage service. The manage service can maintain details of all users permitted to access the pipeline and proxy on their behalf to services, it can maintain details of connections between services. In the case of service failure a new service can be introduced and only the manage service needs to be aware of the change.

6.4 Stereo Render Service

A item of research aligned to the development of the virtualised visualisation architecture a stereo capable render service was developed by Lock [Lock, 2003] who evaluated how different rendering applications could be distributed to provide a stereo capable render service, to this system a number of interfaces were provided including a web service interface to which other web services could provide data for rendering.

This research shows how a stereo capable render service can be built and how external developers can generate services that can be used as part of the virtualised visualisation architecture.

The research also examined a variety of "behind the scenes" distribution mechanisms highlighting the importance of choosing the correct distribution mechanism for the task being approached.

6.5 Summary

Three iterations of the implementation of the virtualised visualisation architecture have been discussed, moving from stateful grid services to stateless web services. Each of these iterations in the development highlighted issues with underlying Grid middleware and the mechanisms for achieving the desired results. Work arounds to these problems have been discussed and a possible solution for achieving state with web services has been presented.

The parts of the pipeline implemented through the iterations of architecture were; read,map,filter, render and write services capable of producing visualisations for two case study scenarios and a prototype composition tool. The parts of the implementation that were not achieved was; state management in services, the Manage service, and the user interface tools.

In producing the services that were able to produce visualisations of the case study scenarios the end to end performance of these pipelines could be evaluated.

Chapter 7

Experiments and Scenarios

In order to evaluate the virtualised visualisation architecture, two regimes were used. Quantative experiments to monitor the performance of the pipeline and qualitative scenarios to highlight how the architecture adapted to different problem classes and to give a user viewpoint evaluation.

7.1 Scenarios

Scenarios provide a qualitative evaluation of the virtualised visualisation architecture. They allows different problem classes to be examined and allow the architecture to be examined from a user perspective.

7.1.1 X-Ray Crystallography

This scenario is taken from the chemistry domain, the X-Ray crystallography field is one that examines the composition of crystals by reducing their temperature and exposing them to X-Rays. The way the X-Rays are affected by the crystals is measured and the structure of the crystal and the electronic field potential around atoms within the crystal can be measured.

To visualise this data requires two types of visualisation be brought together. The first type of visualisation is a representation of the atoms and their connections within the crystal. The representation used for this is a standard for the chemistry field. Atoms are spheres of differing colours and the bonds between atoms are shown

as lines or bars. This type of representation is known as a ball and stick model.

The second type of visualisation used is an isosurface. This is a standard visualisation technique for fitting a surface through points of equal value. This type of surface is used in this representation for the electric field strength within the crystal. The value at which the surface is fitted can be chosen by the user.

The visualisation pipeline for this scenario is shown in Figure 7.1 and consists of a data service which contains the data required for both parts of the visualisation, the data in this scenario is approximately 9MB. The data is split using two filter services, one which extracts the information required for the ball and stick model and one which extracts the information required for the isosurfacing. A mapping service to create the ball and stick model is used and another mapping service is used to create the isosurface.

The render service brings the output of the two mapping services together before it is displayed to the user through a write service.

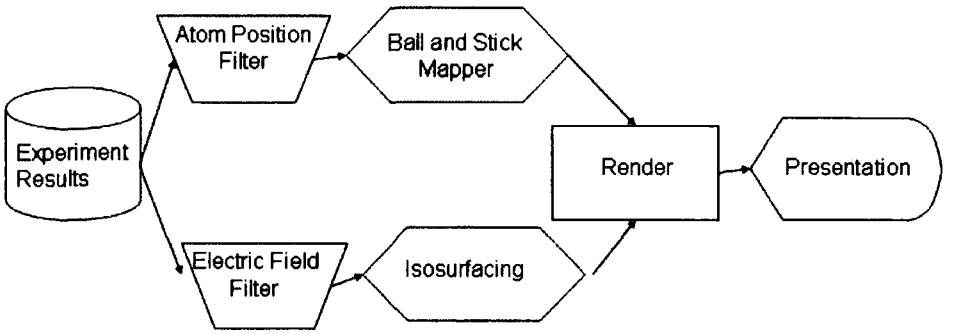


Figure 7.1: X-Ray Crystallography Pipeline

7.1.2 Dark Matter Simulation

This scenario is drawn from the Physics domain, in particular the cosmology area. Simulations of the early universe are run to determine possible distributions of dark matter throughout the universe from the time of the big bang forward. The output of the simulation is a data structure showing the possible distribution of dark matter at that time. The simulation contains approximately 10 billion particles and each

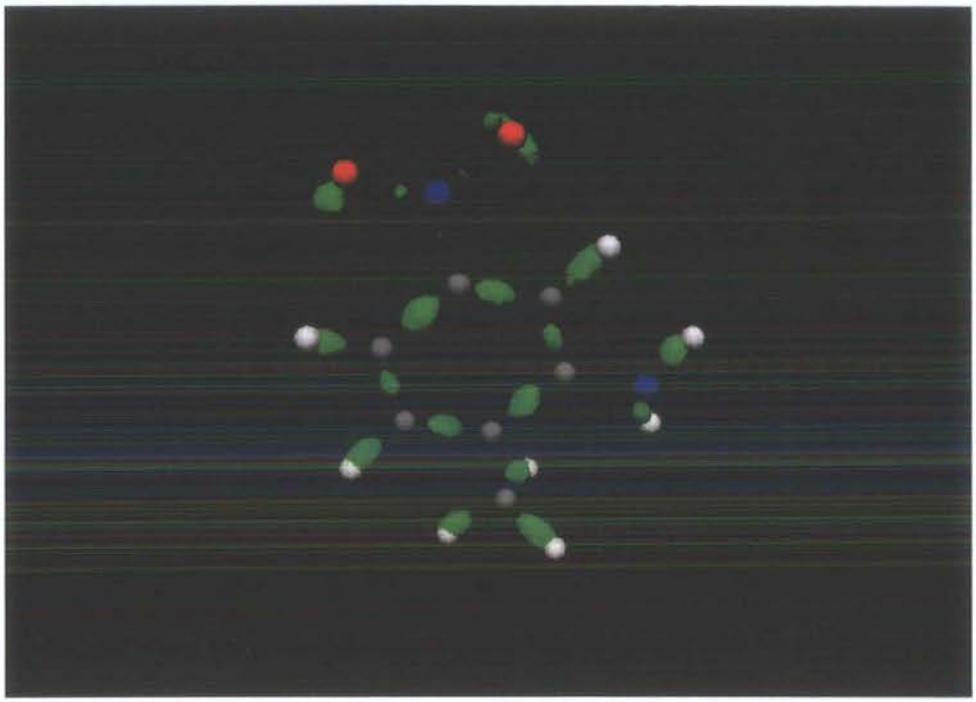


Figure 7.2: X-Ray Crystallography Results

time step in the simulation produces an output of around 300GB. If all time steps are taken together a total data set size of over 15TB is gathered.

The visualisation of this data requires a multi resolution approach, it is not practical, even with ultra-high resolution displays to present all 10 billion particles for display. It is more appropriate to show the general structure of the distribution within the simulation. When areas of interest are found it is required to view these areas in more detail and eventually at the individual particle level.

The pipeline to achieve this type of visualisation requires services to determine the density of particles throughout the simulation to present an abstract level view of the data using an initial 40MB dataset.

7.2 Experimental Evaluation

This section outlines the experiments performed on the architecture, to collect performance statistics for evaluation. The section explains how the architecture was analysed and the results of the experimentation.

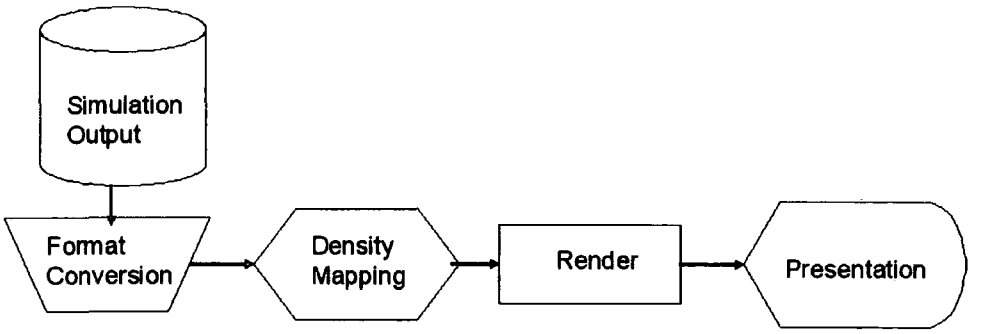


Figure 7.3: Dark Matter Simulation Pipeline

7.2.1 Experimental Procedure

To perform the experiments a grid of standard desktop personal computers (PC) was established. These PCs varied in specification from relatively low powered Celeron processors to Pentium 4 processors. All machines were uniprocessor and installed with a similar software configuration based upon Red Hat Linux with Apache Tomcat installed to host the web services.

The same computers within the grid were used in comparable experiments across scenarios to avoid differing results due to different capability machines.

Each experiment was run fifteen times, this allowed a series of timings to be gathered and for statistical analysis to be performed on the results.

The services were composed together using a script to maintain consistency between experiments and to reduce the impact of any human factors. The experiments were run manually sequentially by executing the service composition script, recording the start time as output by the script and recording the time output by the display client after display of the visualisation is completed. There was no set delay between execution of the script and grid machines were not restarted or reinitalised to reflect the way in which operational grid machines would usually operate.

The experiments were conducted during the normal UK working day, avoiding times such as lunchtime, this was to reduce the impact of variations in network traffic and the effect that this would have on the experimental results.

The data collected during each experimental run was the wall clock time to completion, that is the time from the script being executed to the image being

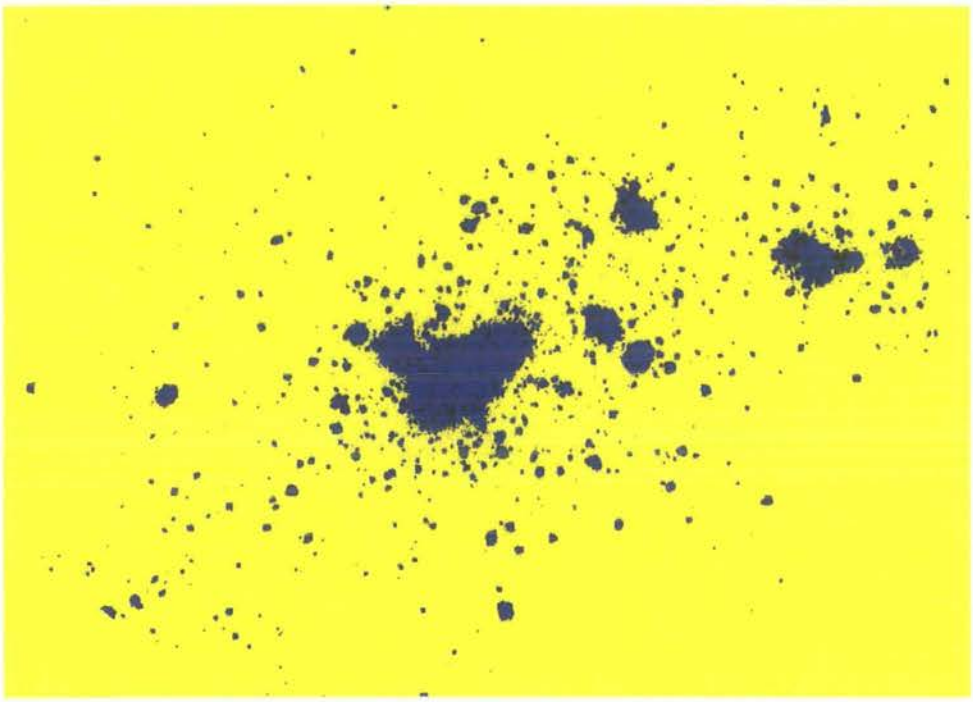


Figure 7.4: Dark Matter Simulation Result

displayed on the presentation service display. The same computer was used for pipeline composition and the presentation service so timings from the system clock could be used to maintain an accurate and consistent measurement.

7.2.2 Experiments

This section describes the experiments that were used to evaluate the architecture.

1. The first experiment run was to establish a control experiment against which the results of other experiments could be compared. The experimental set-up was to run all of the services on a single machine with the data stored locally on that machine. It was also ensured that no other user processes were executing on the machine.

2. The second experiment was to establish the impact of reading the data set from a locally remote service, that is one on the same campus network but not within the same physical network segment. The experimental setup for this experiment was identical to that in experiment one apart from the data

service, which in this experiment was run on a remote computer within the campus network. This experiment was designed to highlight the impact of network distance on the performance of the architecture.

3. The third experiment was again designed to establish the impact of reading data from a remote service, in this experiment however the data would be outside the campus network. The experimental setup for this experiment was identical to the second experiment, apart from the location of the data, which in this experiment was located outside the campus network. This experiment was designed to highlight the impact of network traffic on the performance of the architecture.

4. The fourth experiment was to establish the performance difference by running branches of the pipeline on different computers. In this experiment the data service was run on one computer within the grid and each branch of the pipeline was run on a different computer. This experiment was designed to assess the benefits of distributing computation across multiple machines within a grid.

5. The fifth experiment was to establish the performance difference by running all of the services on different machines. The experimental setup for this experiment involved running each service in the pipeline on a different machine within the grid.

6. The sixth experiment was to establish the impact that locally running processes could have on the performance of the architecture. The experimental setup for this experiment was identical to that of experiment one with the addition of a separate user level processor intensive task. The task run was the mprime tool which is designed to stress the processor subsystems of a computer and as such simulates the effects of heavy use of a resource.

