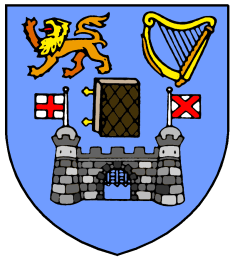
A Generic Framework for Grid-Enabled Visualisation and Computational Steering, and its Characterisation.



A Thesis

Submitted to the Office of Graduate Studies

of

University of Dublin, Trinity College

in Candidacy for the Degree of

Doctor of Philosophy

by Ronan Watson

March 2011

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Ronan Watson

March 16, 2011

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Ronan Watson

March 16, 2011

Acknowledgments

I would like to thank Dr. Brian Coghlan, my supervisor, for all of his help, advice and encouragement. After five years under his supervision I am still amazed at his supervisory skills and above all at the patience he has shown me throughout my time here. This work would not have been possible without him.

I would like to extend my gratitude to the past and present members of the Computer Architecture and Grid Group for making the last five years a thoroughly enjoyable experience. I would especially like to mention Eamonn and John for putting up with all of my questions and for giving me their attention and expertise no matter how busy they were. Their consistent good humoured nature has made working here a pleasure (even with John's terrible terrible jokes).

I particularly thank my parents. They have always emphasised education as an important part of my life and have gone to enormous lengths in providing me support, both personal and financial, to make this possible. I thank them for their relentless encouragement and interest in everything I do, and I hope that this is finally the end of the financial part :)

Thanks to all those who generously gave permission and assistance that enabled benchmarking on resources in Grid-Ireland, UK NGS, CSIC, NIKHEF, PSNC and EGEE, see Section 5.6, and to Science Foundation Ireland for its financial support.

Finally, I would like to thank all my friends. I would not have been able to get through it all without their support and loyalty. Special mention though must go to Marc, Mark, Claire and As'ad. Their friendship has been one of the most important things to me in the last few years, and hopefully, for many more years to come. Thanks guys.

Ronan Watson

Abstract

This thesis presents a generic framework for grid-enabled visualisation and computational steering that is amenable to characterisation using mathematical models based on benchmarked resources. The models enable designers, providers or users to select the most suited interactive computational steering and rendering resources available to them on the Grid. The framework for visualisation aims at giving a greater performance than a desktop computer can provide, achieved at a cost much cheaper than a CAVE, while allowing simultaneous usage by multiple users.

This framework provides multiscale multimodal multiuser grid-enabled visualisation and has the potential to assist the penetration of the Grid into the domain of advanced interactive 3-d visualisation and geographical rendering from a user's desktop. While the existing desktop tools and techniques for interactive visualisation are of a general purpose nature and offer limited compute and data intensive graphical visualisation and interactivity, this framework leverages the power of the grid to offer: 1) more advanced visualisation 2) real time interactivity with the rendered content; 3) and integration with key data intensive graphical applications that would benefit from the use of the grid. The framework is conceptual, to allow users to analyse and/or design for their own specific application, but a comprehensive example application has been implemented.

To enable the characterisation of a framework for grid-enabled visualisation and computational steering, a mathematical model is constructed to help identify the complicated relationships between the variables of the architectures and systems used. The creation of this model allows for it to be used in a number of ways, from the prediction of application performance on a grid, the comparison of site and grid performance, to suggesting for grid providers.

A roadmap is presented for the future evolution of this approach.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Grids are used to allow for the secure sharing of resources across institutional and geographical boundaries and for the collaboration of users through Virtual Organisations (VOs). The resources that are shared using Grids include computational power, data storage, instruments and visualisation. Traditionally users would have had to fit the scale of their scientific discovery to the isolated resources to which they had access so as to produce results in an efficient and acceptable time frame. The time spent gaining access to resources and the consequent difficulty in assembling large collaborations further constrained science. However projects such as EGEE [1], int.eu.grid [2] and Grid-Ireland [3] represent successful efforts to coordinate Virtual Organisations and their resources across Europe, allowing for physicists, biologists and other scientists to access resources that would otherwise have not been available to them and to participate in collaborative science. This provides the users with more time to concentrate on the scientific problem at hand by reducing the time taken in considering the practicalities of resource availability.

While most of these problems can be solved in a batch submission style, some solutions require computational steering and the visualisation of complex data. Computational steering in a Grid environment consists of the timely transfer of results, or partial results, to the user so that they can guide future computations into some area of interest.

This thesis presents a generic framework for grid-enabled visualisation and computational steering that is amenable to characterisation using mathematical models based on benchmarked resources. The models enable designers, providers or users to select the most suited interactive computational steering and rendering resources available to them on the Grid.

1.1 Motivation

With today's volume of data that is collected and analysed, scientists are turning to the visualisation of this data to help their understanding of results from experiments and computations. This need to visualise large and complex data sets has led to the supply of expensive high performance visualisation workstations and clusters to many universities.

Visualisation Spectrum

The Visualisation Spectrum that will be discussed in this thesis ranges from tightly-coupled systems, like Cave Automatic Virtual Environments (CAVEs) for example, to loosely-coupled systems, like desktop machines. Describing a system as tightly-coupled means that each component in that system depends heavily on one or more other components in that system. If one component fails or is not available it will impact heavily on the system as a whole. Loosely-coupled systems are the opposite in that the components of a loosely-coupled system are not as dependent on each other and therefore the system will not degrade as easily as a tightly-coupled system.

Tightly-coupled systems are usually fast and safe in that the risk of transmission errors is very low. Loosely coupled systems are usually more error prone but much more flexible and scalable.

A Cave Automatic Virtual Environment (better known by the recursive acronym CAVE) is an immersive virtual reality environment where projectors are directed to three, four, five or six of the walls of a room-sized cube. The images projected to these walls are generated by a dedicated high performance visualisation cluster situated close to the CAVE. As CAVEs are tightly-coupled systems their performance is usually very good but unfortunately they are not easily scalable, are very costly and the failure of any one component can greatly affect the overall usefulness of the CAVE.

Desktop machines are loosely-coupled systems and are a cheaper alternative to CAVEs, however they are at the lower end of the spectrum. The performance of a desktop machine in rendering visualisation software will greatly depend on the quality of its video card. However the desktop machine has advantages over a CAVE in that it can be scaled to include more desktops and in doing so improve robustness.

Visualisation Space

The Visualisation Space described here is the 3-d space created when graphing the envelope of performance, cost and number of simultaneous users of visualisation systems, shown in Figure 1.1. These systems can lie anywhere in between the tightly-

and loosely-coupled systems that are in the visualisation spectrum described in the previous section.

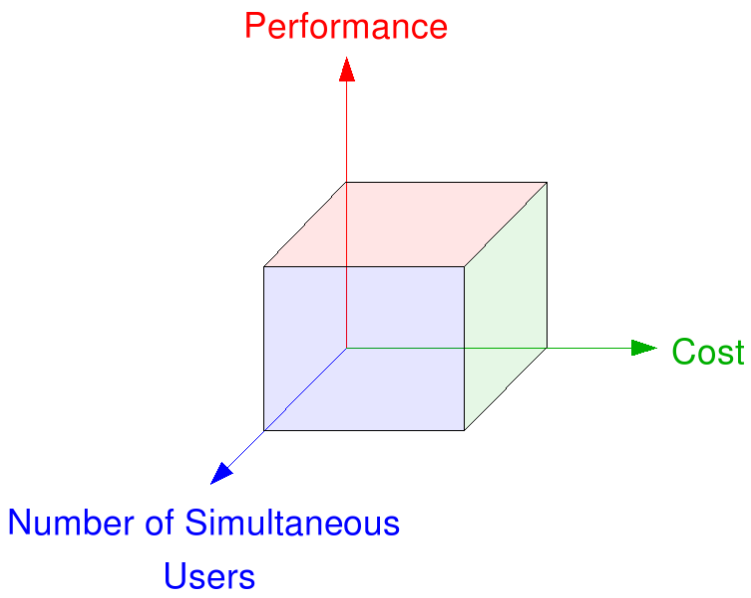
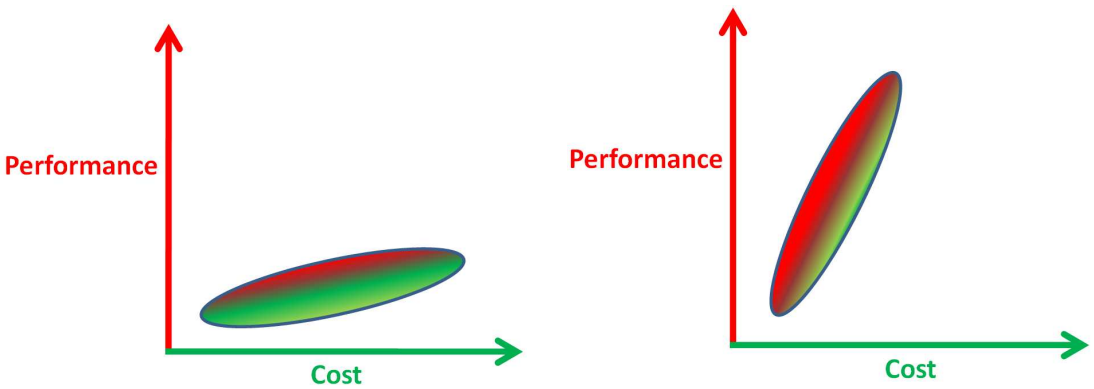


Figure 1.1: Visualisation Space

Figure 1.2(a) demonstrates a visualisation space that is far from ideal as the performance/cost envelope yields a high cost per increment of performance. Preferably, one wants the performance/cost envelope to yield a low cost per performance increment as in Figure 1.2(b).



(a) Non-ideal Performance versus Cost. (b) Ideal Performance versus Cost.

Figure 1.2: Visualisation Space.

The Visualisation Pipeline

The visualisation pipeline is considered the core component of real-time graphics [4], Figure 1.3, and the slowest part of this pipeline determines the rendering speed. The

model section of the pipeline is implemented in software running on CPUs and the *rendering* stage of the pipeline is typically performed on the GPU. Therefore the overall performance of a visualisation workstation is determined by the host system and the GPU.

Figure 1.3: Single Visualisation Pipeline

In today's video card market there is a simple metric in determining the performance that will be achieved by the card and that is the cost of the card. In general the more spent on a video card the better the performance. The technical aspects that determine the performance of a graphics card can be measured by the memory bandwidth (GB/sec), peak fillrate (Billion Pixels/sec) and the number of shaders contained on a card. These metrics of a number of selected graphics cards are plotted against their cost and shown in Appendix B . The trend shows that the more expensive the card, the better values for these metrics.

These figures assume a single user using a visualisation workstation. As visualisation performance is very dependent on the video card, typically the users of these workstations only make use of the workstation's highest-performance pipeline, i.e. a single pipeline, as in Figure 1.3 and graphed on Figure 1.4. What happens when more users are added?

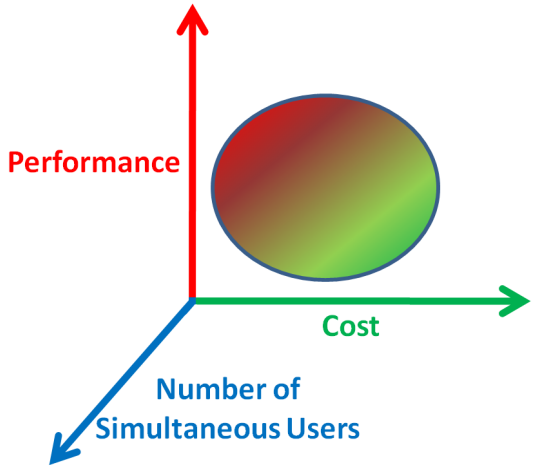


Figure 1.4: Visualisation Space.

Summary

The motivation of this thesis is to consider a framework for visualisation that gives a greater performance than a desktop computer can provide, achieved at a cost much cheaper than a CAVE, while allowing simultaneous usage by multiple users, and to do so on a mathematical basis. That is, we want to achieve a point in the visualisation space that is optimal in cost, performance while maximising the numbers of users that can avail of the visualisation resource, thereby supporting the middle ground.

This thesis presents a methodology that has been developed to share these visualisation facilities on demand in the same way other resources are shared for batch submission style problems and to do so on a mathematical basis. A comprehensive application which tests this framework thoroughly is also described within this thesis. This application also provides a useful example for grid users and developers to help understand better the architecture and components of the framework and its models.

1.2 Objectives

The aim of this undertaking is to investigate whether there is a cost/performance/number of users ratio benefit in combining multiple heterogeneous components of a grid infrastructure to create a loosely-coupled distributed visualisation resource, over traditional single-user tightly-coupled systems. Also, the author wishes to explore the area of benchmarking CPUs and GPUs, for the purpose of creating mathematical models to describe an optimised combination of these heterogeneous grid resources.

Therefore, the main focus of this thesis is the development of an interactive, extensible framework for real-time visualisation applications and associated mathematical models, and the evaluation of these models through testing and experimentation. That is, a multiuser multiscale framework for the middle ground that exploits distributed computing, and is

- between realtime visualisation and int.eu.grid's[2] "human-in-the-loop"
- between multiscale visualisation and int.eu.grid's single-scale visualisation
- between dedicated rendering engines and rendering libraries.

What this thesis intends to show is that there is a way of increasing the users of shared visualisation resources, lowering the cost through this shared use, and increasing the performance available to the users.

1.3 Contributions of the Work

The author's contribution is the development of a generic framework for multiscale multimodal visualisation, described in *Multiscale Multimodal Visualisation on a Grid* [5], using grid infrastructure that is amenable to mathematical characterisation. This framework is built using a grid visualisation architecture developed by the author within the scope of the WebCom-G project [6] and visualisation technology developed by the CrossGrid [7] and int.eu.grid [2] projects. Statistical models for the framework are presented, based on linear regression. The thesis presents an example multimodal visualisation application developed within the framework that will ultimately serve an educational and exploratory objective by simplifying and increasing the understanding of the framework for the grid user (VO), the grid administrator, and for new grid user communities. The framework can integrate existing Grid and non-Grid tools and techniques for high end visualisation and interactivity:

- The techniques developed and utilised are described in detail in the following chapters.
- The framework, including the acquisition of data and process involved in creating a real-time virtual grid application, VirtualGrid [8], is described. The development procedures followed can be of use to other developers interested in creating similar real-time applications.
- The performance of the framework has been modelled in detail and this provides developers with a measure as to the effectiveness of the techniques used, which they can apply to their own projects.

1.4 Overview of Chapters

The rest of the thesis has been divided up into the following chapters:

- **Chapter 2** provides a detailed overview of the previous background and related work in the area of distributed computing and interactive grid applications.
- **Chapter 3** details the visualisation framework for the Grid which allows shared rendering resources to be made available to grid users, and presents an example application of the framework, called VirtualGL.
- **Chapter 4** presents a mathematical model for characterisation of the visualisation framework.

- **Chapter 5** evaluates the performance of the framework described in Chapter 4. Details of the test hardware and conditions are given and results from this evaluation are discussed.
- **Chapter 6** concludes the work detailed in this thesis and suggests future work and improvements, that is, a roadmap for the future evolution of this approach.

Chapter 2

The Grid and Interactive Visualisation

2.1 Introduction

This chapter covers the main research related to distributed computing, workflows, real-time visualisation, visualisation on grids and interactive visualisation applications on the Grid in the following manner:

- An introduction to distributed computing and the many different types of distributed computing which have enabled users to use multiple machines for their compute intensive applications, plus detailed specific examples of distributed computing and those projects which have developed software for research and commercial based distributed computing.
- A description of current work in the area of workflows, presenting scientific and business workflows and the workflow engines that power them.
- An explanation of Real-Time rendering and the current state and performance levels that are implied in using the term real-time.
- An introduction to the many grid projects that use some form of visualisation as part of their offering to the grid environment. These projects range across the world and give a very good insight into the state of visualisation on grids, and the technologies that make these interactions with the Grid possible.
- Finally, a description of the interactive and visualisation applications that are available on the Grid at present, including visualisation of the Grid itself.

2.2 Distributed Computing

This section will describe distributed computing and the many types of systems that come under the umbrella description that is distributed computing. As previously explained, distributed computing is the means by which computers, that are separate entities in their own right, can be loosely coupled together to perform a computational task.

2.2.1 Middleware

Middleware is the software which provides the means by which homogeneous or heterogeneous systems can 'talk' to each other. This communication is usually across a network where the middleware allows multiple processes running on these systems to interact. For a long while, middleware has either been custom coded for individual projects or has come in the form of proprietary products or suites, most notably as Enterprise Application Integration (EAI) software. The emergence of industry-agreed web services specifications is now enabling convergence on standards-based distributed middleware, which in theory should allow all systems to automatically connect together on demand.

Globus

The Globus project [9][10][11][12][13] is an American multi-institutional research effort that seeks to enable the construction of computational Grids. Globus provides a software infrastructure that enables applications to view distributed heterogeneous computing resources as a single virtual machine. A central element of the Globus system is the Globus Toolkit, which defines the basic services and capabilities required for constructing computational Grids. The toolkit consists of a set of components that implement basic services, such as *security*, *resource location*, *resource management*, *data management*, *resource reservation*, and *communications*.

Globus is constructed as a layered architecture in which higher-level services can be developed using the lower level core services [14]. Its emphasis is on the hierarchical integration of Grid components and their services. This feature encourages the usage of one or more lower level services in developing higher-level services. Resource and status information is provided via an Lightweight Directory Access Protocol (LDAP) based network directory called Metacomputing Directory Services (MDS) [15]. MDS consists of two components, Grid Index Information Service (GIIS) and Grid Resource Information Service (GRIS). GRIS implements a uniform interface for querying re-

source providers on a Grid for their current configuration, capabilities, and status. GIIS pulls the information from multiple GRIS services and integrates it into a single coherent resource information database. The resource information providers use a push protocol to update GRIS. Thus MDS follows both push and pull protocols for resource dissemination. Higher-level tools such as resource brokers can perform resource discovery by querying MDS using LDAP protocols. The MDS namespace is organised hierarchically in the form of a tree structure. Globus offers quality of service (QoS) in the form of resource reservation. Globus provides scheduling components as part of its toolkit approach but does not supply scheduling policies, relying instead on higher-level schedulers.

Legion

Whereas Globus is a collection of tools from a toolkit, Legion [16][17] provides standard operating system services - process creation and control, interprocess communication, persistent storage, security and resource management - on a Grid. By doing so, Legion abstracts the heterogeneity inherent in distributed resources and makes them look like part of one virtual machine. Legion is organised by classes and metaclasses, and comprises of the following ideas: *everything is an object*, *classes manage their instances* and *users can define their own classes*. In Legion objects represent all hardware and software components and every object is a process that responds to method calls from other objects within the system. Every Legion object has its own class object which defines and manages it and these class objects can create new instances, schedule an object for execution, activate or deactivate an object and provide state information to client objects. Users can also override or redefine the functionality of a class.

The scheduling process in Legion broadly translates to placing objects on processors. Scheduling is invoked not just for running users' jobs but also to create any object on a Grid, such as a file, a directory, an application or even a scheduler. Legion contains the following core object types: *classes and metaclasses*, *host objects*, *vault objects*, *implementation objects and caches*, *binding agents* and *context objects*. After an object is created on a processor, it can perform its tasks, for example, respond to read/write calls if the object is a file, or respond to status requests if it is a job. Therefore, object placement is crucial to the design of the Legion run-time system because it can influence an object's run-time behaviour greatly. An improper placement decision may impede an object from performing its tasks, for example, because it cannot start on any processor of a given architecture or because the processor is no longer available. Even if a placement decision ensures that an object starts correctly, it does not guarantee that the decision is beneficial to the user. A good placement decision is certain to vary

depending on the object that is being placed and the user's goals, as well as resource usage policies. The Legion interface is described in an Interface Definition Language (IDL) and it is written in the Mentat Programming Language (MPL), which means it is necessary to port MPL onto each platform before Legion can be installed.

Condor

Condor [18] is a high-throughput computing software framework for the parallelisation of computationally intensive tasks. It is used to manage workload on a dedicated cluster of computers and runs on Linux, Unix, Mac OS X, FreeBSD, and the Windows operating systems. Condor is developed by the Condor team at the University of WisconsinMadison [19] and is freely available for use. Condor can run both sequential and parallel jobs and it supports the standard Message Passing Interface (MPI) and Parallel Virtual Machine (PVM) libraries in addition to its own Master Worker (MW) library for extremely parallel tasks.

Condor-G [20][21] allows Condor jobs to be used on resources not under its direct control. It is mostly used to talked to Grid and Cloud resources, Globus, UNICORE and Amazon EC2, but it can also be used to talk to other batch systems, like Torque [22][23], PBS [24][25] and LSF [26][27]. Support for Sun Grid Engine is currently under development as part of the EGEE project. Condor is one of the job scheduler mechanisms supported by GRAM (Grid Resource Allocation Manager), a component of the Globus Toolkit.

Virtual Data Toolkit (VDT)

VDT provides a build test infrastructure for grid software. Its building infrastructure is based on NMI pools of heterogeneous resources, each installed with Condor. The User Interface (UI) to the Condor/NMI pools is based on a Perl implementation. Builds are triggered to produce, in particular, VDT Globus, MyProxy, GSI-OpenSSH and UberFTP-client for the gLite middleware distribution. Currently the NMI pool consists of SL 4/5 Debian 4/5, AIX, PS3(Yellow Dog Linux), MacOS and openSUSE build machines. The value of this software is that it also provides patches to the Globus distribution making the is more scalable.

Unicore

UNICORE (UNiform Interface to COmputer REsources) [28][29] is a Java-based environment for accessing remote supercomputers. The idea behind UNICORE is to support the users by hiding the system and site-specific technologies and by helping

to develop distributed applications. The UNICORE design goals include a uniform and easy to use GUI, an open architecture based on the concept of an abstract job, a consistent security architecture, minimal interference with local administrative procedures, exploitation of existing and emerging technologies, zero administration user interfaces through standard Web browsers and Java applets. UNICORE is designed to support batch jobs; it does not allow for interactive processes. The user is provided with a unique UNICORE user-ID to uniformly get access to all UNICORE sites. An intuitive GUI allows job preparation and control.

gLite

gLite [30][31] is a middleware which is produced by the Enabling Grids for E-sciencE (EGEE) [1] project. It follows on and makes use of contributions from many projects including LCG [32][33], Condor and Globus via VDT [34][35], and EDG [36][37]. gLite middleware is currently deployed on hundreds of sites as part of the EGEE project. The services, or node-types, that are provided by gLite include Computing Element, Workload Management, Storage Element, Catalog, Information and Monitoring, and Security and each of these services has taken components from the previously mentioned projects.