The different experiments tested the architecture under a variety of different circumstances to see how it would perform with possible conditions that would affect the architecture when it was deployed on the grid.

7.3 Results

This section contains the results from the experiments above. The experiment number relates to the number above describing the experiments. The run number relates to the iteration of the experiment.

7.3.1 X-Ray Crystallography Experiments

Experiment One

The first experiment run was to establish a control experiment against which the results of other experiments could be run. The experimental set-up was to run all of the services on a single machine with the data stored locally to that machine. It was also ensured that no other user processes were executing on the machine. The table A.1 shows the results from Experiment one and Figure 7.5 shows these results as a bar chart.

The mean of the data is 19.8 seconds and the standard deviation is 1.11

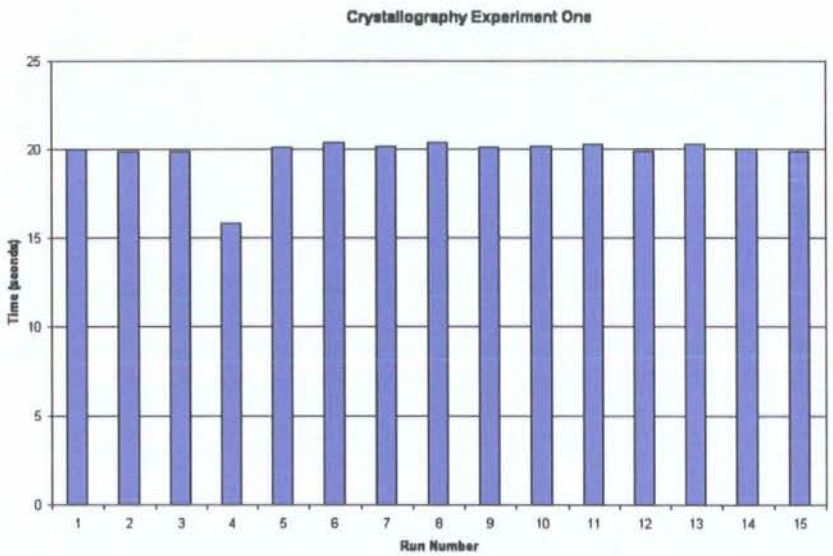


Figure 7.5: Chart of Experiment One Results

Experiment Two

The second experiment was to establish the impact of reading the data set from a locally remote service, that is one on the same campus network. The experimental setup for this experiment was identical to that of the control experiment apart from the data service, which was run on a remote computer. This experiment was designed to highlight the impact of network distance on the performance of the architecture.

The experimental data is shown in Table A.2 and in Figure 7.6 as a bar chart.

The mean of the data is 20.7 seconds and the standard deviation is 0.18

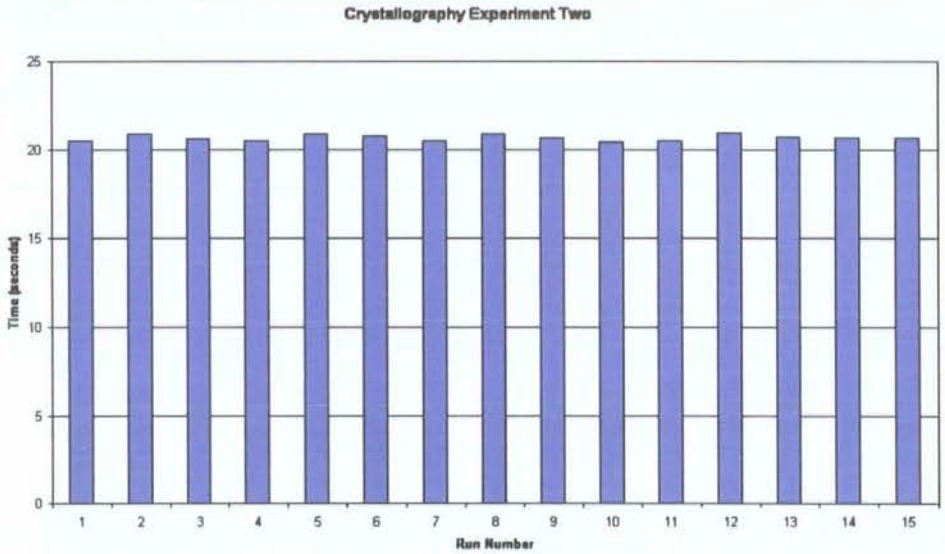


Figure 7.6: Chart of Experiment Two Results

Experiment Three

The third experiment was again to establish the impact of reading data from a remote service, in this experiment however the data is outside the campus network. The experimental setup for this experiment was identical to the second experiment. Again this experiment was designed to highlight the impact of network distance on the performance of the architecture.

The experimental data is shown in Table A.3 and in Figure 7.7 as a bar chart.

The mean of the data is 20.3 seconds and the standard deviation is 0.19

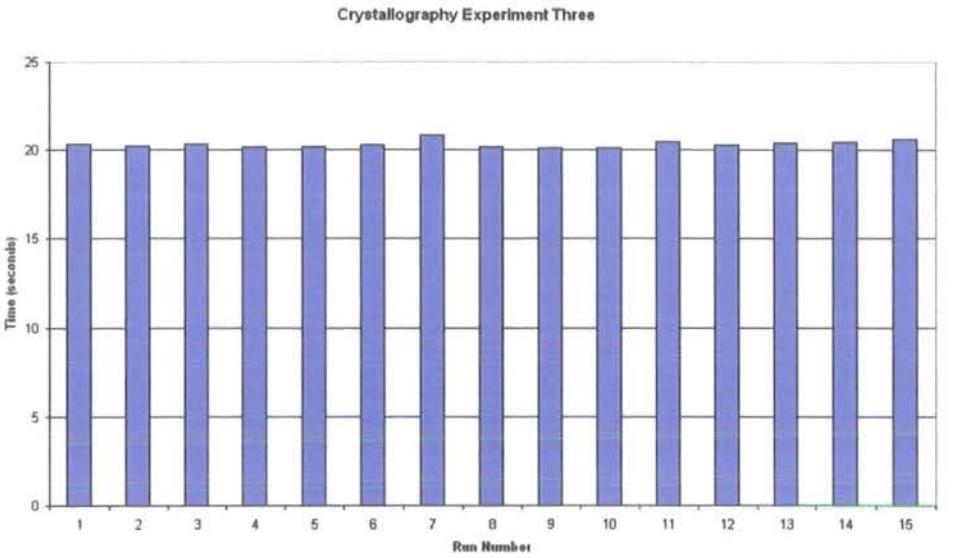


Figure 7.7: Chart of Experiment Three Results

Experiment Four

The fourth experiment was to establish the performance difference by running the two branches of the pipeline on different computers. In this experiment the data service was run on one computer, the ball and stick extraction and modeling services were run on another computer and the isosurfacing service was run on a different computer.

Table A.4 shows the experimental data for this experiment and Figure 7.8 shows this data as a bar chart.

The mean of the data is 20.0 seconds and the standard deviation is 0.18

Experiment Five

The fifth experiment was to establish the performance difference by running all of the services on different machines, the experimental setup for this experiment was to run each service on a different machine.

The experimental data is shown in Table A.5 and in Figure 7.9 as a bar chart.

The mean of the data is 20.7 seconds and the standard deviation is 0.16

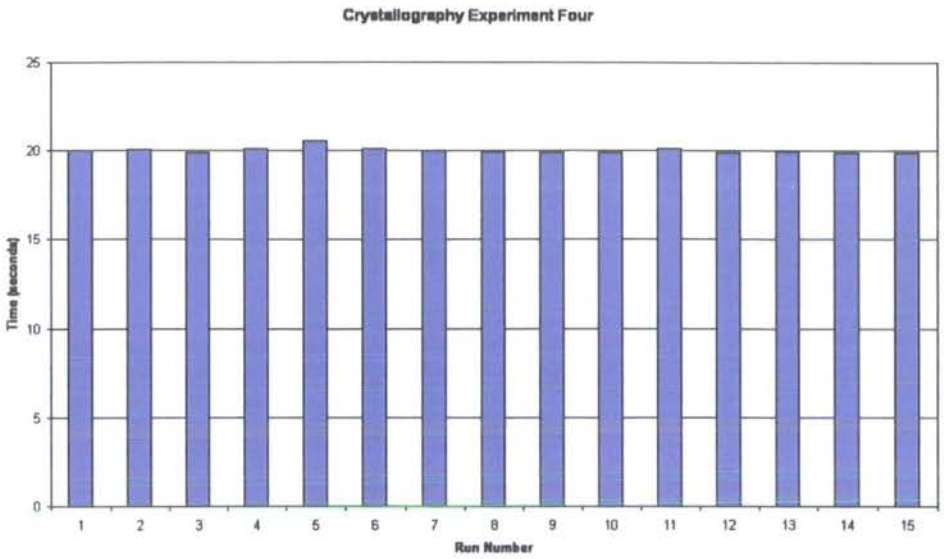


Figure 7.8: Chart of Experiment Four Results

Experiment Six

The sixth experiment was to establish the impact that locally running processes could have on the performance of the architecture. The experimental set up for this experiment was the same as the first experiment except that a processor intensive user process was executed at the same time to simulate the effects of heavy use of the resource by other users.

The experimental data for this experiment is shown in Table A.6 and also in Figure 7.10 as a bar chart.

The mean of the data is 41.9 seconds and the standard deviation is 77.1

7.3.2 Dark Matter Simulation

This section reports the results of the Dark Matter Simulation scenario being run through the same experimental series as the X-Ray Crystallography data, this scenario made use of a simpler visualisation pipeline but a larger dataset.

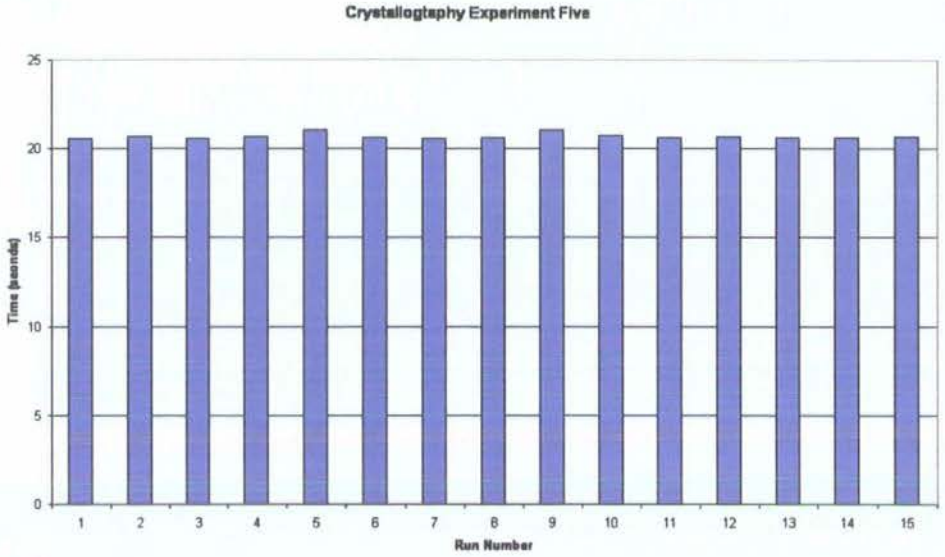


Figure 7.9: Chart of Experiment Five Results

Experiment One

All services in this experiment are run on the same machine to establish a base case against which to compare.

The results of this experiment are shown in Table B.1 and Figure 7.11 as a bar chart.

The mean of the data is 93.4 seconds and the standard deviation is 0.053

Experiment Two

This experiment involved all services apart from the data service being run on the same machine as experiment one, the data service was run on a remote machine within the campus network but on a different network segment.

Table B.2 shows the experimental data for this experiment and Figure 7.12 shows this as a bar chart with error bars indicating one standard deviation.

The mean of the data is 120.9 seconds and the standard deviation is 0.49

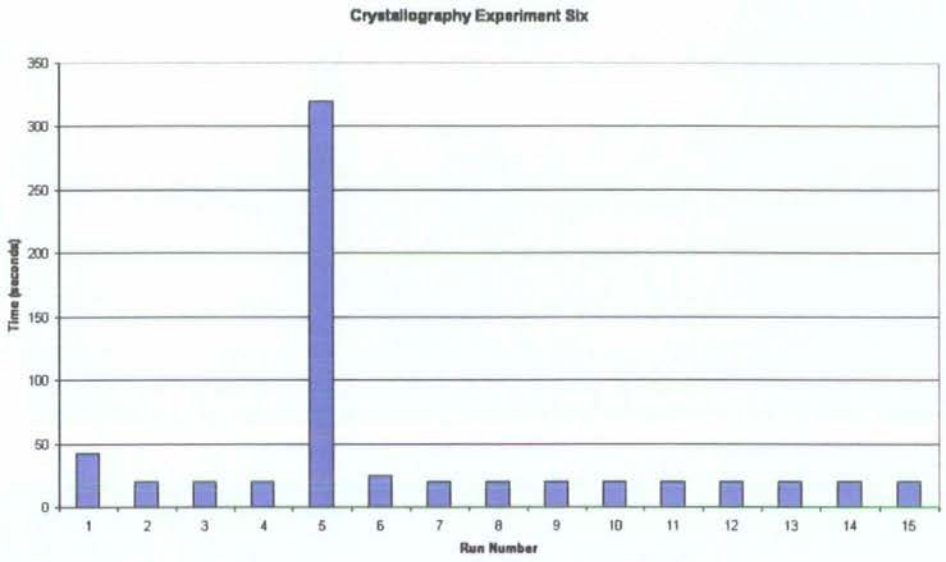


Figure 7.10: Chart of Experiment Six Results

Experiment Three

This experiment involved all services apart from the data service being run on the same machine as experiment one, the data service was run on a remote machine outside the campus network.