2.2.2 Grids

This section describes some of the grid projects that exist and that have used some of the middleware software described in the previous section.

EDG

The EDG (European Data Grid) project [36][37], which spanned from 2000 to 2004, greatly extended the Globus Toolkit version 2.4 with higher-level services for authorisation, job submission, brokerage and information gathering. It can be considered the progenitor of the gLite middleware. It supported 12 Virtual Organisations and at its peak had over 1000 CPUs available to users.

EGEE

One of the largest grid computing projects in Europe is the EGEE (Enabling Grids for E-sciencE) [1] project which is funded by the European Commission from 2004 to 2010. The project aims to provide researchers in academia and industry with access to major computing resources, independent of their geographic location. EGEE uses

the gLite middleware and generally, the EGEE Grid infrastructure is ideal for any loosely-coupled scientific application, although support is growing for more tightly-coupled applications using the MPI API [38][39][40]. The EGEE Grid consists of, as of October 2009 [41], 296 sites in 56 countries with 76,380 CPUs available to users 24 hours a day, 7 days a week and connected to more than 131 Petabytes (over 137 million Gigabytes) of storage. Its successor will be the European Grid Infrastructure (EGI) [42], beginning in 2010.

TeraGrid

The TeraGrid project [43] was launched in 2001 by the National Science Foundation in the United States and in 2005 was extended up until 2010. Currently, TeraGrid resources include more than a petaflop of computing capability and more than 30 petabytes of online and archival data storage, with rapid access and retrieval over high-performance optical networks. TeraGrid users primarily come from United States universities and currently it has 4,000 users at over 200 universities. Academic researchers in the United States can obtain exploratory allocations in "CPU hours" based on an abstract describing the work to be done. TeraGrid computational resources run a set of software packages called "Coordinated TeraGrid Software and Services" (CTSS). CTSS provides functions such as single sign-on, remote job submission, workflow support and data movement tools. CTSS includes the Globus Toolkit and Condor, verification and validation software, and a set of compilers and programming tools.

OSG

Begun in 2004, The Open Science Grid (OSG) project [44] provides high-throughput computing across the United States. There are 72 sites and 30 virtual organisations in OSG and in total, the OSG comprises over 25,000 computers with over 43,000 processors. There are 90 distinct computational and storage nodes in the grid, which are distributed across the United States and Brazil. VDT is a product of the OSG and is used as its grid middleware distribution. High-energy physics uses a large chunk of OSG's resources but there are several other sciences that are actively using OSG:

- nanoHUB [45][46]: is comprised of resources contributed by the Nanotechnology community in the United States and is aimed at providing educational applications, professional networking, and interactive nanotechnology simulation tools.
- LIGO [47][48]: is a project set up in 1992 at MIT and Caltech, and is a large physics experiment which is attempting to directly detect gravitational waves. It currently provides the compute resources of two LIGO sites to OSG.

- CHARMM [49][50]: or Chemistry at HARvard Macromolecular Mechanics is a software application developed at Harvard University for modelling the structure and behaviour of molecular systems. Molecular dynamics jobs usually take days to complete, and as the OSG sites are optimised for shorter jobs, each long job is split into smaller jobs.

2.2.3 Clouds

Cloud computing is an emerging approach to shared infrastructure in which large pools of systems are linked together to provide IT services.

In *A vision for the Internet* [51], Kleinrock presents his vision from 1969 of the computing utility based on the service provisioning model, 'As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of "computer utilities" which, like present electric and telephone utilities, will service individual homes and offices across the country.' This vision anticipates the massive transformation of the entire computing industry in the 21st century whereby computing services will be readily available on demand.

The need for such environments is fuelled by dramatic growth in connected devices, real-time data streams, and the adoption of service oriented architectures and Web 2.0 applications, such as mashups, open collaboration, social networking and mobile commerce. Continuing advances in the performance of digital components has resulted in a massive increase in the scale of IT environments, driving the need to manage them as a unified cloud. Virtualisation is playing an increasingly important role and it is one of the reasons that Cloud Computing has been named as such. With virtualization the resources available to a user can expand and collapse depending on the needs of the user. This is achieved by creating or deleting virtual machines using software like Xen [52][53]. Therefore the resources available to the user are not fixed, but continually moving and changing size (according to the requirements of the user), like a cloud. Figure 2.1 describes what a possible cloud system could be like. Here there are services such as the User Interface (UI), Management Systems and Monitoring that are persistent and in the middle lies the "cloud" resource. Obviously the owners and administrators of the cloud computing resources have a fixed set of resources, but for the user on the outside, with the aid of Xen and virtual machines, the available resources appear to be variable.

In *MarketOriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility*, Buyya et al. present a 21st century vision of computing along with an analysis of the major Cloud platform offerings available. In the following sec-

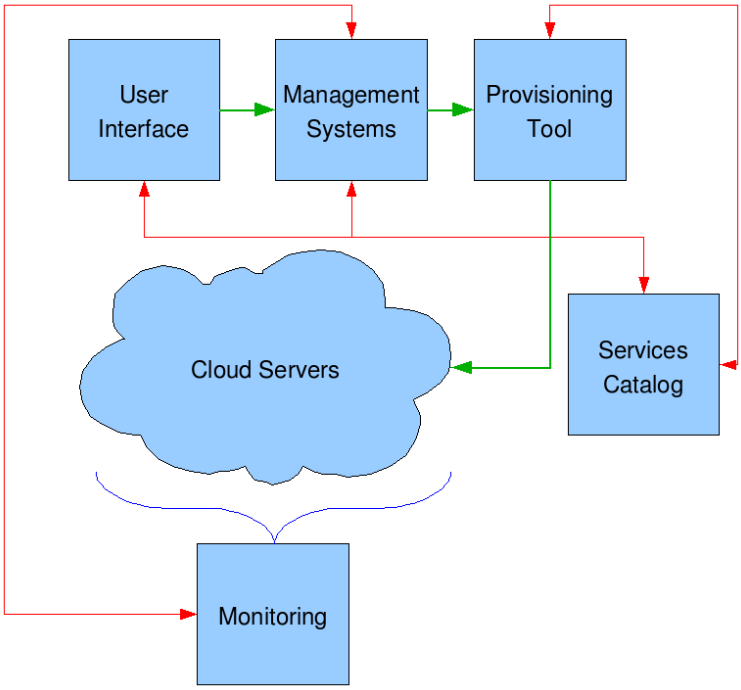


Figure 2.1: Cloud Computing

tions I will describe these platforms including some of the smaller commercial offerings.

Amazon Elastic Compute Cloud - EC2

EC2 [54][55] lets a user rent time on a "cloud" of computers. The computer specifications depend on the the computing needs of each user and how much the user can afford to pay. Amazon split their machines into separate instance types which differ in the speed of the CPU, amount of memory available, whether it's a 32-bit or 64-bit platform, the I/O performance and the storage capacity. A *Standard Small Instance* offers computers with one EC2 Compute Unit, which is equivalent to a CPU of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor, 1.7GB of RAM, a 32-bit platform, moderate I/O performance and 160 GB of local disk. The cost of these machines is 10 cents per instance per hour. The most expensive instance available at 80 cents per instance per hour is the *High-CPU Extra Large Instance*. This provices the renter with 20 EC2 Compute Units, 7GB of RAM, a 64-bit platform, high I/O performance and 1690GB of local disk.

The "machines" that Amazon delivers with EC2 are actually virtual machines, each running on top of the Xen platform. You create a virtual machine by storing a disk image inside S3 using special tools that Amazon provides and then running a Java program that instantiates the virtual machine. A second Java program lets

the user monitor the progress of the machine's creation; when it is ready, the script displays the computer's hostname. The image that is instantiated should have an account that lets the user log into the machine. Because EC2 is based on Xen, it should support any Linux distribution as well as NetBSD, FreeBSD, Plan 9, and some other operating systems. In practice, EC2 is largely based on the RedHat Fedora Core operating system, although there are instructions on the Internet for using it with Ubuntu distributions. Amazon makes no promises about the reliability of the EC2 computers: each machine can crash at any moment, and they are not backed up. Under normal circumstances these machines don't crash, but, sometimes computers do fail. If a user wants reliable storage, it is advisable to run two or more EC2 machines as a cluster. A better approach, though, is to have the EC2 machines store information in S3, which is sold as a reliable, replicated service.

IBM

Blue Cloud [56] is a series of cloud computing offerings that will allow corporate data centres to operate more like the Internet by enabling computing across a distributed, globally accessible fabric of resources, rather than on local machines or remote server farms. Blue Cloud is based on IBM's Almaden Research Centre cloud infrastructure. It includes Xen and PowerVM [57] virtualised Linux operating system images and Hadoop [58][59] parallel workload scheduling. It is based on an open-source project called Hadoop that manages computing resources across large clusters of computers. Hadoop includes an open-source version of *MapReduce*, the same functional software Google uses to efficiently distribute its computing chores across its servers around the world.

Sun Cloud

Sun Cloud [60] is an on-demand Cloud computing service operated by Sun Microsystems. The Sun Cloud Compute Utility provides access to a substantial computing resource over the Internet for one US dollar per CPU-hour. It is based on and supports open source technologies such as Solaris 10, Sun Grid Engine, and the Java platform and is available worldwide.

A typical application that can run on the Compute Utility must be self-contained, able to run on the Solaris 10 Operating System (OS) and implemented with standard object libraries included with the Solaris 10 OS or user libraries packaged with the executable. The application must run to completion under control of shell scripts and have no requirement for interactive access. The total maximum size of applications

and data can not exceed 10 gigabytes and must be packaged for upload to Sun Cloud as one or more ZIP files of 300 megabytes or smaller.

GoGrid

GoGrid [61] is a cloud infrastructure service and is the cloud hosting division of the ServePath Dedicated Hosting company [62] which is based in the United States. It uses Xen technology to host Linux and Windows virtual machines which are managed by a multi-server web-based control panel. They have a pay-as-you-go pricing plan that ranges from just under 10 cents an hour for one Xeon processor, 0.5GB of memory and 30GB of local storage to 152 cents and hour for 6 Xeon processors, 8GB of memory and 480 GB of local storage.

2.2.4 Future Grids-n-Clouds

In essence, Grids ease sharing and collaboration, and Clouds ease provisioning. These are not mutually exclusive, and so future marriages of the technologies are likely.

2.3 Workflow Engines

Workflow engines are applications that allow users to graphically assemble a connected job or set of jobs that they want executed. These workflow engines give simplified access to large sets of libraries, e.g. R [63][64] and Matlab [65]. There are many workflow systems in existence today, each addressing some aspect of the workflow management problem [66][67].

Workflow systems can be generally classified into two broad categories: Task-based or service-based. Task-based systems, for example Pegasus [68], generally focus on the mapping and execution capabilities and leave the higher-level composition tasks to other tools. There are many workflow systems that support application component composition and execution. Some, such as Triana [69], Kepler [70][71], and VisTrails [72], provide graphical user interfaces for workflow composition and some, such as Karajan [73], provide a scripting language to specify the workflow. Some of these workflow systems support more complex control structure loops, conditionals, and hierarchical workflow definitions.

A commercial workflow application is a software application which automates, at least to some degree, a process or processes. The processes are usually business-related, but it may be any process that requires a series of steps that can be automated via

software. Advanced applications allow users to introduce new components into the operation.

Scientific workflows found wide acceptance in the fields of bioinformatics and cheminformatics in the early 2000s, where they successfully met the need for multiple interconnected tools, handling of multiple data formats and large data quantities.

In attempting to hide the complexity of the Grid from users, workflow engines have been adapted to include services that allow the user to utilise grid resources without having prior knowledge of grid job submission techniques. The following four sections describe examples of workflow engines that have components that are grid-enabled.

2.3.1 Pegasus

Pegasus [68], Planning for Execution in Grids, was developed as part of the GriPhyN project [47]. Pegasus is a configurable system that can map and execute complex workflows on the Grid. Pegasus receives an abstract workflow description from Chimera [74], produces a concrete workflow by means of the Concrete Workflow Generator [75], and submits it to Condor's Directed Acyclic Graph Manager, DAGMan [18], for execution. The abstract workflow describes the transformations and data in terms of their logical names.

For example, in astronomy, users often do not want to know the details of the underlying system, instead they want to retrieve images of an area of the sky of interest to them. In such cases Pegasus is usually integrated into a portal environment where the user is presented with a web form to fill in the desired metadata attributes. Inside the portal, the workflow instance is generated automatically based on the user's input and is given to Pegasus for mapping and then to DAGMan for execution. Examples of this approach can be seen in the Montage project [76] (an astronomy application), the Telescience portal [77] (a neuroscience application), and the Earthworks portal [78] (an earthquake science application). In all these applications, Pegasus and DAGMan are being used to run the application workflows on a national scale infrastructure such as the TeraGrid [43].

2.3.2 WebCom-G

WebCom-G [6] is a grid middleware developed in Ireland, based on work in University College Cork, that uses Condensed Graphs [79] to hide the underlying complexity of a grid infrastructure. To facilitate the development of new applications and the integration of existing applications into WebCom-G, a number of programming and development tools were developed to allow the easy creation of Condensed Graphs.

These tools include

- APIs for languages like Java and C++.
- An XML Schema for expressing Condensed Graphs.
- Visual tools that allow users to create Condensed Graph applications graphically, e.g WebCom-G IDE, see Figure 2.2
- High Level Language compilers that compile existing applications (written in languages like Java) directly into Condensed Graphs. These compilers can extract parallelism from sequentially written applications and have the advantage of not requiring application developers to know anything about the Condensed Graph Model of Computing

Applications can be built using the WebCom-G IDE and submitted to a WebCom-G grid. These jobs can be executed on this grid alone but WebCom-G also provides interoperability with other existing middlewares, e.g Globus.

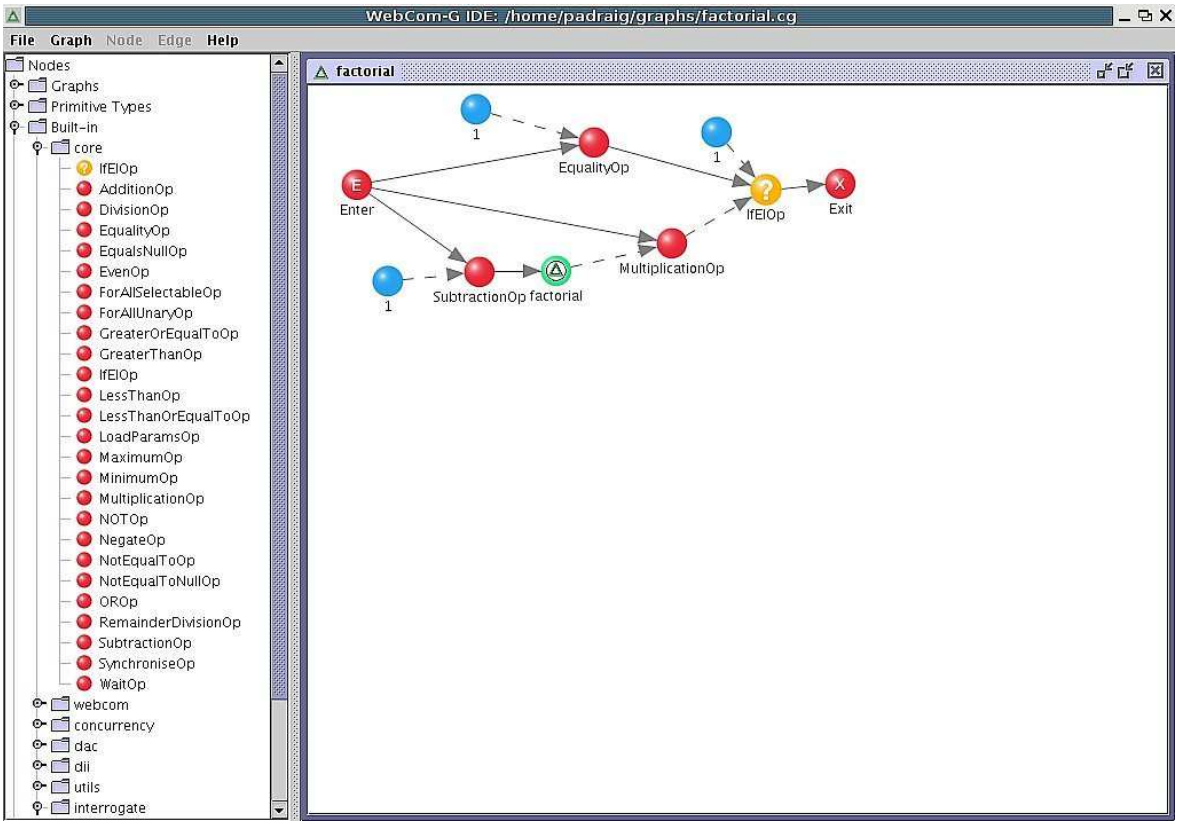


Figure 2.2: WebCom-G IDE. Image courtesy of Dr. Eamonn Kenny of the author's host research group.

2.3.3 Kepler

The Kepler workflow application [70][71] is a scientific workflow system based on Ptolemy II [80][81], developed by researchers from the US at the National Center for Ecological Analysis and Synthesis (NCEAS) at the University of California, Santa Barbara and the San Diego Supercomputer Center at the University of California, San Diego. Ptolemy II is a set of Java packages supporting heterogeneous, concurrent modelling and design. Its kernel page supports clustered hierarchical graphs, which are collections of entities and relations between those entities. Most models of computation in Ptolemy II support actor-oriented design and Kepler extends Ptolemy II by creating an ever increasing number of components, called *actors*, aimed particularly at scientific applications, e.g. for remote data and metadata access, data transformations, data analysis, interfacing with legacy applications, web service invocation and deployment, etc. Target application areas include bioinformatics, cheminformatics, ecoinformatics, and geoinformatics workflows, among others. Figure 2.3 shows a snapshot of the Kepler Integrated Development Environment (IDE) solving two couple differential equations and the plotted results.

2.3.4 Taverna Workbench

Taverna Workbench [82][83] is a workflow IDE, see Figure 2.4, created by the myGrid project [84][85] based on work in Manchester, which allows for the automation of experimental methods through the use of a number of services, including Web Services. The Taverna Workbench is used by users in many domains, such as bioinformatics, cheminformatics, astronomy, social science and music. Taverna workflows can include sub-workflows and these sub-workflows can be included in many workflows and can be shared and edited separately. Taverna workflows do not need to be executed within Taverna Workbench as they can also be run by a command line execution tool, on computational grids and from web pages or portlets. Taverna is also used to run services on computational grids such as EGEE.

2.4 Real-Time Rendering

Real-time rendering covers the generation of graphical images for display at interactive frame-rates. While it is possible to render graphical images off-line at arbitrary detail, as practised in animated feature films, the generation of detailed images in real-time is a difficult task. The most common real-time applications found currently are computer games. Typically, however, a computer entertainment title maintains its detail and

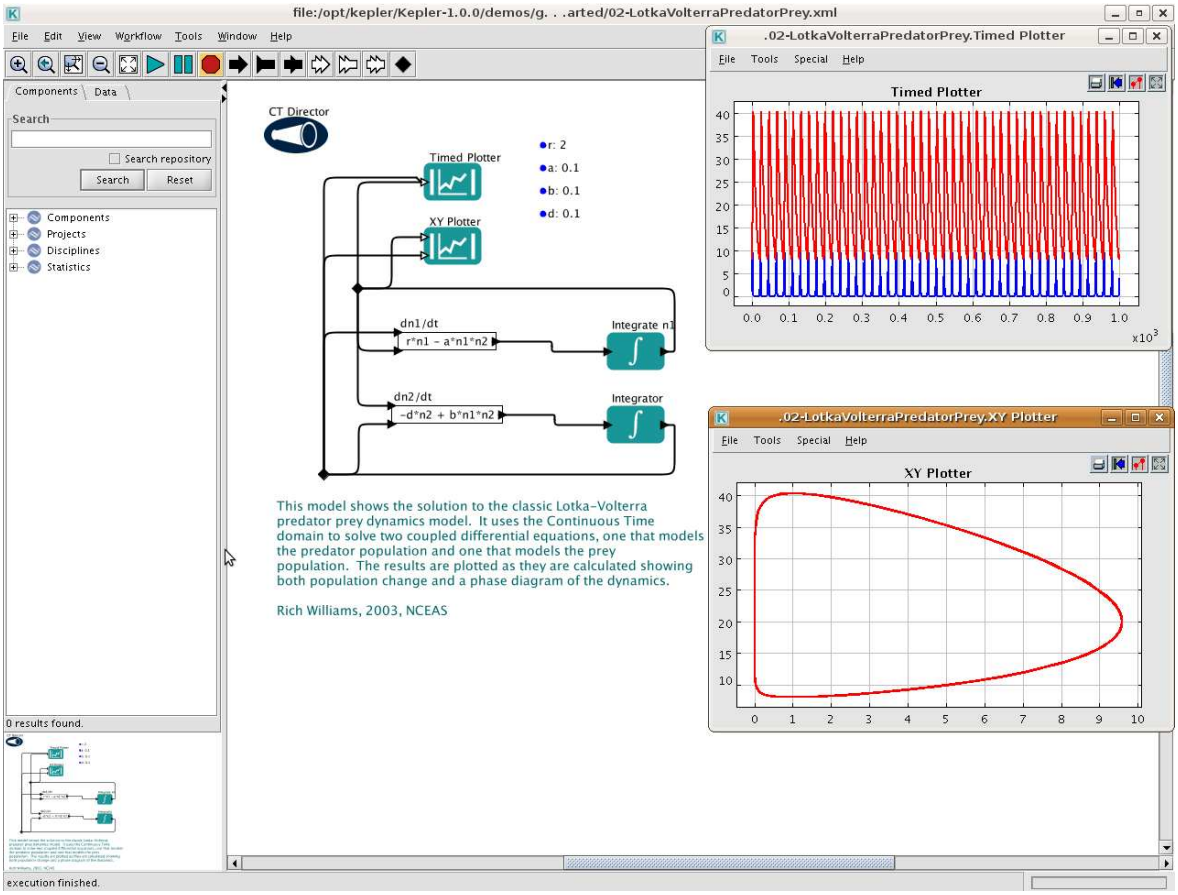


Figure 2.3: Kepler IDE showing the setup involved to solve two couple differential equations and the resulting plots. Image courtesy of Dr. John Ryan of the author's host research group.

real-time display rates by only covering a small physical area or Level, or restricting the users path through the virtual world with obstacles and barriers. Larger areas may be constructed by chaining levels together, but typically there is a transitional period as a new level is loaded into the computer's memory. Real-Time Rendering is typically understood to be over 30 frames per second (fps) [86]. The visualisation pipeline can be conceived in several ways [87]. In this thesis the visualisation pipeline is considered as a sequence of four tasks,

1. computation.
2. interaction.
3. rendering.
4. display.