Table B.3 shows the experimental data for this experiment and Figure 7.13 shows this as a bar chart.

The mean of the data is 126.4 seconds and the standard deviation is 5.4

Experiment Four

This experiment was not run for this scenario as the pipeline does not have multiple branches.

Experiment Five

This experiment involved the services within the pipeline being run on different machines within the local grid.

Table B.4 shows the experimental data for this experiment and Figure 7.14 shows this as a bar chart.

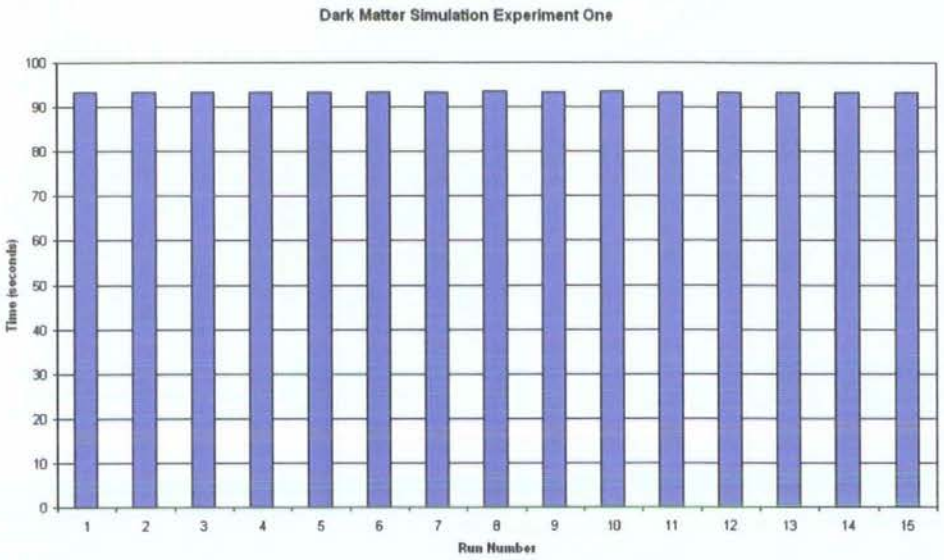


Figure 7.11: Chart of Dark Matter Experiment One Results

The mean of the data is 122.9 seconds and the standard deviation is 2.9

Experiment Six

Experiment Six had the same experimental set-up as Experiment One except that the machine was loaded to simulate heavy use by other users to determine the impact that this may have upon the execution of a pipeline.

The data for this experiment is shown in Table B.5 and in Figure 7.15 as a bar chart.

The mean of the data is 108.8 seconds and the standard deviation is 8.8

7.4 Summary

Scenarios and experimentation are both used to analyse the performance of the virtualised visualisation architecture this allows both qualative and quantative data to be collected, this data is analysed and evaluated in the next chapter. A summary of the results for the two experimental scenarios are shown in Table 7.1 and Table 7.2. This summary presents an average and standard deviation for each experiment within each scenario allowing an overview of the experimental results to be gained.

Experiment Number	Average Time(seconds)	Std. Deviation
1	19.82	1.11
2	20.69	0.18
3	20.34	0.19
4	20.03	0.18
5	20.69	0.16
6	41.89	77.14

Table 7.1: Overview of X-Ray Crystallography Experimental Result Data

Experiment Number	Average Time(seconds)	Std. Deviation
1	93.42	0.05
2	120.86	0.49
3	126.39	5.41
4	N/A	N/A
5	122.94	2.94
6	108.81	8.78

Table 7.2: Overview of the Dark Matter Case Study Experimental Results

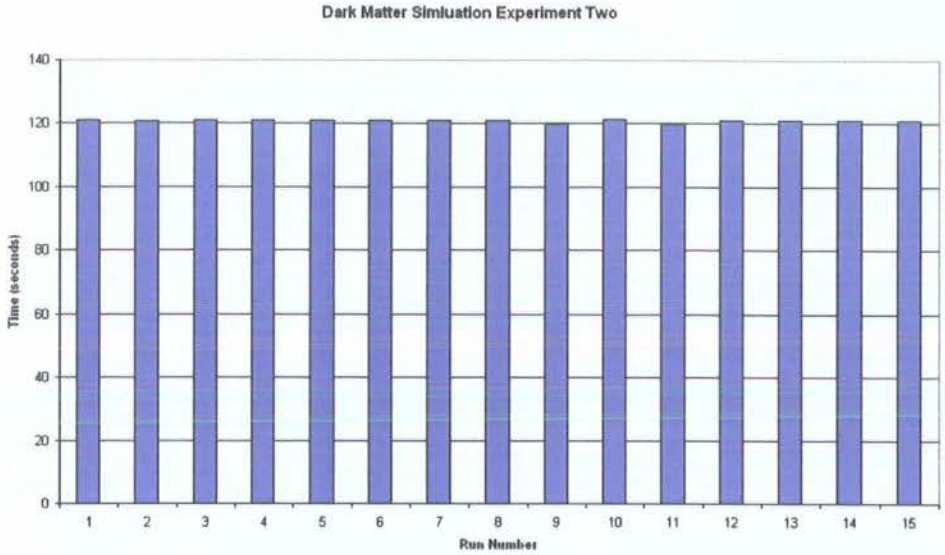


Figure 7.12: Chart of Dark Matter Experiment Two Results

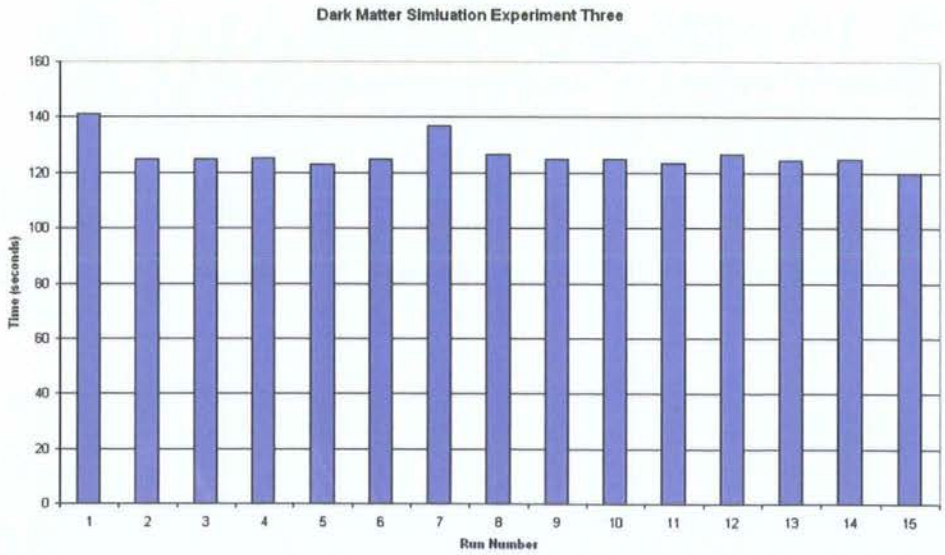


Figure 7.13: Chart of Dark Matter Experiment Three Results

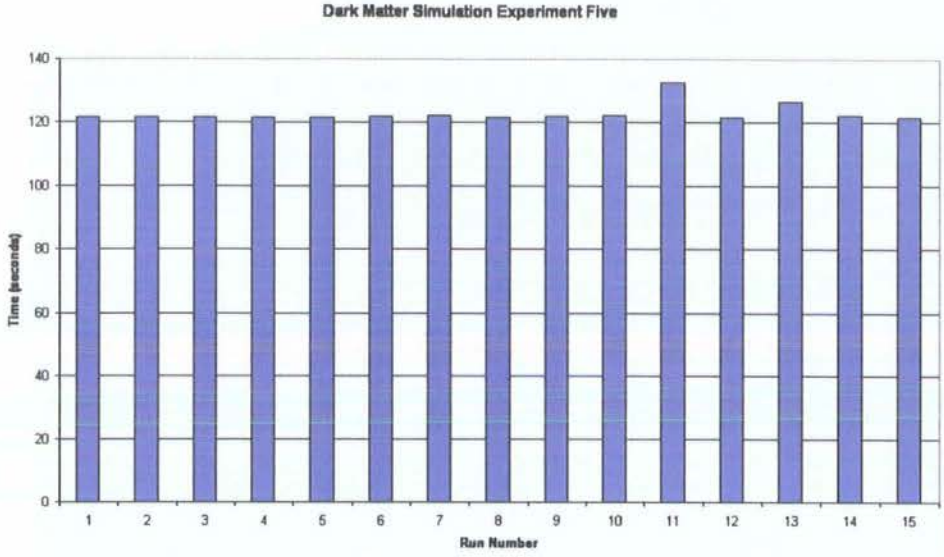


Figure 7.14: Chart of Dark Matter Experiment Five Results

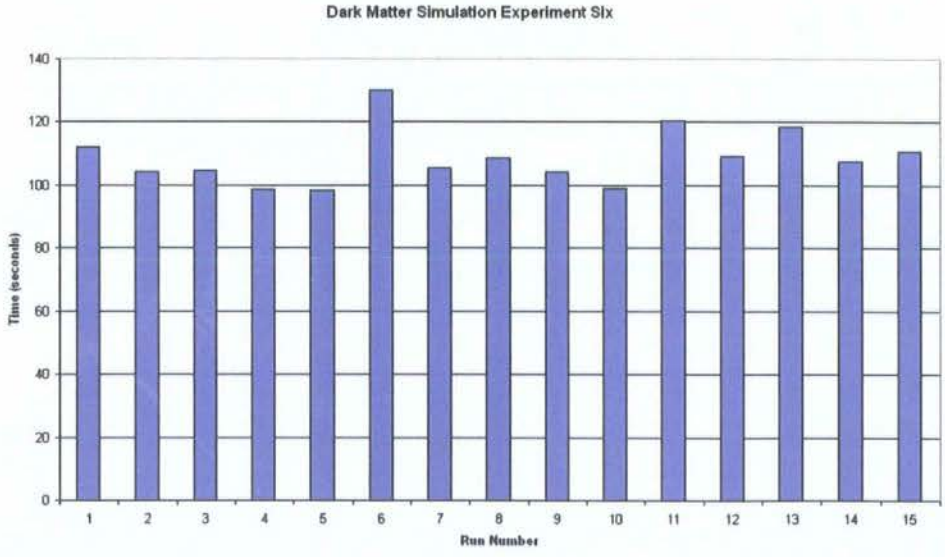


Figure 7.15: Chart of Dark Matter Experiment Six Results

Chapter 8

Evaluation

This chapter evaluates the virtualised visualisation architecture specified in Chapter 5. It does so by comparing the architecture against requirements set out at the beginning of Chapter 5. Two case study scenarios are used to evaluate the applicability of the architecture and evaluate the performance of the architecture using the experimental results from Chapter 7. The architecture is also assessed for its suitability for types of visualisation other than Scientific Visualisation through a worked example of Information Visualisation.

8.1 Requirements

The requirements for the implementation as stated in Chapter 5 are:

1. That the virtualised visualisation architecture can run on a variety of machine architectures.

2. That the visualisation can be displayed on a variety of different display platforms, including stereoscopic devices.

3. That the visualisation should be available on the desktop.

4. That the architecture will operate across multiple institutional boundaries, accounting for security issues.

Requirement	Architecture	Implementation	Dark Matter	X-Ray
1	Y	Y	Y	Y
2	Y	P	P	P
3	Y	Y	Y	Y
4	Y	P	P	P
5	Y	N	N/A	N/A
6	Y	Y	P	N/A
7	Y	P	P	P

Table 8.1: Summary of Evaluation

5. That the architecture will allow a number of scientists to collaborate from different locations, either synchronously or asynchronously.

6. The architecture should cope with very large remote data.

7. The architecture should allow visualisation to be performed interactively.

These requirements are used in the evaluation the architecture allowing the level to which they have been satisfied to be determined, a summary of the evaluation is presented in Table 8.1.

The table indicates the level to which the architectural design, the case studies and the implementation of the architecture meet the requirements for the architecture that have driven the research activity.

The following symbols are used, listed with their definition

Y - This requirement is met.

P - This requirement is partially met.

N - This requirement is not met.

N/A - This requirement does not apply in this case.

The table presents a summary of the evaluation. The architectural design is evaluated against meeting the requirements and shows that all of those requirements have been met by the design. The implementation is evaluated to show if the requirements have been implemented and the case studies evaluated against if the

requirement is demonstrated in that particular case study. The case studies were chosen to illustrate certain features of the architecture and the table shows the extent to which coverage was achieved. Not all features of the architecture were fully implemented and the table shows the extent to which each feature was achieved. This summary is discussed in more detail in the sections below.

8.2 Architectural Evaluation

This section examines in detail the design of the visualisation grid architecture against the requirements outlined previously.

1. *That the virtualised visualisation architecture can run on a variety of machine architectures.*

   The architecture is designed in a service oriented manner which provides a separation between the implementation that performs the computation at each stage of the visualisation pipeline and the communication between each stage of the pipeline. This separation means that each stage of the pipeline can run on a different resource, and those resources may have different underlying architectures. Platform agnostic programming languages such as Java can be used to implement each stage of the pipeline thus providing another layer of separation between machine architectures.

2. *That the visualisation can be displayed on a variety of different display platforms, including stereoscopic devices.*

   The design of the architecture splits the final stage of the traditional visualisation pipeline, render, into two parts, a render and a write service. This separation allows a range of display devices to be used with the architecture without the need to produce a large number of specialist rendering implementations for each display type. The render service can be developed as a generic service with the write service, a very lightweight implementation, customised where required for the different display types. This separation of render and write service also provides the possibility of using non-visual display devices

such as haptic interfaces and auralisation or sonification outputs.