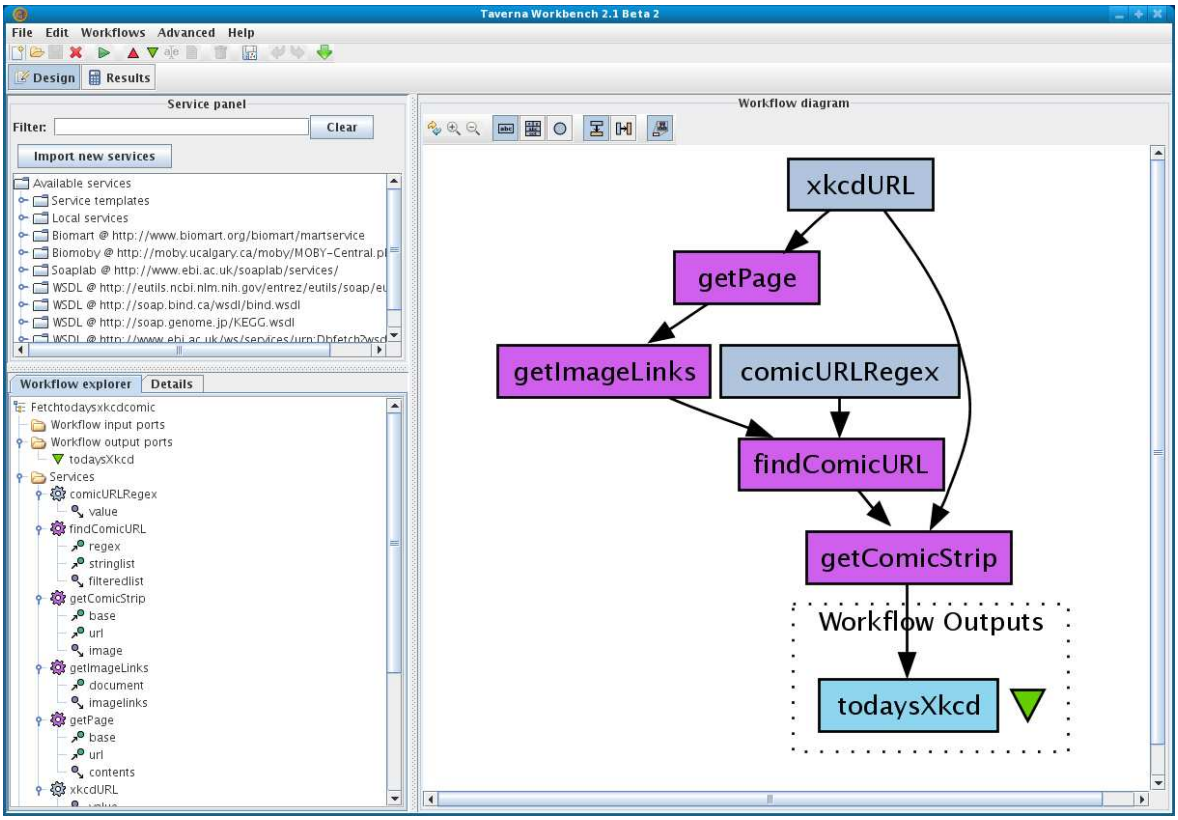


Figure 2.4: Taverna IDE showing the workflow set up to retrieve a html web page. Image courtesy of Dr. John Ryan of the author's host research group.

The Grid can handle the computation and delegate the data produced from the Grid to a dedicated rendering engine which can then stream the images to a display.

As the Grid is spread across a large geographical area, latency becomes a problem to real-time interaction and computational steering of applications. If there is high network latency between the *rendering* and *display* tasks of the pipeline, the minimum frame rate of 30fps might not be achievable. Computational steering may also be affected by high-latency scheduling, which is the delay between job submission and the execution of the job.

The framework proposed in this thesis deals with the provision of this rendering engine (hereafter called the Visualisation Engine), methods of interaction and the streaming of images which reduces the effects of latency. The framework thus completes tasks 1 to 4 of the visualisation pipeline.

2.5 Visualisation and Steering on Grids

This section overviews the technologies, architectures and tools used so far in grid visualisation. The limitations of those solutions are highlighted and the degree of

success of these solutions in various applications is examined.

2.5.1 Visualisation in TeraGrid

No matter how users conduct science, a crucial aspect of scientific discovery is the process of understanding the information contained in the data. Analysis, visualisation, and data exploration are key components in this process. The goal of the American TeraGrid Visualisation (TGviz) effort is to combine existing resources and current technology (including commodity clusters and commodity graphics, terascale visualisation clusters, Grid technology, and efforts, expertise, and tools from each of the TeraGrid partner sites) to enable new and novel ways of visually interacting with and gaining insight into science through the analysis of simulations and data. This is done through the production of visualisation services for users, and providing tools and libraries for researchers in visualisation and graphics. TeraGrid also deploys visualisation resources for batch, interactive, and collaborative visualisation. Users access TeraGrid-aware visualisation applications in a way that appears to be a desktop visualisation solution but in reality makes use of the rich TGviz resources.

The TGviz effort emphasises three principal areas:

- Investigation of challenges to running a large-scale visualisation Grid facility;
- Development of methods to effectively enable the remote interaction of users with sophisticated visualisation systems, and the subsequent delivery of results to users from those systems
- Improvement of the data analysis capabilities of the TeraGrid community through the availability of visualisation services.

The TeraGrid Visualisation Gateway is a web interface for gaining simplified access to a variety of TeraGrid visualisation resources and services. All new users receive a "New User Form" via U.S. postal mail that contains a username and password for the TeraGrid User Portal. This same username and password can also be used to login to the Visualisation Gateway. Once logged in to the gateway, four different services are made available to the user from the gateway:

- **Proxy Manager:** helps the user manage the credentials that are used to access TeraGrid resources. The gateway automatically loads the proxy for the user when logging in. The other services will use the current default proxy to access remote resources and this service allows for additional proxies to be loaded, which can then be set as the default for the currently active session.

- **Remote Visualisation:** enables the launching of remote visualisation sessions on Maverick, Texas Advanced Computing Center's (TACC) TeraGrid Visualisation System [88], through the use of a Virtual Network Computing (VNC) server [1] running on Maverick and a VNC client running in a web browser on the local resource.

- **ParaView:** enables the user to launch a ParaView server [94][95], a parallel visualisation application for large datasets, on the Argonne TeraGrid Visualisation cluster [96], and connect to it via a ParaView client running on a local resource, as shown in Figure 2.5.

- **File Management:** enables the movement of data onto and off of the TeraGrid, as well as between TeraGrid resources. If multiple proxies are loaded, it is possible to individually select which proxy to use for the source and destination of the transfer.

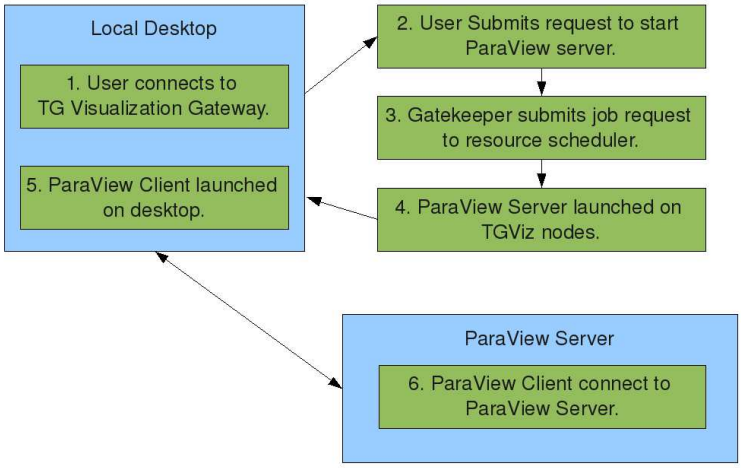


Figure 2.5: ParaView Service on UChicago/Argonne

In addition to Common TeraGrid Software and Services (CTSS) available on all TeraGrid resources, the TeraGrid Visualisation Working Group has also defined Visualisation TeraGrid Software and Services (VTSS). The VTSS consists of a common set of applications and libraries for enabling scientific visualisation that can be found on all resources that choose to support it.

[1]The original AT&T VNC version [89][90] is not widely used any more, because there are different branches with significant improvements available, such as RealVNC [91], TightVNC [92] and UltraVNC [93], that are still full backwards compatible, at least as far as their basic remote control function is concerned.

2.5.2 RealityGrid

A central theme of the UK based RealityGrid [97][98] is the facilitation of distributed and collaborative steering of parallel simulation codes and simultaneous on-line, high-end visualisation. It's architecture is illustrated in Figure 2.6.

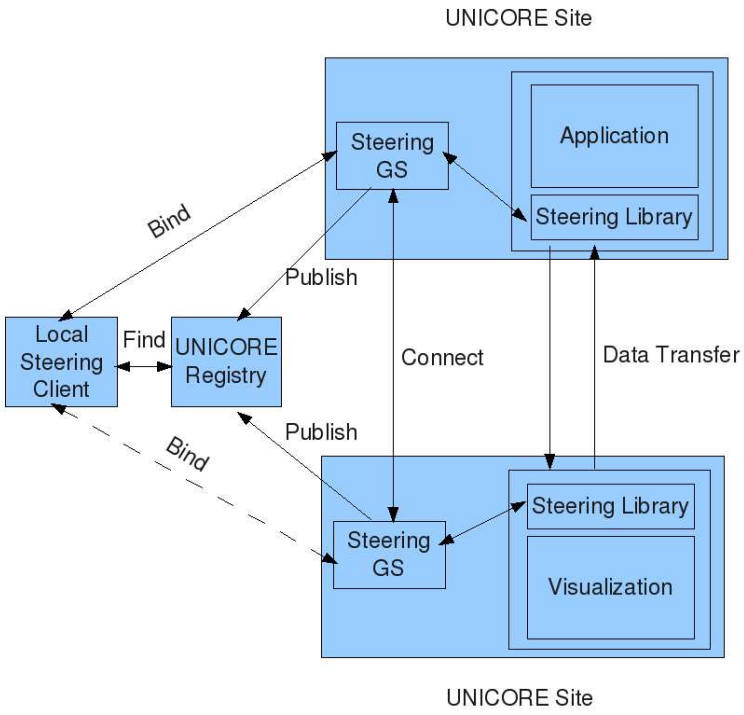


Figure 2.6: RealityGrid

Computational steering may be used to watch and control a calculation as it runs. A simulation which is no longer evolving may be spotted and terminated, saving CPU time wastage. More powerfully, a simulation may be steered through parameter space until it is unambiguously seen to be producing interesting results: this technique is very powerful when searching for emergent phenomena which are not clearly related to the underlying simulation parameters. Steering was performed using the RealityGrid steering library. Once the application has initialised the steering library and informed it of which parameters are to be steered, then after every timestep of the simulation, it is possible to perform tasks such as checkpointing the simulation, saving output data, stopping the simulation, or restarting from an existing checkpoint. The RealityGrid project provides support for jobs that contain Application-Level Checkpointing (ALC) functionality. This support enables jobs to implement fault-tolerance and to implement strategies for long running computations to save state at the end of a fixed length batch run. In either case, jobs can be restarted from checkpoints in subsequent batch allocations. When a steered simulation is started, a Steering Grid Service (SGS) is also

created, to represent the steerable simulation on the Grid. The SGS publishes its location to a Registry service, so that steering clients may find it. This design means that it is possible for clients to dynamically attach to and detach from running simulations. Each running simulation emits output files after certain periods of simulation time have elapsed. The period between output emission is initially determined by guessing a timescale over which the simulation will change in a substantial way; however, this period is a steerable parameter, so that the output rate can be adjusted for optimum visualisation without producing an excessive amount of data. Visualisation clusters were used to render the data and output volumes were sent using Globus I/O from the simulation machine to the remote visualisation machine, so that the simulation could proceed independently of the visualisation; these were then rendered using the open source Visualisation Toolkit (VTK) [99][100] visualisation library into bitmap images, which were in turn multicast over the AccessGrid [101][102] using the Chromium [103], so that the state of the simulation could be viewed by scientists around the globe. In particular, this was demonstrated by performing and interacting with a simulation in front of a live worldwide audience, as part of the SCGlobal track of the SuperComputing 2004 conference. There are many parameters for such a visualisation, such as the region of the simulation being visualised, colour maps, isosurface levels, and orientation of the visualisation geometry. These were controlled through SGIs VizServer software, allowing control of the geometry from remote sites.

Adaptive modelling is a technique which can reduce the need for computational steering. By capturing the user's knowledge base in more detail, information can be adapted and presented to a user in a way that is more intuitive. If the user has little grid and system knowledge, then creating jobs that gather very detailed information is a waste of resources, both storage and compute.

A lightweight visualisation system was created, which can be hosted on a mobile phone, laptop or desktop computer, to provide a set of grid-enabled software components and middleware. The basis of lightweight visualisation was to facilitate efficient and collaborative remote user access to high-end visualisation on the grid. The RealityGrid PDA client [104] was designed as an intuitive visual front-end, enabling the user to discover their applications on the RealityGrid, steer their simulations in real-time and interact with high-end visualisations as if the supercomputing applications were running locally. The PDA Client provides convenient, remote, handheld access to the primary aspects of supercomputing functionality:

- **Resource discovery:** The PDA Client retrieves a list of all the user's currently deployed and active jobs on the grid. The interface enables the user to select an individual job to interact with. It will then establish a remote connection

to the job over the grid and display the appropriate computational steering or visualisation interface, depending on the type of job selected.

- **Real-time computational steering:** The user remotely steers a simulation by entering new parameter values into an input dialog, which is displayed by the user interface. The inputted new parameter values are then dispatched to the grid to update the relevant parameters within the simulation. Once the required steering activity has been instigated it is reflected back to the PDA interface, through a client-requested parameter update, almost instantaneously.

- **Visualisation:** The configurable visualisation interface has been designed to provide the same user services and level of user interaction as that of the desktop computer front-end ensuring a familiar user interface on the PDA. The user can remotely configure the image encoder settings to enable varying levels of image compression, in order to adapt the client for use on low or medium bandwidth wireless networks with higher levels of image compression yielding increased image serving throughput on slower networks. This method also allows the system to produce images with lossless compression when required.

2.5.3 Cactus

The programming framework Cactus [105][106] and the remote steering/visualisation architecture, see Figure 2.7, was developed at Potsdam University in Germany since the mid-1990's specifically to enable scientists and engineers to perform the large scale simulations needed for their science. Cactus was used to perform intercontinental distributed simulation-visualisation scenarios at SC97, and at SC98, a simulation of colliding neutron stars across the two continents was shown, distributing the computational grid across three supercomputers in Garching, Berlin and San Diego. Distributed demonstrations were also shown at SC'99 and SC'2000, with emphasis more on sophisticated and robust interactive visualisation, monitoring and steering, dynamic scenarios, exploitation of networks and the development of user portals and testbeds.

Data from a simulation can either be located on a remote file system, or it could be contained in a virtual file, streamed live from a running simulation via socket connections. Cactus provides several different implementations of data streaming. The most generic approach, for arbitrary data of any type, is based on the HDF5 remote I/O library [107] and its Virtual File Driver (VFD) layer [108]. A Stream driver holds the data to be streamed out of the Cactus simulation as an in-memory file, which can then be sent through a socket to a connected client. The client application uses

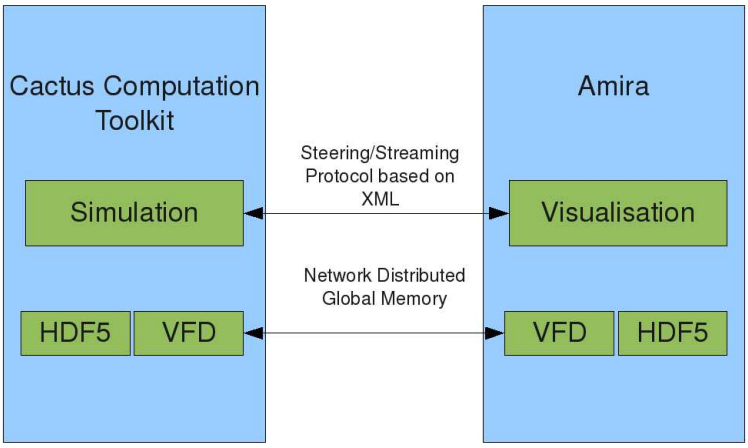


Figure 2.7: The Remote Steering/Visualisation Architecture of Cactus

the same driver to reconstruct the in-memory file which then can be accessed to read the datasets. The design means that applications can use their existing file-based I/O methods immediately for online remote data access without changing I/O interfaces. Such live streaming data from Cactus simulations can be viewed using many different visualisation tools, including Amira [109][110], IBM Data Explorer [111] and LCA Vision [112].

2.5.4 GridLab

The EU GridLab project [113], led by Poznan Supercomputing Centre in Poland and finished in April 2005, aimed to allow the integration of applications with emerging Grid technologies, providing a Grid Application Toolkit (GAT) [114][115] that can be used to develop grid-aware applications without having to understand, or even being aware of the underlying technologies. The main goal of the project was to provide a software environment for grid-enabled scientific applications. GridLab provided a set of high-level GAT APIs through which applications access and use available resources.

The overall architecture of GridLab, see Figure 2.8, is set out in a layered environment providing an abstract view of the underlying infrastructure. This allows for the applications to access all of the capabilities of the lower level resources and services using the GAT API. The services available to applications included resource brokering, monitoring and data management, and Gridlab used two widely used application frameworks to help prototype the GAT interface, Cactus [105] and Triana [69][116]. Although collaborating with the developers of Cactus and Tirana for the development of GridLab, the project was designed to enable any application to run on the Grid.

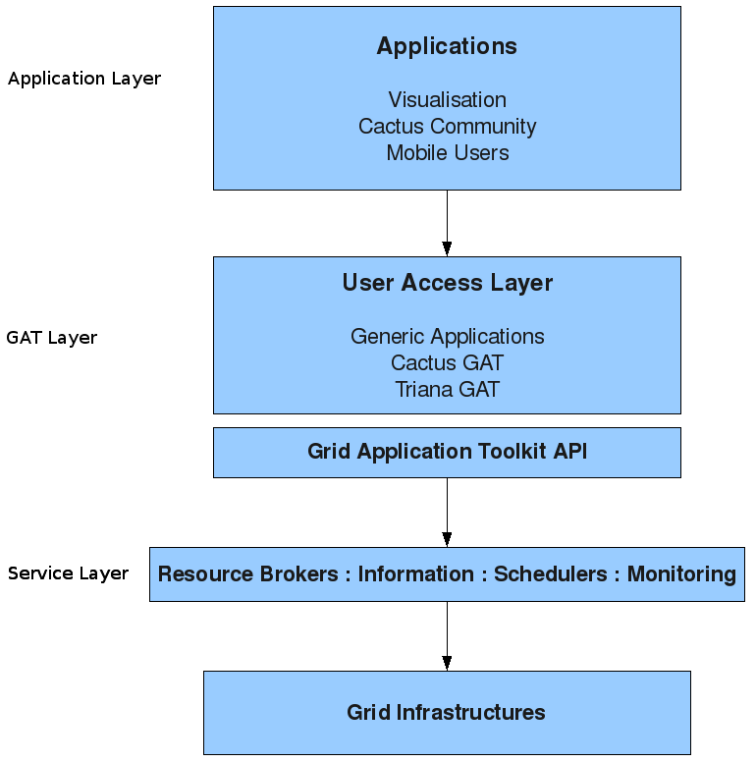


Figure 2.8: The general GridLab Architecture.

2.5.5 Image-Processing Grid Environment

The Image-Processing Grid Environment (IPGE) [117] is a project that started in 2004 ,supported by ChinaGrid [118], that aims at providing high performance image-processing platform in a grid computing environment. IPGE is a combination of grid resources and workflow techniques on which complex applications can be modelled as grid workflows with local or grid-enabled remote service components. Figure 2.9 shows the architecture which provides the integration platform for both techniques. It consists of several layered parts and the IPGE middleware is located between the basic grid middleware and the grid-enabled image processing applications. The Grid resources and grid middleware are the lowest layer and provide the computing power and generic interfaces to services such as meta-data management, service management and security. The service-based environment of IPGE is designed with an OGSA-compliant grid which makes it possible to model image processing as a workflow to enable collaborative processing for complex applications. The IPGE workflow engine classifies these services and then assembles them as complicated components. Users can then consult the engine on-line and customise the services to build their own compatible workflow.