3. *That the visualisation should be available on the desktop.*

   The render and write service separation also allow visualisation to be more effectively displayed upon desktop computers, this is particularly the case if the desktop computer has limited computational or graphical capabilities, the render service can render complex scenes and then send these to the desktop computer for display. More advanced implementations of the services could perform different parts of the rendering in order that the user had a more responsive and interactive experience when exploring the visualisation

4. *That the architecture will operate across multiple institutional boundaries, accounting for security issues.*

   The architecture is required to operate across multiple institutional boundaries, this is designed for by using standardised approach to the service oriented design making use of communication protocols which are internet standards. There are a wide range of differing security arrangements at different institutions and as such special configuration may be required for services to be deployed or to communicate however as far as possible the architecture has aimed to use standard mechanisms for communication to reduce the number of special arrangements required for such an architecture. The operation of the architecture across institution boundaries is often dependent on the implementation of the architecture, by using a service oriented architecture the ability to work across institutional boundaries is maximised but not guaranteed.

5. *That the architecture will allow a number of scientists to collaborate from different locations, either synchronously or asynchronously.*

   The architecture is designed such that a number of scientists can collaborate on the same pipeline, this is achieved by a single pipeline existing and a manage service being put in place to co-ordinate the scientists and control access to the pipeline for steering and configuration. The architecture can support multiple render services and write services so that different displays can be used by each

scientist and geographically local render services can be used by scientists who are geographically dispersed to improve performance.

6. *The architecture should cope with very large remote data.*

   The architecture is designed in such a manner that each service has a standard interface to the rest of the pipeline, this interface hides the implementation details. Thus the implementation can be tailored in a manner that best suits the resource upon which the service is deployed. The implementation can be one that is capable of scaling massively to cope with data sets of any size. Therefore the architecture is designed to cope with large remote data sets.

7. *The architecture should allow visualisation to be performed interactively.*

   Interactivity is an important part of visualisation and as such this has been designed into the architecture, the separation of the render and write services, allow interactivity at a image level and the provision of the manage service allows a higher level of interactivity by allowing users to exercise control over all parts of the visualisation pipeline.

The architecture is designed to be highly flexible to cater for all of the needs that visualisation users have and allowing support for novel forms of technology such as Autostereoscopic displays and haptic interfaces to be incorporated without a total re-implementation of significant parts of the pipeline.

8.3 Implementation

This section discusses the implementation of the visualisation grid architecture evaluating against the architectural requirements and the criteria for success.

1. *That the virtualised visualisation architecture can run on a variety of machine architectures.*

   The requirement for the architecture to run on a variety of different machine architectures has been met by making use of web service technology and the

Java programming language. For platforms that support the Java language no change to the implementation of services needs to be made.

For those platforms that do not support the Java language or where another implementation language is used for performance or integration reasons, the use of web service technologies means that data can be interchanged between different services and all services can be called by other services as they conform to a contract defined by the WSDL.

2. *That the visualisation can be displayed on a variety of different display platforms, including stereoscopic devices.*

   The architecture is designed such that each part of the pipeline is a separate service. The render service is separate to the service running the display, the result of this is that render services can be developed in such a way that they can cope with multiple display types and the service running on the display machine can be customised as required without having to re-implement the rendering code. The render service can therefore be deployed on a suitable resource, making use of scaling, distributed and parallel computing as appropriate and also maintain the ability to operate with a variety of display types. The implementation of this requirement was partially completed, a stereo render service was created by another project as reported in Chapter 6 but this has not been incorporated into the architecture to date. The render service created for the current implementation was able to perform some local stereo rendering using the stereo functionality in VTK.

3. *That the visualisation should be available on the desktop.*

   The requirement that visualisation should be available on the user's desktop is again met by the distributed nature of the grid visualisation architecture, by off loading the majority of the computational requirements from the desktop computer of the user the visualisation is not limited by the capabilities of that resource. This allows large data sets to be visualised using a relatively low capability computer to display the results.

4. *That the architecture will operate across multiple institutional boundaries, accounting for security issues.*

   The requirement that the architecture should operate between institutions whilst accounting for security issues, is mitigated as far as is possible by the use of web service technology that is based on common protocols. However due to the different security arrangements in place at different institutions and the ever changing nature of these security arrangements the architecture may not work seamlessly without configuration as to the institutional security arrangements and in some cases special provision for those resources that host services. The current implementation required special configuration to bypass a web cache at the network boundary this was implemented through the firewall at the campus network boundary.

5. *That the architecture will allow a number of scientists to collaborate from different locations, either synchronously or asynchronously.*

   This requirement is met in the design of the architecture through the use of the manage service which allows a pipeline to persist even when there are no users connected. The manage service acts as a user proxy in this instance, and the co-ordination functions of the service allow synchronous collaboration to take place. However the manage service has not been implemented therefore the implementation is unable to meet this requirement.

6. *The architecture should cope with very large remote data.*

   The current implementation does not work with very large data sets, the largest data set used currently is 40MB. However the service oriented nature of the architecture means that each service could be replaced with an extremely scalable version. To achieve this however the format of the data set used in the visualisation needs to be of a type where small segments of data can easily be extracted and where the whole data file is not required to be analysed to understand the data set.

7. *The architecture should allow visualisation to be performed interactively.*

The requirement that the architecture should allow visualisations to be performed interactively is met in part by the current implementation, as the visualisation can be manipulated locally due to the implementation of the viewer. However the pipeline does not support true interaction due to the one-shot nature of the current implementation. The implementation of a Manage service would allow a persistent pipeline and support for service steering and interaction to be implemented.

8.4 Case Study Scenario Evaluation

This section examines the scenarios taken from the science domain and implemented in the architecture to demonstrate the architecture in operation and provide a mechanism for evaluating the architecture and the implementation of the architecture. The architecture is flexible and non-prescriptive in how each of the services should be implemented internally.

8.4.1 X-Ray Crystallography

The X-Ray Crystallography case study is taken from the chemistry domain and demonstrates the pipeline required to produce a visualisation from a multi-purpose data set. The source data set for the visualisation contains two kinds of data, one describes the position of atoms within the crystal structure being examined and the second describes the values of the electric field surrounding the crystals as measured by experimentation. As the data set contains two types of data it is a good case study to demonstrate complex pipelines with multiple computational branches each producing a different part of the final visualisation.

The total size of the data set used in this case study is comparatively small and as such evaluation of the impact of critical paths through the visualisation has not been possible. However it allowed the demonstration of the pipeline working with data sets containing multiple types of data and pipelines producing two separate visualisation representations that are required to be merged to form the final visualisation.

The complexity of the pipeline allowed full coverage of all the experiments to determine the factors affecting the performance of the visualisation which would not have been possible with a less complex pipeline. The experiments and measurements of the architecture provided the data which could be used in the evaluation of the architecture against its requirements and criteria for success.

8.4.2 Dark Matter Simulation

The Dark Matter case study is taken from the cosmology domain in Physics. The visualisation for this case study is generated from a single large data set containing the position of dark matter particles as generated by a simulation of the distribution of dark matter since the big bang. The visualisation of this simulation shows how the dark matter is distributed through the universe at a specific point in time following the big bang.

The pipeline to generate this visualisation is simpler than the one required for the X-Ray Crystallography case study and contains a single branch. The aim of this pipeline is to demonstrate that the architecture is capable of handling large data sets through the services in the pipeline and allows experiments to be performed to determine the effects of distributing the pipeline across multiple resources. The use of large data sets allows the validation of requirement six in particular and the data from the experiments can be compared against the results from the X-Ray Crystallography case study. The experiments also aid in the evaluation of the architecture against the other requirements and also the criteria for success.

8.5 Performance Evaluation

This section discusses the results of the performance experiments run on the visualisation grid architecture. These experiments were designed to investigate the benefits achieved by splitting the visualisation pipeline into multiple separate services and distributing those services across different resources. The experiments are designed to examine different factors in the composition of pipelines, this allows the factors that are important when distributing pipelines to be determined.

Experiment Number	Average Time(seconds)	Std. Deviation
1	19.82	1.11
2	20.69	0.18
3	20.34	0.19
4	20.03	0.18
5	20.69	0.16
6	41.89	77.14

Table 8.2: Overview of X-Ray Crystallography Experimental Result Data

8.5.1 X-Ray Crystallography

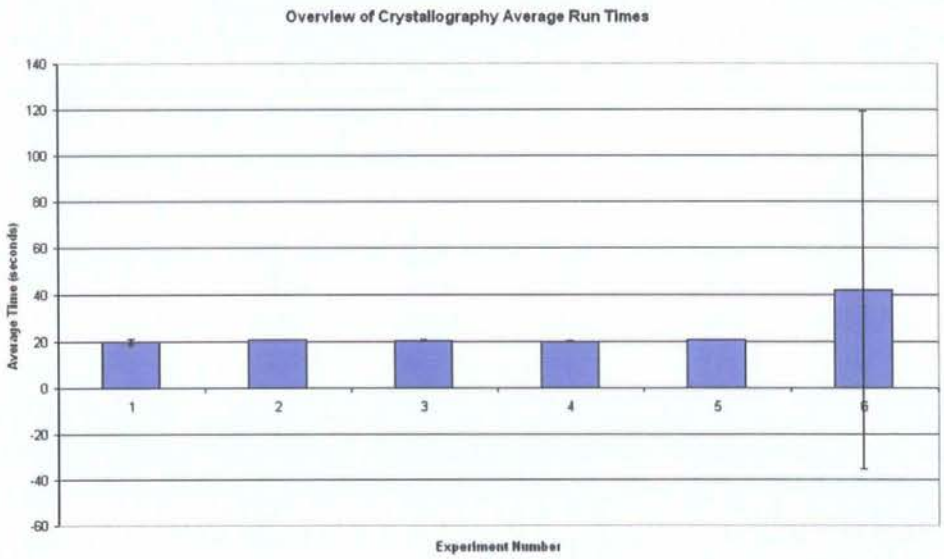


Figure 8.1: Overview of X-Ray Crystallography Experimental Results

The X-Ray Crystallography pipeline scenario allowed the evaluation of pipelines that have multiple branches to produce the final visualisation.

This pipeline has a small quantity of data transfer between computations, the result of this is that factors affecting the speed of computation have a much greater impact on the total running time of the pipeline than those that affect the transfer of data. This can be seen in Table 8.2 and Figure 8.1 where the average time for each experiment is around 20 seconds apart from Experiment Six where a heavily loaded resource is used. In this instance the average time doubles, however it is also

important to note that the standard deviation also increases dramatically. Indicating that the effects of loading on a resource is harder to predict and quantify. The outlying values in Experiment Six that cause the large variation in the standard deviation are to be expected for this type of experiment. With heavy CPU loading on a single CPU machine execution of processes maybe delayed whilst other intensive processes are executing, the time to completion of a processes is difficult to determine and therefore the variation shown in the results for Experiment Six not unexpected.

As the architecture is currently implemented the pipeline is run once only, in this way the result is displayed only once all computation is completed. In a multi branch pipeline the slowest branch of the pipeline therefore determines the total runtime. The slowest branch of the pipeline is therefore the 'critical path', efforts to speed the pipeline that do not aid this branch therefore are wasted. Techniques such as critical path analysis could therefore be used to determine which parts of the pipeline to invest extra resources in.

8.5.2 Dark Matter Simulation

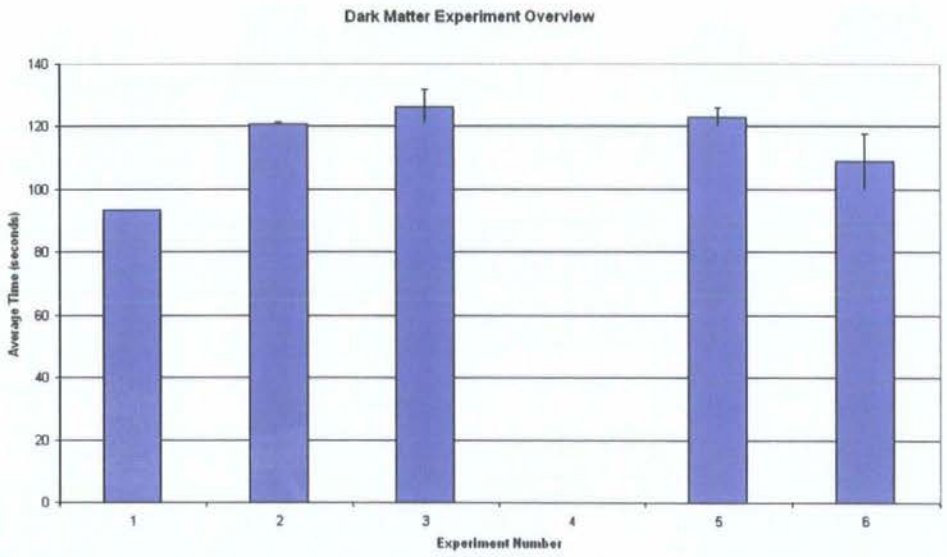


Figure 8.2: Overview of Dark Matter Experimental Results

The Dark Matter simulation allowed the evaluation of pipelines that have a larger

Experiment Number	Average Time(seconds)	Std. Deviation
1	93.42	0.05
2	120.86	0.49
3	126.39	5.41
4	N/A	N/A
5	122.94	2.94
6	108.81	8.78

Table 8.3: Overview of the Dark Matter Case Study Experimental Results

amount of data to be processed to produce a visualisation. The overview results for this case study are shown in Figure 8.2 and in Table 8.3.

This increase in data size means that factors that affect the data transfer have a bigger impact than those that affect the speed of computation, however a larger data set also requires more computation and therefore the difference in impact between the effect of network bandwidth and reduced computational ability is less.