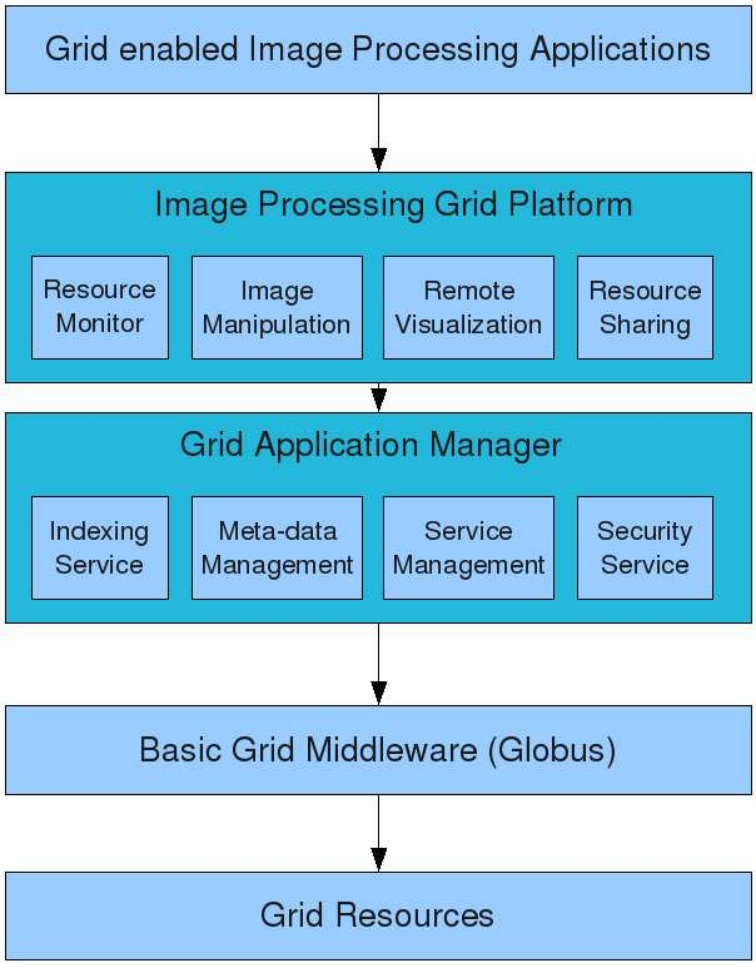


Figure 2.9: Image Processing Grid Architecture

2.5.6 Crossgrid/int.eu.grid Visualisation Technologies

A set of grid-enabled visualisation technologies were developed within the EU CrossGrid [7] and int.eu.grid [2] projects over the period 2002-2006, that are oriented towards compute and data-intensive applications that involve the interaction of a user in the processing loop. Such applications require a response from the Grid to an action by a human agent in different time scales. Tools developed within the CrossGrid project are glogin [119], GVid [120], Grid Visualisation Kernel (GVK) [121], and the Migrating Desktop [122]. These are briefly described here.

glogin: This tool provides a tunnel into the Grid and therefore facilitates interaction with the Grid's resources.

GVid: GVid allows rendering to be done on Grid resources, with transmission of resulting video over the Grid.

GVK: With GVK the user is able to control the execution of a grid application by installing a bi-directional interactive link between the scientific application and the

visualisation tool. The link itself is established using the glogin tool

Migrating Desktop: The Migrating Desktop is a framework for a graphical user interface for application management, grid and job monitoring, data and metadata management. This graphical environment is used as an advanced client for accessing grid resources in CrossGrid.

2.5.7 Limitations

In each of the examples above there is a problem in that the application developer's task is to write application-specific plugins that can communicate with the appropriate web services. Is there away around this problem? For every application a plugin is required. This plugin can be very specific to the type of simulation being run within the visualisation application. For widely used software, like VTK, it may be possible to have a range of plugins available, but it would be better if no extra software was needed on the application side.

Other interactive visualisation applications have used tools such glogin to tunnel into the Grid and run the applications on the Grid through this pseudo terminal. This indeed is running an interactive application on the Grid but there are two problems with this. Firstly, when using common middleware such as Globus [9], LCG2 [32] or EGEE [1] the application has to be installed on the "gatekeeper" that the user uses glogin to connect to. Secondly this application is running on the gatekeeper and not a general compute-node.

2.6 Interactive Visualisation applications on the Grid.

Interactive applications are characterised by interaction with a person in a processing loop. Each application requires a response from the Grid to an action by that person in different time scales: from real time through intermediate delays to long waiting periods. The applications are simultaneously compute- and data-intensive.

2.6.1 Crossgrid Flood Crisis Simulation

Grid-enabled simulations of three physical systems pertinent to flood crisis management: meteorology, hydrology and hydraulics. The main component of the system is a highly automated early warning system, based on hydro-meteorological (snowmelt) rainfall-runoff simulations. This application won the "best-demo" prize at the EGEE'05 User Forum and a example of the simulation's output can be seen in Figure 2.10.

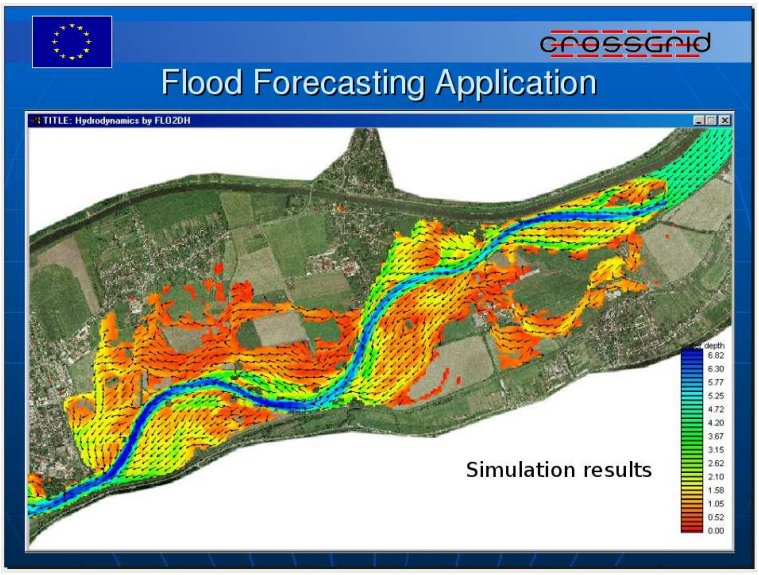


Figure 2.10: Crossgrid Flood Crisis Simulation. Image taken from the *Flood Crisis Team Decision Support System Presentation* [123][124]

2.6.2 Crossgrid Blood Flow Simulation

A Grid-based prototype system for pretreatment planning in vascular interventional and surgical procedures through real-time interactive simulation of vascular structure and flow. The system consists of a distributed real-time simulation environment, with which a user interacts in Virtual Reality (VR). A 3-d model of a patient's arteries, derived using medical imaging techniques, serves as input to the environment for blood flow calculations. An example of the simulation's output is shown in Figure 2.11.

To display the images a Virtual Radiology Explorer (VRE) environment is used. The aim of the VRE is to provide an end user with an intuitive virtual simulated environment to access medical image data, visualise it, and explore patient vascular condition. Average transfer times for the medical image data, once taking into account the Globus caching mechanism, did not vary much above 400 milliseconds for the smaller size files and no more than 850 milliseconds for the larger size files.

Security is, naturally, an important concern. The sensitive nature of clinical patient data, together with concerns that data and resources be made available in a timely fashion to just those who are authorised to access them, is supported by Grid authentication and authorisation components which span all aspects of the infrastructure. Maintaining confidentiality is important in the development of monitoring protocols and procedures. Therefore, in order to guarantee patient confidentiality, database access is limited and anonymised.

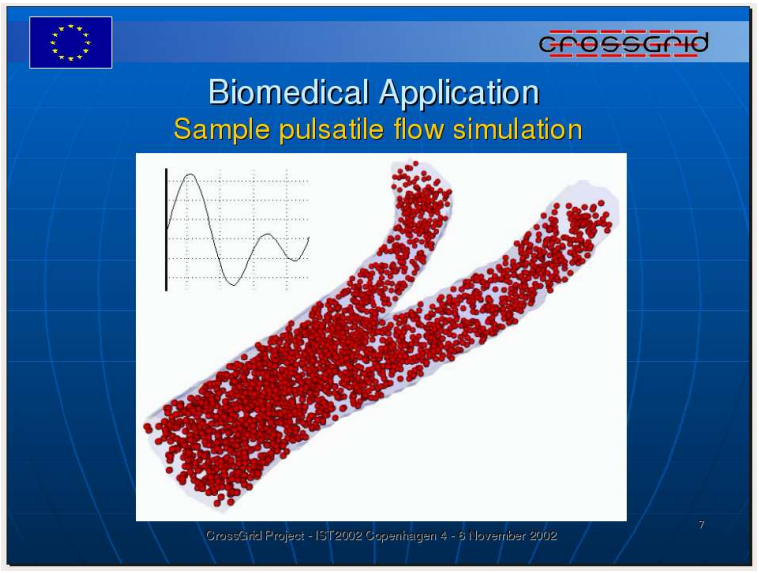


Figure 2.11: Crossgrid Blood Flow Simulation. Image taken from the *Surgery Decision Support Application Presentation* [125][126]

2.6.3 int.eu.grid Fusion Plasma Application

The Fusion Plasma application [127], which won the "best- demo" prize at the EGEE'07 User Forum, visualises the behaviour of plasma inside a Fusion Reactor and executions of this simulation are foreseen as a part of a so called Fusion Virtual Session. An example of the output of this program can be seen in Figure 2.12. The plasma is analysed as a many body system consisting of N particles where the N particles are distributed among a number of processors, which calculate the individual trajectories (independent). For every particle the position, velocity, etc. are stored in a binary file which is transmitted for visualisation purposes. This application allows for interaction with the simulation and the possibility of allocating more resources in runtime. It also allow for the changing of physical parameters in the simulation. As visualisation is an important part to this application, there is a graphical interface which visualises the plasma trajectory in the reactor. The minimum bandwidth required for this visualisation is 30 Mb/s.

2.6.4 RealityGrid TeraGyroid Simulation

TeraGyroid [129] used RealityGrid for computational steering over the Grid to study the self-assembly and dynamics of gyroid mesophases (found in novel materials and living cells) using the largest set of lattice Boltzmann simulations ever performed. SGS was adopted to provide the steering and VTK was used to provide a visual output of this simulation. Two different visualisation configurations were used:

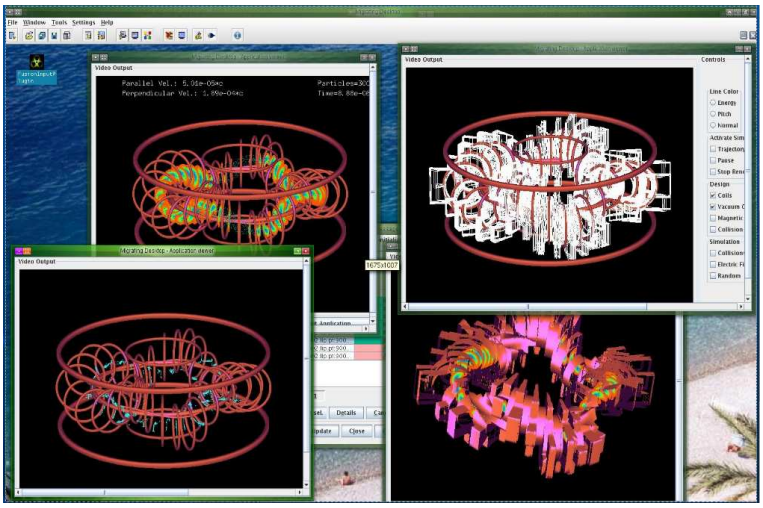


Figure 2.12: Different instances of the Fusion Plasma application running on the grid and visualised using MD. Image taken from *Fusion Simulations, data visualisation and future requirements for the interactive grid infrastructure* [128]

- Firstly the visualisation was run on a SGI Onyx system [130] and the graphical output was distributed to multiple locations using OpenGL Vizserver [131]. This allowed users to view the rendered output on machines with a low specification local to them. The visualisation output could also be streamed using the Access Grid software [101].
- The second configuration used the visualisation cluster at Argonne National Laboratory. This cluster is installed with Chromium [103] which allows the rendering load to be distributed across the machines. Each machine then fed their output to Access Grid for viewing on a tiled display.