The pipeline currently processes data sequentially and as such the size of the data set directly affects the speed of computation. Where data could be processed in a parallel manner the computation speed of the overall pipeline could be greatly improved.

The resource intensive experiment (Experiment Six) in this scenario had a lower overall runtime than some of the other experiments this is perhaps due to the amount disk access performed by the visualisation task. The intensive process run to load the processor and simulate a heavily utilised resource performed little I/O and could execute whilst the visualisation task was waiting for I/O operations to complete. The large data movement by the visualisation task altered the distribution of time spent on different tasks compared with the X-Ray crystallography experiment where most time was spent in CPU processing tasks. In the Dark Matter Simulation a large proportion on the time was spent moving data as such different resource subsystems are stressed and the profile of the experiments is slightly different.

The distribution of the pipeline causes a greater increase in the total runtime of the pipeline indicating that data transfer over a network is a major factor affecting

Experiment	X-Ray	Dark Matter
1	100%	100%
2	104%	129%
3	102%	135%
4	101%	N/A
5	104%	131%
6	211%	116%

Table 8.4: Percentage of Experiment One Comparison

this type of pipeline. The network distance however does not have a significant impact on the average runtime but the larger the network distance the larger the standard deviation indicating that it is less predictable as to its effects.

8.5.3 Generalizations and Overall Findings

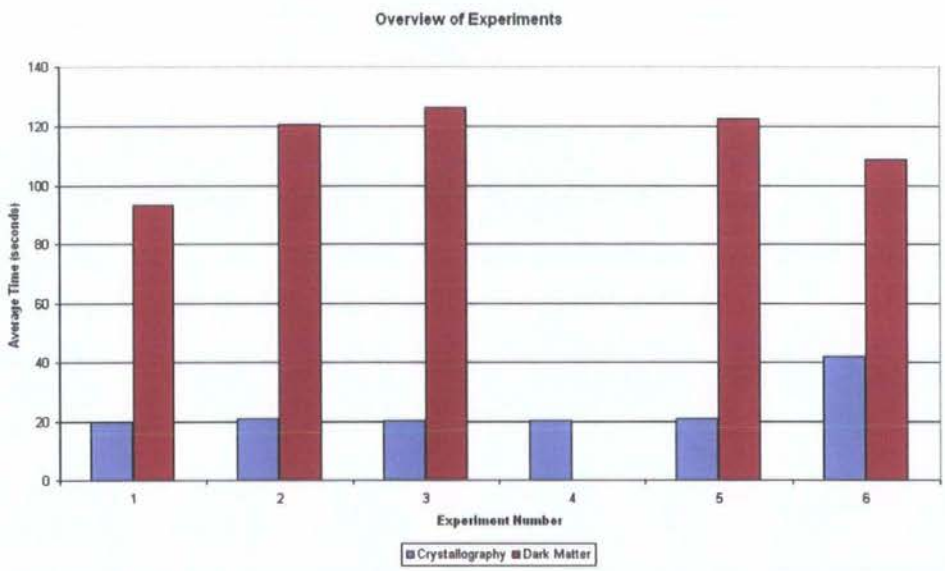


Figure 8.3: Overview of Experimental Results from both Scenarios

This section brings together the findings from the two scenarios for pipeline performance and highlights some common themes that affect the pipelines. Figure 8.3 presents the average run time for each experiment across both scenarios. Some limited comparisons can be drawn about the scenarios from this. Therefore Table 8.4

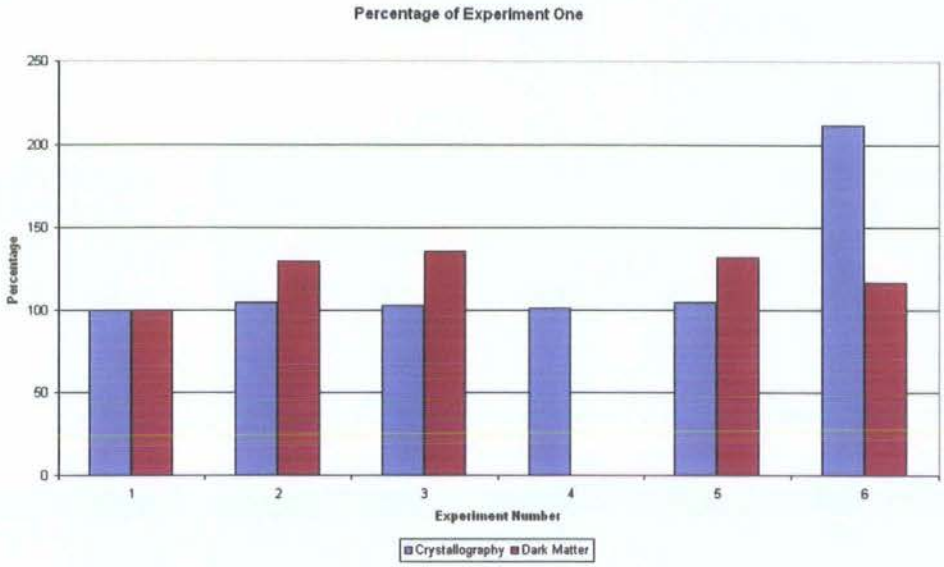


Figure 8.4: Overview of Percentage of Experiment One for both Scenarios

and Figure 8.4 show the consolidated experimental results for each experiment as a percentage of the base line experiment (experiment one). This allows for comparisons to be drawn between the two case studies.

Factors that impact the computation produce a more variable effect to the total runtime of the pipeline than those factors that impact the data transfer time. Therefore it may be easier to account for the impact of network connectivity, bandwidth and distance than machine loadings when selecting resources for use in a pipeline.

When looking to improve the visualisation pipeline it appears that distributing a large number of small computational jobs would be more successful than distributing a smaller number of large jobs. There will be a balance between the amount of data and computation and the network transmission times to be found. This balance it is felt would be best investigated through further experimentation.

This hypothesis is made due to the percentage changes across the experiments in the two case studies as shown in Table 8.4. In the case study that used a large data set a variation of 29% to 35% increase in the run time was observed across most experiments. If this variation is compared with that in the X-Ray Crystallography case study where a much smaller data set was used, it can be seen that the variation in that case study was a 1% to 4% increase in the total run time across experiments.

This smaller variation in completion time suggests that a large number of small jobs would have a more predictable and shorter computation time.

Comparison against Batch Processing Systems

If Experiment One is taken to be equivalent to operation in a batch queue system for application run time then a comparison can be performed with the execution of the architecture in its distributed configurations.

In a batch queue system jobs will remain in a queue until such times as all other jobs before them have been executed. Therefore if the computation is submitted and is the first job in the queue and the machine is currently idle the job can be executed immediately so the time to completion is the run time of the job. However if the queue is not empty the time to complete is:

$$W = CJT + \sum QJ + RT$$

where

W = Total time to completion.

CJT = Current Job Time to Completion.

QJ = Queued Jobs Run Time.

RT = Run Time of Visualisation Pipeline.

The time for the queued jobs to complete could vary from less than one minute to over several hours or longer, therefore the time spent waiting for execution could be significantly longer than the time to complete execution of the pipeline submitted.

In the service oriented architecture the execution time may be longer due to loading on machines and the time to transfer data across a network however execution is done on demand which reduces the number of resources that the scientist has to arrange time on, the 'co-allocation problem'. By distributing the computation of the pipeline across a variety of resources the effect of delays at a single resource can be minimised.

In the batch queue system once the execution of the pipeline is completed then the computation is discarded which requires a scientist to re-establish, re-configure and re-submit a pipeline to the queue if they wish to make any modifications. With

the service oriented architecture the pipelines can exist until the scientist has completed all of their analysis, this is termed pipeline persistence, this allows changes to the configuration to be made and parts of the pipeline re-executed as required.

The advantages of a service oriented architecture over a traditional batch queue architecture are:

- On demand Execution
- Re-configurable pipelines
- Scalability
- Interactive Visualisation
- Persistent Pipelines
- Reduce need for co-allocation of resources

These advantages mean that a service oriented architecture makes visualisation more usable as an everyday scientific visualisation tool over approaches which rely on batch queue systems and require advance co-allocation of resources.

8.6 Information Visualisation

The virtualised grid visualisation architecture has so far been shown to work for scientific visualisation, it can also be applied to other forms of visualisation such as information visualisation. The Component City visualisation is presented as an information visualisation that can be implemented using the distributed pipeline model of the architecture.

A visualisation previously developed is discussed and it is shown how this visualisation could be realised using the distributed architecture developed through this research.

The visualisation presented is the Component City Visualisation [Charters et al., 2002] which was developed as part of a project looking at software components. The project investigated the decision making process for selecting Software Components

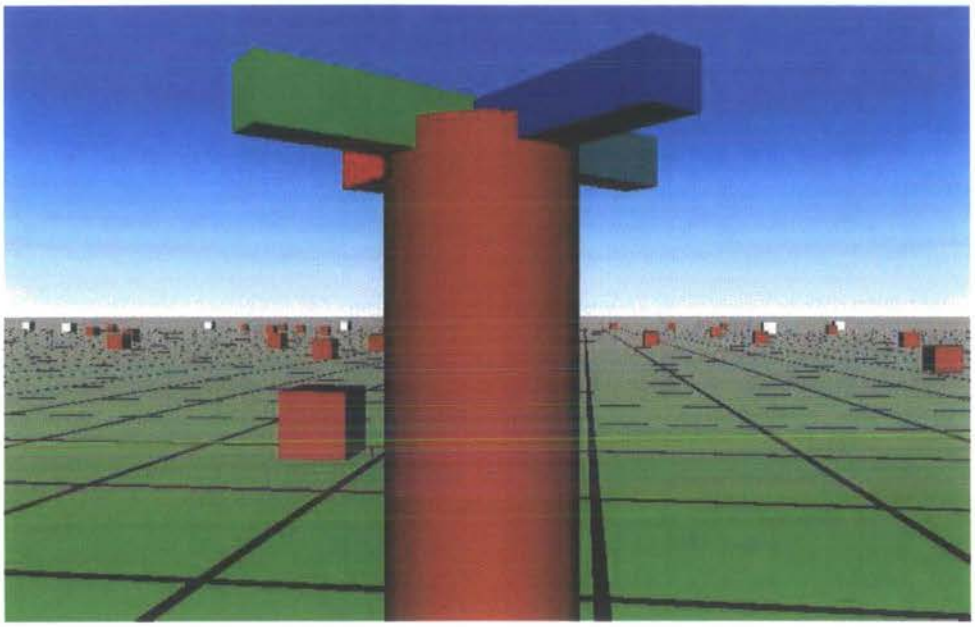


Figure 8.5: Component City Visualisation

and mechanisms to support that process. The visualisation developed in this project used concepts developed by Knight in Software World [Knight, 2000] to display groups of software components in a 3D landscape that represented a city, with buildings, roads and monuments.

The visualisation uses a self-organizing map to group an input set of components from a component repository each with a structured textual description. The components are grouped by the neural networks of the self-organizing map based on the description of their functional properties. The self-organizing map produces an output grid showing which components are functionally close together. This output is the converted into a visual representation which is based upon the idea of an american city grid layout, the roads run along the grid lines and the buildings along those roads represent the components. Different types of buildings are used to indicate the number of components at each location, the buildings in use are, house, mansion and skyscraper. To the roads and buildings, monuments are added, these monuments are added at the centre and corners of the city landscape to aid in navigation around the city. The output produced by the visualisation is shown in Figure 8.5 which depicts the central monument in the foreground with houses and

mansions behind and fading into the distance.

The visualisation was initially developed as a stand alone visualisation and the challenge is to see if this type of visualisation could be transferred to work with the visualisation architecture developed. Achieving this would demonstrate that the architecture can work for both scientific visualisation as shown by the earlier case studies and also information visualisation as shown here.

The visualisation differs from the two previous case studies as the input data set is a structured XML document which contains textual data describing the components. The output of the self-organizing map is also a structured textual file.

To implement the Component City visualisation in the visualisation architecture the stand alone program must be decomposed into stages that can then be implemented in the visualisation architecture. This can be achieved through an examination of the way the visualisation is generated. A component repository provides the data to the visualisation so this can be re factored as a read service, the data is manipulated by a self-organizing map, this can therefore be wrapped as a filter service in the architecture. The output of the self-organizing map is then used to create the description of the cityscape by inserting the correct type of building to represent the number of components at a particular location and attaching the correct labels this will form the map service, the output from this service can then be rendered using an existing render service and an existing write service can be used as a presentation client for the user to interact with the visualisation.

The pipeline formed for this visualisation can be seen in Figure 8.6, it can be seen that this pipeline is similar to the Dark Matter Simulation pipeline in that it is a simple pipeline with no branches.

The re-factoring of the Component City visualisation to use the distributed architecture would allow the visualisation to operate with more components in the input set as the self-organizing map could be run on a resource with large computational power and the rendering of the visualisation could be performed on a resource with a high graphical capability, the architecture would also allow multiple users to collaborate on the selection of components.

Figure 8.6: Component City Pipeline

8.7 Summary

This chapter has evaluated the results of both scenario based examination of the grid visualisation architecture and the numerical experimental results. The scenario evaluations highlights the flexibility of the architecture and the ability to deal with multiple visualisation types. The quantative numerically based experiments evaluate the performance of the architecture when pipelines enacting the scenarios have been executed. The meeting of the requirements by the architecture has also been evaluated.

Chapter 9

Conclusion and Future Work

9.1 Introduction

This chapter draws together the results and the evaluation of those results to present conclusions about the visualisation grid architecture and describes future work that builds upon the findings of the research.

The major achievements of the research have been:

- Definition of a Service Oriented Architecture for Visualisation on the Grid
- Implementation of the Service Oriented Architecture
- Real World Case Study Scenarios to demonstrate the architecture
- Performance Analysis of a Service Oriented Architecture
- Identification of areas for Future Work for both Visualisation and Visualisation Architectures

These achievements and others are discussed in the sections below.

This section outlines a review of the work done and then presents conclusions about the research undertaken.

Visualisation is a tool that has been used for centuries to communicate and to help with problem solving. Visualisation has evolved into a branch of computer science where the power of, and the graphical ability of, computers is exploited to produce the visualisations.

There exist a wide range of tools available to perform visualisation tasks, these tools can be general purpose or specialized and provide a variety of different mechanisms to produce visualisations.

One of the most common types of visualisation tool is the modular visualisation environment. It is this type of tool that has the most widespread general purpose use, these tools are often powerful but some deficiencies have been highlighted in their mechanisms for dealing with large data sets and for collaboration between multiple scientists.

The work to develop an architecture suitable for visualisation using the grid is motivated by the needs of scientists and other information workers to process and analyse increasingly large datasets routinely. This style of working is demonstrated by the scenario outlined in Chapter One and is becoming increasingly common as the use of computers becomes a necessity for scientific analysis.

The current state of the art in visualisation has been examined in Chapter Two along with an indepth look at several models for visualisation and several visualisation environments, these have been examined to identify weaknesses and strengths. These strengths have been built upon and applied to the visualisation architecture whilst the weaknesses have been addressed.

Stereoscopic display technology discussed in Chapter Three and the unique challenges and opportunities it presents was examined highlighting the importance of generating the correct stereo images for the display technology being used. The implementation of services to support stereoscopic displays is discussed in Chapter Six.

Grid technology in its various revisions was also extensively investigated in Chapter Four highlighting the current deficiencies and the ways in which the area is progressing.

Chapter Five of the thesis presented the conceptual design of the architecture, resulting in the model shown in Figure 9.1. This is a service based architecture that is designed to allow maximum flexibility in the implementation of visualisation pipelines, leveraging multiple resources each specialised for the task in hand to provide an efficient end to end pipeline that can cope with large data sets and

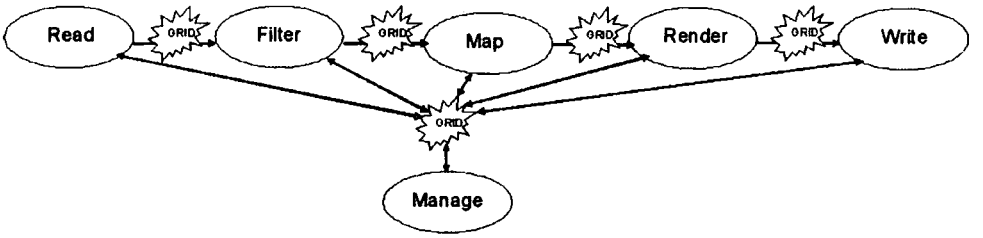


Figure 9.1: Final Pipeline Definition

interactive visualisation.

The implementation of the architecture was completed using the Java programming language making use of Web Service technology as outlined in Chapter Six. Three versions of the architecture were developed in an iterative manner each building upon the successes and learning from the failures of the previous version. These versions made use of grid technology as it evolved and was refined.

The grid technology as it currently exists is not ideal for the development of a grid visualisation architecture. It is still maturing and some of the underlying technology has not reached an exploitable stage. This lack of maturity required work arounds to be developed in order to achieve desired goals.

Changes in the underlying middleware, features that were not present or not fully functioning required the architecture to build in work around solutions to overcome the deficiencies. These work around solutions in many cases hampered the efficiency of the architecture, especially with regards to data transport and management.

Chapter Four in the review of Grid Technologies and Chapter Six describing the implementation of the architecture have highlighted some of the difficulties with grid systems. The changes in paradigm from batch computing, to stateful services and finally stateless services are each huge shifts in the way that systems are built. These changes impact upon the way architectures are thought about, designed and reasoned about.

In the grid visualisation architecture this has been particularly brought out with the change from stateful to stateless services. The initial reasoning about the architecture allowed for state to be held at each service and for pipelines to be composed of service instances. The move to stateless services meant that the service instances

were lost and the 'state for free' was also removed. This change necessitated the design of the architecture to include a service with the role of managing state across the entire pipeline. For services in the pipeline the state that was lost will have to be grafted back on through a mechanism that is non-standard. This use of non-standard state management would reduce the interoperability of services across different systems. The proposals of the addition of state to services is described in detail in Section 6.3.2 in Chapter Six.

Each grid middleware had its own learning curve, this curve was often steep with only partial transfer of knowledge gained with previous middlewares. The move between technologies in this way hampered the development of the architecture as each technology change required a return to the design and to restart the implementation from the ground up this is highlighted in extensive detail through Chapter Six in sections 6.1, 6.2 and, 6.3.

The multiple implementations of the grid visualisation architecture undertaken began to highlight ways in which the underlying grid middleware could potentially be put to use. Many of these uses not been considered in the design of the middleware or during the implementation of it. One area where this has been particularly apparent is that of data movement. Current data mechanisms were established for the movement of data files. Data files are static and unchanging, for an effective visualisation system the ability to transport data streams (data that is constantly being produced) would be required. No mechanisms to support this exist.

Two types of results about the grid visualisation architecture have been presented. The first type is the use of scenarios to show how the pipeline can be used to achieve different types of visualisation, both scientific visualisation and information visualisation. The second set of results are a quantative experimental analysis of the performance of the architecture as implemented using the scenarios that have been described previously these results show that the architecture meets in whole or in part all of the requirements and highlights areas for development to allow the achievement of that areas that are currently lacking.

9.2 Criteria for Success

The success of the research can be measured against the criteria for success which are restated here with an analysis of how they were achieved or otherwise, by the architecture.

1. *Allows a visualisation to be displayed on a desktop display.*

   The architecture has achieved this criteria through the decoupling of the write service from the render service which allows a scalable and powerful rendering solution to be used with a low powered display client as has been implemented in the current version of the architecture. This criteria for success is demonstrated through the two case study scenarios highlighted in Chapter 8 and through the design of the architecture in Chapter 5 with the separation between render services and write services.

2. *Runs across multiple computers/resources.*

   This criteria has been demonstrated by both scenarios and the experiments that have been to run to evaluate the architecture the results of these experiments are shown in Chapter 7. For example Experiment Four in the X-Ray Crystallography Scenario had services running on different resources for each branch of the pipeline. The results of these branches were then combined on another resource before being displayed. The design of the architecture in Chapter 5 highlights the reasoning behind the design decisions made to enable this to be achieved.

3. *Performs visualisation of a data set*

   This criteria is demonstrated by the output shown in Chapter 8 from the scenarios and the output results are reproduced in Figure 9.2 and Figure 9.3 for completeness. Figure 9.2 shows the X-Ray Crystallography Scenario with a the structure of a molecule from a crystal being displayed with an iso-surface representation of the Electric Field within the molecule superimposed on top. Figure 9.3 is produced from the Dark Matter Simulation scenario which looks at the growth of the universe and the spread of dark matter over time. The

visualisation shows a representation of the dark matter within the a region of the universe.

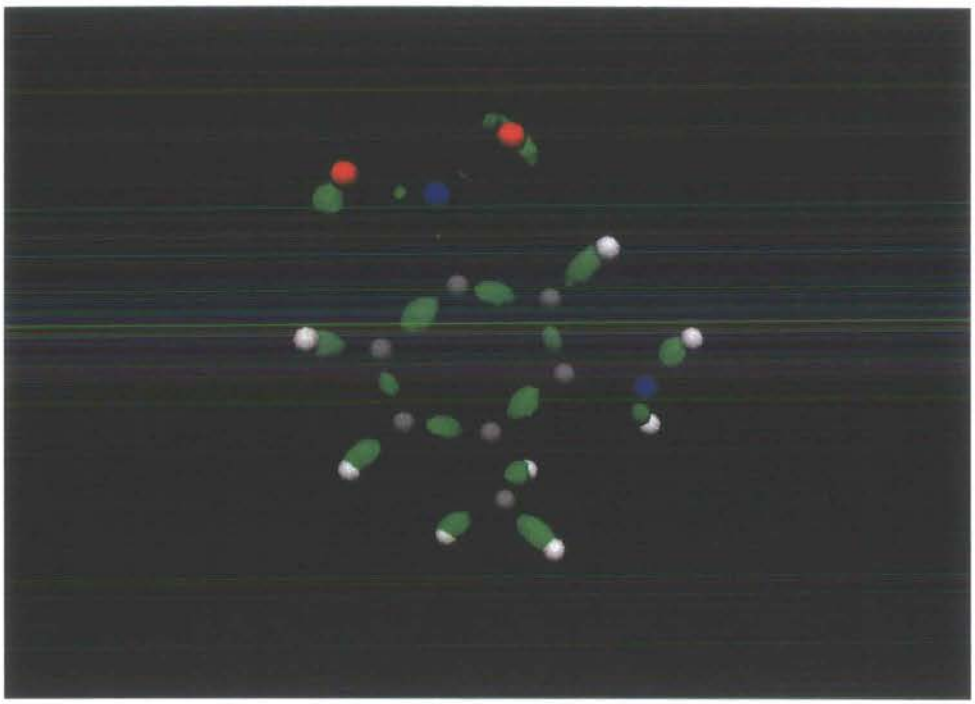


Figure 9.2: X-Ray Crystallography Scenario Result

4. *Supports interaction within the visualisation.*

   The architectural design is able to support interaction however the current implementation is only able to support interaction at the write service level. The implementation of the manage service in the architecture would allow interaction through the whole pipeline. The design of the architecture as discussed in Chapter 5 examines how the support of interaction with visualisation can be made possible with regards to remote interaction.

5. *Supports collaborative visualisation.*

   The architecture design supports collaboration allowing multiple write service from a render service, or multiple render services each with one or more write services. This has not currently been implemented. The implementation of the Manage service would help to support synchronisation across multiple views

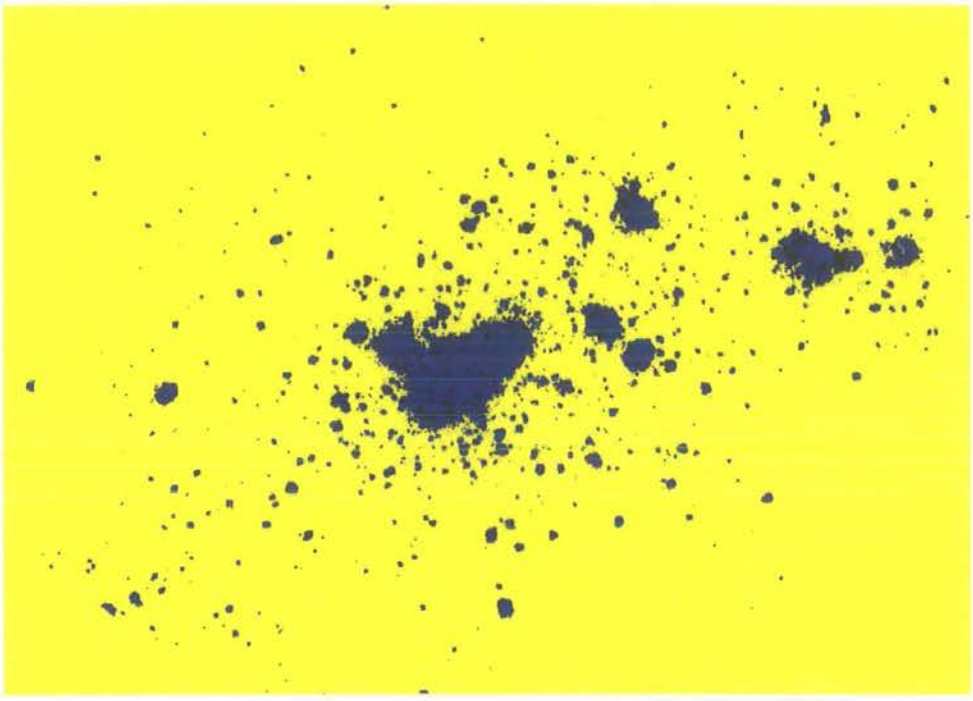


Figure 9.3: Dark Matter Simulation Pipeline Result

of the output. The design of the Manage service is discussed in Chapter 5 and this illustrates how collaborative visualisation can be supported.

6. *Supports multiple display types, including autostereoscopic desktop displays.*

   The architecture can support multiple display types and other visualisation mechanisms such as sonification and haptics through the use of services rendering and output of the visualisation. The stereo render service discussed in Chapter 6 demonstrates how the architecture can support autostereoscopic displays. The definition of write services in Chapter 5 explains how other types of output devices can be supported. The ability to support of multiple display types is a novel feature of the architecture developed. Current visualisation systems have limited support for stereo display devices as discussed in Chapter Three, by allowing the development of custom render services and output services a plethora of output devices can be supported. The support for these devices can be achieved without modification to other services in the pipeline and therefore provides great scope of maximum device support by all

visualisations.

7. *Supports multiple visualisation types*

   The architecture supports multiple visualisation types as demonstrated through the X-Ray Crystallography and Dark Matter Simulation scenarios and the Component City Information Visualisation scenario in Chapter 8. The X-Ray Crystallography and Dark Matter Simulation are both Scientific visualisations and reflect the scenario outlined in Chapter One describing the way in which scientists may work, and therefore the basis upon which the architecture was developed. The Component City visualisation demonstrates an alternative type of visualisation, an Information Visualisation. This type of visualisation makes use of the same stages in the visualisation pipeline but with very different data. Support for multiple visualisation types in important as multiple data sources are aggregated and as visualisation is used in other domains than numerical science.

9.3 Conclusion

The design of the architecture is such that it addresses many of the issues raised in the critical examination of existing visualisation systems

- Single Resource Bounded

  In many of the visualisation systems examined the software was executed on a single resource, the grid architecture by splitting the visualisation pipeline into its component parts allows the computation in the pipeline to be spread across multiple resources.

- Whole System Rebuild

  The visualisation environment SCIRun requires a rebuild of the entire system when it is extended or expanded, this does not encourage development of components for such a system by external developers, the grid architecture that has been developed solves this issue by encapsulating as services each stage of

the pipeline and providing a common set of interfaces to allow developers to develop new services that can be easily integrated into the pipeline.

- Client Centred

  Current modular visualisation environments are client centred in that the tool (client) used to compose the pipeline is the container within which the pipeline is executed. The result of this is that the client needs to be run on the resource where the pipeline is to be executed and that the client software must be running at all times whilst the visualisation pipeline is executing.

  The grid architecture by using services for each stage of the pipeline and a visualisation management service to control the pipeline overall can allow execution on any resource and supports pipeline composition using a tool that can be disengaged once the pipeline construction is completed. This means that a pipeline can be constructed on any resource, for example, the computer that the user has in their normal working environment.

- Generalised Resources

  Currently visualisation systems are run on a single resource, this resource must be a generalised resource with a suitable processor, memory and graphics capability to handle all operations in the visualisation pipeline. In the grid architecture as each part of the pipeline is encapsulated as a separate service and can be deployed on a different resource the resource can be tailored to the needs of that service this deploying of services on suitable resources can allow more effective use to be made of specialised resources and reducing the need for them to do general computing.

9.4 Future Work

Many areas of further work have been identified by the development of the architecture for visualisation.

These areas can be categorised into work that completes the architecture, fundamental research, both for grid middleware and visualisation, and more advanced

research that builds on the concepts developed so far.

9.4.1 Architecture Developments

This section outlines areas of research that build upon and extend the current architecture to make it more robust and comprehensive.

Service Development

The development of extra services that can be used in the architecture will allow it to be applied to a more diverse set of problems that will further stretch and test the architecture.

Collaboration Support

One of the aims of the architecture was that of collaboration between scientists, further developments to allow this particularly in conjunction with the development of context management within services which would allow greater flexibility in the visualisations developed.

Context Management

Context management is both a development that builds directly into the architecture as it stands and a more advanced area of development for the architecture and is therefore discussed more fully later in this chapter.

9.4.2 Fundamental Grid Research

This section outlines what I regard as important issues to address for grid middleware to allow service based applications such as the virtualised visualisation architecture to be developed.

High Level Reliable Data Transports

High level data transports means providing a mechanism for grid services to return data to other services without having to concern itself with the underlying transport.

It also encompasses allowing a service to read data without having to worry about the underlying transport.

This type of service should support blocking and multiple destinations for the data, either independently or collectively.

One development that shows promise in this area is the Styx data transport mechanism [Blower et al., 2005] which simulates files on the local and remote hosts and can be used with a variety of web service transport mechanisms.

Context and State Management

Web services are stateless and many existing applications are written for a single user to execute at a time, and as such don't concern themselves with state management.

A context management service would allow services to be written with concerning themselves with how to manage multiple users. This would also aid in leveraging existing code, for example modules from existing MVE systems.

Various approaches to managing state in web services including session identifiers, tokens and identifiers in URLs have been proposed by none have been standardised.

9.4.3 Fundamental Visualisation Research

This section looks at research that is vital to the development of visualisation or visualisation services.

Standard Steering Library

As simulations and visualisations become deployed widely and increasingly used in collaborative manner with international multidisciplinary teams the needs for a standard format for steering simulations and controlling visualisation services will be required.

Current steering libraries include those developed by the RealityGrid team [Coveney et al., 2005] and also that developed by the gViz team [Aslanidi et al., 2005] these libraries could be used as the basis for the development of a standard visualisation steering library.

Visualisation Ontology

The development of a visualisation ontology would be an advance that would allow visualisation services to be described and searched for when deployed across the grid.

Access Grid Integration

A mechanism for collaboration that is becoming widely deployed in academic institutions is the Access Grid. To allow visualisations to be brought into this collaborative environment would stimulate the use of visualisation in same time, different place collaborations.

Initial work on using the Access Grid with visualisation has been done by the ICENI project [Kong et al., 2003] which illustrates how such integration may work.

Standard Visualisation Data Formats

The current range of visualisation system have a wide range of data formats for input and for data management between sections of the pipeline. To encourage interoperability and reduce time consuming and processor intensive data format conversions a standard range of formats for visualisation would be of great assistance.

Automatically Scaling Services

To take full advantage of grid technology and to allow services to deal with increasingly large data sets, research into services that can scale themselves adaptively should be undertaken.

9.4.4 Advanced Research

This section outlines research that would build on the research that has already been undertaken.

Automatically Composed Pipelines

End users do not want to be concerned with building visualisation pipelines, to this end a mechanism that would allow end users to specify the visualisation they require and for it to be automatically composed from the available services.

Curation and Provenance

Scientific experiments must be repeatable, at present their are no facilities for preserving a visualisation in its entirety however it is done, the only storage mechanism is the recording of a visualisation output in a video format. This however does not store the method through which the visualisation was achieved, or any of the associated data with the visualisation and any collaboration surrounding it.

The UK has established a Digital Curation Centre which is looking generally at the issues surrounding Digital Curation. Work on Provenance in e-Science has been investigated by the myGrid project [Zhao et al., 2003] among others and has highlighted some of the issues that need to be considered by scientists who wish to include provenance information into visualisation pipelines.

9.5 Summary

In this chapter conclusions about the research that have been undertaken are presented and directions for future research have been outlined. Current visualisation systems have many deficiencies, these deficiencies in the main stem from the fact that they were developed before the era of grid computing and before scientists had come to value the role visualisation can take in the analysis of large data sets.

This research defines a new architecture centered around a the service based architecture paradigm which addresses the shortcomings identified with existing approaches to visualisation systems and provides a mechanism by which much of the technical complexity of grid computing can be hidden behind services.

The architecture is designed to exploit highly heterogeneous environments making use of multiple resources rather than a single supercomputer or HPC cluster. The architecture provides the ability for scientists to establish pipelines that persist

without concerning themselves with the co-allocation of resources to achieve their analysis goals. The architecture allows the scientist to bring the visualisation to their desktop rather than take themselves to the visualisation suite which then provides the opportunity for visualisation to be come an integral part of the scientists normal working practises when it comes to data analysis rather than the icing on the cake.

The architecture defined is highly flexible and shows great promise for e-Science visualisation of very large data sets.

Bibliography

Globus Alliance. Globus Toolkit 2.4. http://www.globus.org/gt2.4/, 2004.

L. M. Applegate. Technology support for cooperative work: A framework for studying introduction and assimilation in organizations. *Journal Organizational Computing*, pages 11–39, 1991.

OV Aslanidi, KW Brodlie, RH Clayton, JW Handley, AV Holden, and J Wood. Remote visualization and computational steering of cardiac virtual tissues using gviz. In *Proceedings of UK e-Science All Hands Meeting 2005*, 2005.

Malcolm Atkinson, David DeRoure, Alistair Dunlop, Geoffrey Fox, Peter Henderson, Tony Hey, Norman Paton, Steven Newhouse, Savas Parastatidis, Anne Trefethen, Paul Watson, and Jim Webber. Web service grids: An evolutionary approach. Technical Report UKeS-2004-05, UK National e-Science Centre, http://www.nesc.ac.uk/technical_papers/UKeS-2004-05.pdf, August 2004.

T. Banks, A. Djaoui, S. Parastatids, A. Mani, S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D. Snelling, and P. Vanderbilt. Open grid service infrastructure primer. Technical Report GFD.31, Global Grid Forum, http://www.ggf.org/documents/GFD.31.pdf, 2004.

Tim Beardsley. Humans Unite! *Scientific American*, March 1999.

W. Bethel, R. Frank, S. Fulcomer, C. Hansen, K. Joy, J. Kohl, and D. Middleton. Visual data analysis - report of the visualization breakout session. In *2003 SCaLeS Workshop - Volume II*, 2003.

J. Blower, K. Haines, and E. Llewellin. Data streaming, workflow and firewall-friendly grid services with styx. In S.J. Cox and D.W. Walker, editors, *Proceedings of the UK e-Science All Hands Meeting 2005*, 2005.

K. Brodlie, J. Brooke, M. Chen, D. Chisnall, A. Fewings, C. Hughes, N. W. John, M. W. Jones, M. Riding, and N. Roard. Visual Supercomputing - Technologies, Applications and Challenges, STAR Report. In *Procedings of Eurographics 2004*. Eurographics Association, 2004a.

K. W. Brodlie, D. A. Duce, J.R. Gallop, J. P. R. B. Walton, and J. D. Wood. Distributed and collaborative visualization. *Computer Graphics Forum*, 23(2): 223–251, 2004b.

Stuart K. Card, Jock D. MacKinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc, San Francisco, California, 1999. ISBN: 1558605339.

S. M. Charters, C. Knight, N. Thomas, and M. Munro. Visualisation for Informed Decision Making; From Code to Components. In *Proceedings of the Workshop on Software Engineering Decision Support, 14th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 765–772. ACM Press, July 2002.

climateprediction.net. climateprediction.net. http://climateprediction.net/, 2004.

P.V. Coveney, G. De Fabritiis, M.J. Harvey, S.M. Pickles, and A.R. Porter. On steering coupled models. In *Proceedings of the UK e-Science All Hands Meeting 2005*, 2005.

Roger Crawfis, Nelson Max, Barry Becker, and Brian Cabral. Volume rendering of 3d scalar and vector fields at llnl. In *Supercomputing '93*, 1993.

K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke. From open grid services infrastructure to ws-resource framework: Refactoring & evolution. http://www-106.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf, March 2004.

D. A. Duce, D. Giorgetti, C. S. Cooper, J. R. Gallop, I. J. Johnson, and C. D. Seelig. Reference Models for Distributed Cooperative Visualization. *Computer Graphics Forum*, 17(4):pp. 219–233, 1998.

e-Viz. e-Viz Project. http://www.eviz.org/, 2004.

I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Von Reich. The open grid services architecture, version 1.0. Technical Report GFD.30, Global Grid Forum, 2004.

Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

D. Foulser. Iris explorer: A framework for investigation. *Computer Graphics*, 29(2): 13–16, 1995.

Al Globus and Eric Raible. Fourteen ways to say nothing with scientific visualization. *IEEE Computer*, pages 86–88, July 1994.

Martin Gudgin and Timothy Ewald. Pork Barrel Protocols. http://webservices.xml.com/pub/a/ws/2001/09/12/porkbarrel.html.

gViz. gViz project website. Available at http://www.visualization.leeds.ac.uk/gViz/, 2004.

R. B. Haber and D. A. McNabb. *Visualization In Scientific Computing*, chapter Visualization Idioms: A conceptual model for scientific visualization systems, pages 74–93. IEEE Computer Society Press, 1990.

Bernd Hamann, E. Wes Bethel, Horst Simon, and Juan Meza. NERSC "VISUALIZATION GREENBOOK" FUTURE VISUALIZATION NEEDS OF THE DOE COMPUTATIONAL SCIENCE COMMUNITY HOSTED AT NERSC. *The International Journal of High Performance Computing Applications*, 17(2):pp. 97–123, Summer 2003.

R. W. Hamming. *Numerical Methods for Scientists and Engineers.* McGraw-Hill, 1962.

Pat Hanrahan. The future of computer graphics: Realism or abstraction. In *IS&T/SPIE 17th Annual Symposium Electronic Imaging Science & Technology*, 2005.

Bill Hibbard. VisAD. http://www.ssec.wisc.edu/ billh/visad.html.

N. S. Holliman. *Handbook of Optoelectronics*, chapter 3D Display Systems. The Institute of Physics, November 2004. ISBN 0750306467.

Kitware. Paraview. http://www.paraview.org/.

Claire Knight. *Virtual Software in Reality.* PhD thesis, Department of Computer Science, University of Durham, June 2000.

Gary Kong, Jim Stanton, Steven Newhouse, and John Darlington. Collaborative visualisation over the access grid using the iceni grid middleware. In *Proceedings of the UK e-Science All Hands Meeting 2003*, 2003.

M. Leigh. Human visual system. http://www.cs.unm.edu/ leigh/pix/.

Neil Lock. Stereo render service. Master's thesis, University of Durham, 2003.

C. Michaels and M. Bailey. Vizwiz: a java applet for interactive 3d scientific visualization on the web. In *Proceedings of Eighth IEEE Visualisation 1997*, page 261, 1997.

NASA Goddard Space Flight Center. Scientfic Visualization Studio. http://svs.gsfc.nasa.gov/.

NERSC. FY2002 User Survey Results - Visualization and Grid Computing. http://www.nersc.gov/news/survey/2002/viz.html, 2002.

Ocuity. Ocuity reconfigurable 2d/3d displays. http://www.ocuity.co.uk/Ocuity_2D-3D_display_brochure.pdf, 2005.

S. G. Parker and Chris Johnson. SCIRun: A scientific programming environment for computational steering. In H. W. Meuer, editor, *Proceedings of Supercomputing '95, New York*. Springer-Verlag, Berlin, 1995.

RAVE. Rave. http://www.wesc.ac.uk/projects/rave/, 2004.

Royal Phillips Electronics. Philips 3d multiview lenticular display technology. http://www.business-sites.philips.com/3dsolutions/technology/index.html, 2005.

Lakshmi Sastry and Martin Craig. Scalable application visualisation services toolkit for problem solving environments. In *Proceedings of the UK e-Science All Hands Meeting 2003*, 2003.

W. J. Schroeder, K. M. Martin, and W. E. Lorensen. *The Visualization Toolkit: An object oriented approach to 3D graphics*. Kitware Inc., third edition edition, 2003.

John Shalf and E. Wes. Bethel. How the Grid will affect the Architecture of Future Visualization Systems. Technical Report LBNL-51723, Lawrence Berkeley National Laboratory.

Larry Smarr and Charles E. Catlett. Metacomputing. *Communications of the ACM*, 35(6):45–52, June 1992.

Ian Taylor, Matthew Shields, Ian Wang, and Roger Philp. Distributed P2P computing with triana: A galaxy visualization test case. In *Procceedings of 17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, pages 16–27. IEEE Computer Society, 2003.

H. Theisel and M. Kreuseler. An enhanced spring model for information visualization. *Computer Graphics Forum*, 17(3):335–344, 1998. http://www.mpi-sb.mpg.de/ theisel/gallery/enhanced_springmodel/enhanced_springmodel.html.

D. Trayner and E. Orr. Autostereoscopic display using holographic optical elements. In *Stereoscopic Displays and Virtual Reality Systems III*, 1996.

Yunsong Wang, Gordon Erlebacher, Zachary A. Garbow, and David A. Yuen. Web-Based Service of a Visualization Package "Amira" in the Geosciences. *Visual Geosciences*, 2003.

Colin Ware. *Information Visualization: Perception for Design.* Morgan Kaufmann, 2000. ISBN: 1558605118.

Andreas Wierse. Performance of the collaborative visualization environment (covise) visualization system under different conditions. In Georges G. Grinstein and Robert F. Erbacher, editors, *Proceedings of SPIE Visual Data Exploration and Analysis II*, volume 2410, pages 218–229, 1995.

J. D. Wood, H. Wright, and Ken W. Brodlie. Collaborative Visualization. In R. Yagel and H. Hagen, editors, *IEEE Visualization '97*, pages 253–259, 1997.

Jun Zhao, Carole Goble, Mark Greenwood, Chris Wroe, and Robert Stevens. Annotating, linking and browsing provenance logs for e-science. In *Proceedings of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

Appendix A

X-Ray Crystallography Data

Run	Start	Stop	Total - milliseconds
1	1096293962096	1096293982095	19999 (19.999 sec)
2	1096294021452	1096294041360	19908 (19.908 sec)
3	1096294674220	1096294694089	19869 (19.869 sec)
4	1096294718271	1096294734116	15845 (15.845 sec)
5	1096294748327	1096294768416	20089 (20.089 sec)
6	1096294801573	1096294821973	20400 (20.4 sec)
7	1096294842633	1096294862792	20159 (20.159 sec)
8	1096294902739	1096294923118	20379 (20.379 sec)
9	1096294938350	1096294958479	20129 (20.129 sec)
10	1096294978137	1096294998296	20159 (20.159 sec)
11	1108726373947	1108726394247	20300 (20.3 sec)
12	1108726412112	1108726510644	19899 (19.899 sec)
13	1108726530452	1108726510644	20279 (20.279 sec)
14	1108726530452	1108726550451	19999 (19.999 sec)
15	1108726567666	1108726587545	19879 (19.879 sec)

Table A.1: Experiment One: Single Machine

Run	Start	Stop	Total - milliseconds
1	1113556984689	1113557005179	20490 (20.490 sec)
2	1113557766323	1113557787243	20920 (20.92 sec)
3	1113557836164	1113557856803	20639 (20.639 sec)
4	1113557926534	1113557947043	20509 (20.509 sec)
5	1113558046236	1113558067146	20910 (20.91 sec)
6	1113558098711	1113558119491	20780 (20.78 sec)
7	1113558146119	1113558166619	20500 (20.5 sec)
8	1113558193197	1113558214097	20900 (20.9 sec)
9	1113558237360	1113558258060	20700 (20.7 sec)
10	1113558281764	1113558302214	20450 (20.45 sec)
11	1113558322222	1113558342712	20490 (20.49 sec)
12	1113558370232	1113558391212	20980 (20.98 sec)
13	1113563672386	1113563693115	20729 (20.729 sec)
14	1113563720915	1113563741575	20660 (20.66 sec)
15	1113563764939	1113563785598	20659 (20.659 sec)

Table A.2: Experiment Two: Locally Remote Data

Run	Start	Stop	Total - milliseconds
1	1118997345256	1118997365625	20369 (20.369 sec)
2	1118997453271	1118997473520	20249 (20.249 sec)
3	1118997498196	1118997518535	20339 (20.339 sec)
4	1118997757378	1118997777547	20169 (20.169 sec)
5	1118997886975	1118997907164	20189 (20.189 sec)
6	1118998071370	1118998091659	20289 (20.289 sec)
7	1118998151755	1118998172585	20830 (20.83 sec)
8	1118998195568	1118998215767	20199 (20.199 sec)
9	1118998246492	1118998266631	20139 (20.139 sec)
10	1118998288752	1118998308871	20119 (20.119 sec)
11	1118998527826	1118998548266	20440 (20.44 sec)
12	1118998577498	1118998597787	20289 (20.289 sec)
13	1118998646757	1118998667177	20420 (20.42 sec)
14	1118998743927	1118998764376	20449 (20.449 sec)
15	1118999016619	1118999037219	20600 (20.6 sec)

Table A.3: Experiment Three: Remote Data

Run	Start	Stop	Total - milliseconds
1	1112956593543	1112956613572	20029 (20.029 sec)
2	1112956678395	1112956698464	20069 (20.069 sec)
3	1112956720966	1112956740885	19919 (19.919 sec)
4	1112956765951	1112956786080	20129 (20.129 sec)
5	1112956953521	1112956974110	20589 (20.589 sec)
6	1112957006036	1112957026145	20109 (20.109 sec)
7	1112957050460	1112957070449	19989 (19.989 sec)
8	1112957195949	1112957215878	19929 (19.929 sec)
9	1112957235436	1112957255335	19899 (19.899 sec)
10	1112957278398	1112957298317	19919 (19.919 sec)
11	1112957318005	1112957338124	20119 (20.119 sec)
12	1112957363861	1112957383769	19908 (19.908 sec)
13	1112957425229	1112957445208	19979 (19.979 sec)
14	1112957470584	1112957490483	19899 (19.899 sec)
15	1112957510361	1112957530260	19899 (19.899 sec)

Table A.4: Experiment Four: Split Pipeline

Run	Start	Stop	Total - milliseconds
1	1119279424722	1119279445311	20589 (20.589 sec)
2	1119279477608	1119279498277	20669 (20.669 sec)
3	1119279539587	1119279560176	20589 (20.589 sec)
4	1119279680379	1119279701029	20650 (20.65 sec)
5	1119279847700	1119279868750	21050 (21.05 sec)
6	1119279891593	1119279912213	20620 (20.62 sec)
7	1119279931440	1119279951980	20540 (20.54 sec)
8	1119279969054	1119279989664	20610 (20.61 sec)
9	1119280010364	1119280031434	21070 (21.07 sec)
10	1119280052614	1119280073364	20750 (20.75 sec)
11	1119280119491	1119280140130	20639 (20.639 sec)
12	1119280390460	1119280411120	20660 (20.66 sec)
13	1119280431339	1119280451979	20640 (20.64 sec)
14	1119280471557	1119280492197	20640 (20.64 sec)
15	1119280511104	1119280531803	20699 (20.699 sec)

Table A.5: Experiment Five: Services on Different Machines

Run	Start	Stop	Total - milliseconds
1	1113825655850	1113825698511	42661 (42.661 sec)
2	1113826258837	1113826278895	20058 (20.058 sec)
3	1113826460787	1113826480756	19969 (19.969 sec)
4	1113827065286	1113827085225	19939 (19.939 sec)
5	1113827148666	1113827468595	319929 (319.929 sec)
6	1113827212298	1113827237374	25076 (25.076 sec)
7	1113827273095	1113827292994	19899 (19.899 sec)
8	1113827726687	1113827746646	19959 (19.959 sec)
9	1113828065895	1113828085854	19959 (19.959 sec)
10	1113828141524	1113828161783	20259 (20.259 sec)
11	1113828738963	1113828758972	20009 (20.009 sec)
12	1113828796496	1113828816815	20319 (20.319 sec)
13	1113828857353	1113828877342	19989 (19.989 sec)
14	1113828916268	1113828936467	20199 (20.199 sec)
15	1113829083068	1113829103126	20058 (20.058 sec)

Table A.6: Experiment Six: Loaded Machine

Appendix B

Dark Matter Simulation Data

Run	Start	Stop	Total - milliseconds
1	1114683019153	1114683112597	93444 (93.444 sec)
2	1114683562053	1114683655488	93435 (93.435 sec)
3	1114684645481	1114684738855	93374 (93.374 sec)
4	1114684854041	1114684947455	93414 (93.414 sec)
5	1114685293843	1114685387248	93405 (93.405 sec)
6	1114685754396	1114685847780	93384 (93.384 sec)
7	1114686026707	1114686120082	93375 (93.375 sec)
8	1114686396058	1114686489613	93555 (93.555 sec)
9	1114686821440	1114686914824	93384 (93.384 sec)
10	1114689195914	1114689289439	93525 (93.525 sec)
11	1114690138640	1114690232074	93434 (93.434 sec)
12	1114690300713	1114690394107	93394 (93.394 sec)
13	1114691018185	1114691111579	93394 (93.394 sec)
14	1114691425621	1114691519005	93384 (93.384 sec)
15	1114692418558	1114692511993	93435 (93.435 sec)

Table B.1: Experiment One: Dark Matter

Run	Start	Stop	Total - milliseconds
1	1114697306377	1114697427481	121104 (121.104 sec)
2	1114697746349	1114697867253	120904 (120.904 sec)
3	1114697914080	1114698035084	121004 (121.004 sec)
4	1114698110663	1114698231677	121014 (121.014 sec)
5	1114698463821	1114698584795	120974 (120.974 sec)
6	1114699069492	1114699190496	121004 (121.004 sec)
7	1114699458221	1114699579295	121074 (121.074 sec)
8	1114699706808	1114699827792	120984 (120.984 sec)
9	1114699906555	1114700026248	119693 (119.693 sec)
10	1114700308103	1114700429427	121324 (121.324 sec)
11	1114700505096	1114700624788	119692 (119.692 sec)
12	1114700787232	1114700908246	121014 (121.014 sec)
13	1114701185384	1114701306378	120994 (120.994 sec)
14	1114701528227	1114701649412	121185 (121.185 sec)
15	1114701701406	1114701822430	121024 (121.024 sec)

Table B.2: Experiment Two: Dark Matter

Run	Start	Stop	Total - milliseconds
1	1118654767182	1118654908245	141063 (141.063 sec)
2	1118655025493	1118655150373	124880 (124.880 sec)
3	1118655207826	1118655332665	124839 (124.839 sec)
4	1118737890300	1118738015470	125170 (125.170 sec)
5	1118738121622	1118738244689	123067 (123.067 sec)
6	1118738323733	1118738448643	124910 (124.910 sec)
7	1118738591628	1118738728541	136913 (136.913 sec)
8	1118825089991	1118825216563	126572 (126.572 sec)
9	1118825344177	1118825468976	124799 (124.799 sec)
10	1118825587246	1118825712046	124800 (124.800 sec)
11	1118826024195	1118826147482	123287 (123.287 sec)
12	1118910148205	1118910274827	126622 (126.622 sec)
13	1118910567868	1118910692247	124379 (124.379 sec)
14	1118911051604	1118911176413	124809 (124.809 sec)
15	1118911338807	1118911458619	119812 (119.812 sec)

Table B.3: Experiment Three: Dark Matter

Run	Start	Stop	Total - milliseconds
1	1119345388563	1119345510338	121775 (121.775 sec)
2	1119346218850	1119346340545	121695 (121.695 sec)
3	1119346629771	1119346751476	121705 (121.705 sec)
4	1119347125814	1119347247519	121705 (121.705 sec)
5	1119347370556	1119347492341	121785 (121.785 sec)
6	1119347821815	1119347943690	121875 (121.875 sec)
7	1119348041691	1119348164147	122456 (122.456 sec)
8	1119351884557	1119352006352	121795 (121.795 sec)
9	1119352098815	1119352220670	121855 (121.855 sec)
10	1119352422100	1119352544416	122316 (122.316 sec)
11	1119352910793	1119353043323	132530 (132.530 sec)
12	1119353276619	1119353398424	121805 (121.805 sec)
13	1119356579778	1119356706541	126763 (126.763 sec)
14	1119356980935	1119357103241	122306 (122.306 sec)
15	1119357776119	1119357897864	121745 (121.745 sec)

Table B.4: Experiment Five: Dark Matter

Run	Start	Stop	Total - milliseconds
1	1114692630072	1114692742244	112172 (112.172 sec)
2	1114692783563	1114692887863	104300 (104.3 sec)
3	1114692938366	1114693042856	104490 (104.49 sec)
4	1114693071898	1114693170469	98571 (98.571 sec)
5	1114693346753	1114693445174	98421 (98.421 sec)
6	1114693662827	1114693792955	130128 (130.128 sec)
7	1114693833062	1114693938594	105532 (105.532 sec)
8	1114693964732	1114694073358	108626 (108.626 sec)
9	1114694136348	1114694240678	104330 (104.33 sec)
10	1114694349725	1114694448637	98912 (98.912 sec)
11	1114694482216	1114694602569	120353 (120.353 sec)
12	1114694695653	1114694804980	109327 (109.327 sec)
13	1114694998728	1114695117239	118511 (118.511 sec)
14	1114695264651	1114695372366	107715 (107.715 sec)
15	1114695512607	1114695623347	110740 (110.740 sec)

Table B.5: Experiment Six: Dark Matter