Figure 2.13 shows sample output from the TeraGyroid simulation.

2.7 Visualising the Grid Itself

In this section we give an overview of existing 2-d and 3-d visualisation applications which are used in network traffic monitoring and intrusion detection systems, and also existing systems for visualising computing grids. This is a broad overview giving the advantages of using these applications and also their limitations.

2.7.1 Network Visualisation

Firstly a visualised distributed environment could be of three types: a simple physical network, a wireless network, or an infrastructure (such as the grid). Secondly visuali-

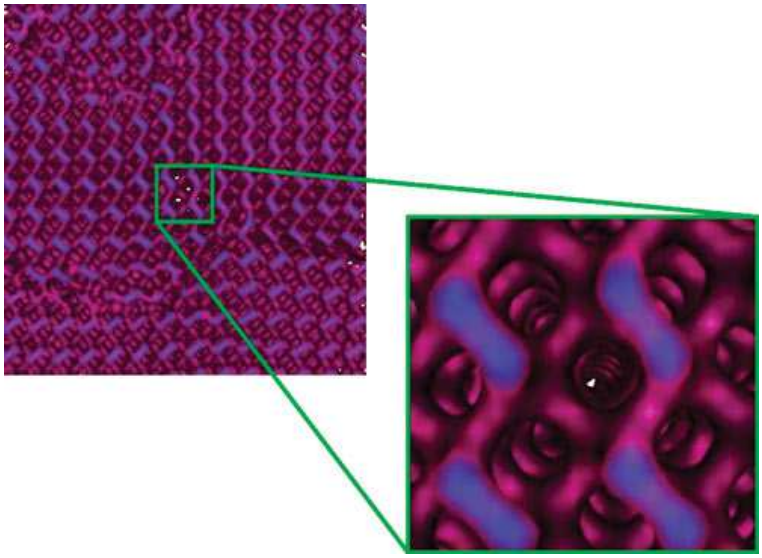


Figure 2.13: Gyroid domains with differing orientations and close up showing the regular crystalline gyroid structure within a domain. Image taken from *CCLRC Annual Report 2004-2005* [132]

sation of a distributed environment could be used for learning purposes, for monitoring the distributed environment, for management, or for the user-friendly development of applications within this distributed environment. Thirdly the medium of visualisation could range from a simple 2-d applet or desktop application to 3-d/VR/AR and rich-media immersive worlds.

2.7.2 2-d Network Visualisation applications

Etherape [133] is a graphical network monitor for Unix modeled after Etherman [134]. Featuring link layer, IP and TCP modes, it displays network activity graphically, as in Figure 2.14. Hosts and links change in size with traffic and the different protocols are colour coded. It supports Ethernet, FDDI, Token Ring, ISDN, PPP and SLIP devices and it can read traffic from a file as well as live from the network.

Etherape is well suited to monitoring a limited number of hosts but as the number of hosts increase it cannot efficiently display information about the network, with an effective display limit of 100 nodes. VISUAL [135], see Figures 2.15-2.18, focuses on communication between local networks and the outside world. This application is useful for networks with fewer than 12,500 nodes, where 2,500 are internal. It presents an efficient view of interactions between a monitored network and the outside hosts, but again in this application the number of external hosts that can be represented is limited and internal activity is not displayed.

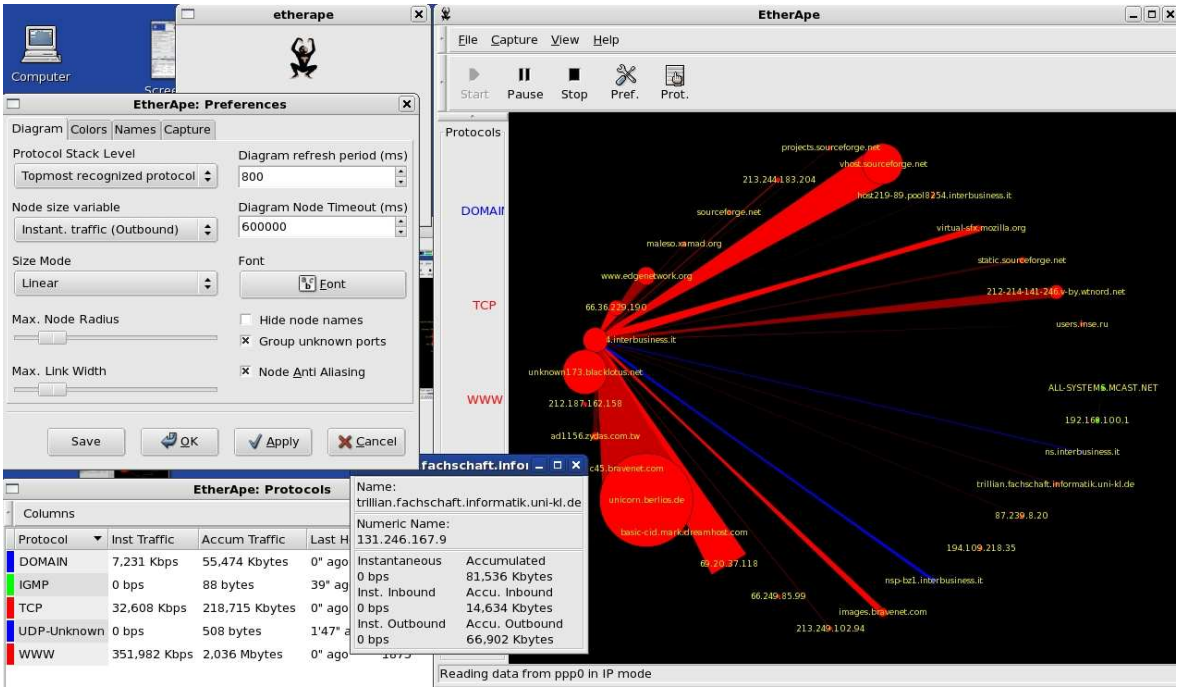


Figure 2.14: Etherape displaying network traffic. Image taken from *Etherape: A Graphical Network Monitor* [133]

2.7.3 3-d Network Visualisation applications

The 3-d Spinning Cube of Potential Doom [136] is an animated visual display of network traffic collected through the Bro Intrusion Detection System [137][138], see Figure 2.19. It displays the information within a 3-d cube which the user can spin at will. Port scans are easy to detect with this visual display, but after detection the exact origins of attacks are difficult to retrieve. Another 3-d application called the Intrusion Detection Toolkit (IDtk) [139], see Figure 2.21 processes either raw TCP network traffic or intrusion detection system alerts. This toolkit is based on glyphs and the attributes of these are used to represent components of the input data. The components can be encoded in 3-d coordinates, colour, size and/or the shape of the glyphs, with the user being free to customise the visualisation zone. This design is limited as it only permits detection of relationships between elements of the data, and the interactions between the hosts are not intuitively displayed. Le Malecot et al [140], see Figure 2.20, describe an attempt to interactively combine 2-d and 3-d visual representations of network traffic as a hierarchical representation of network space based on interactive 16x16 and 256x256 grids. This allows users to access and visualise an activity of any network ranging from a single host to a global network.

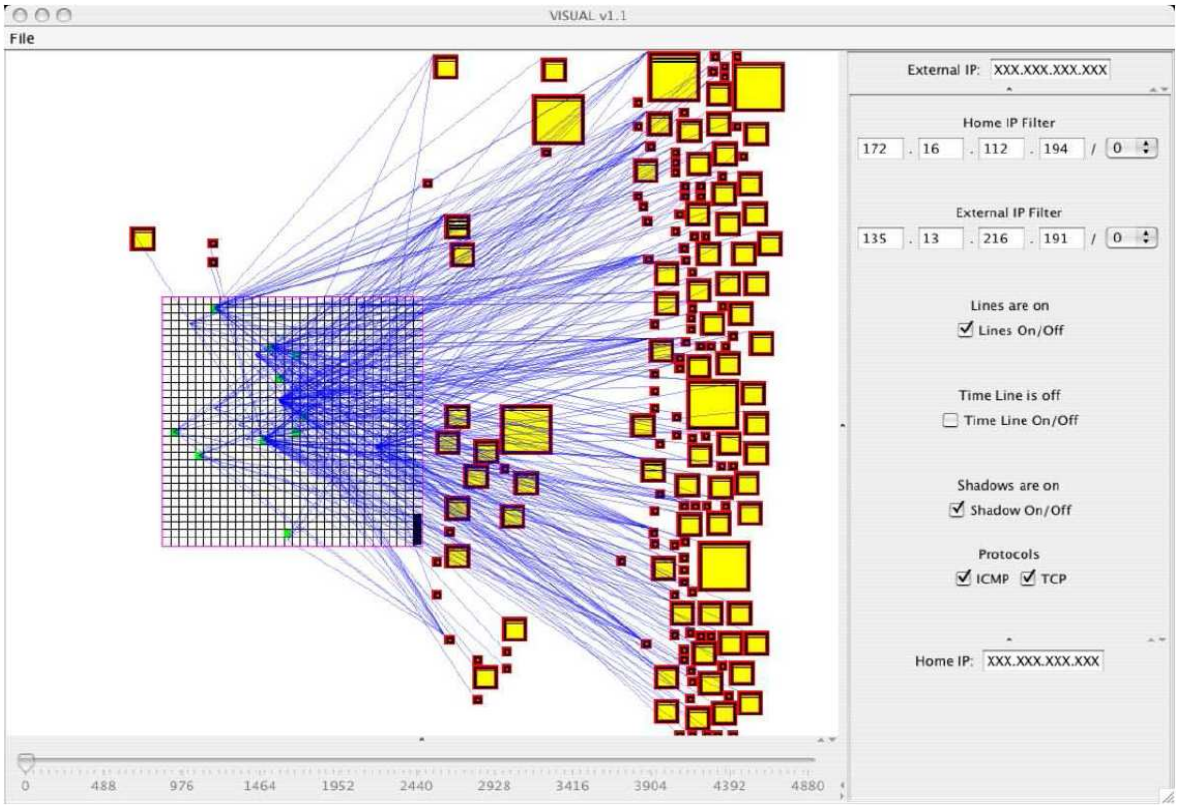


Figure 2.15: VISUAL: A 2D visualisation of 80 hours of network data on a home network of 1020 hosts. Image taken from *Home-centric visualisation of network traffic for security administration* [135]

2.7.4 2-d and 3-d applications for Visualising Grids

There are some applications which visualise the grid and its activity in either 2-d or 3-d, but one application, *Real Time Monitor* [141], can display grid information in both 2-d and 3-d. Its 2-d interface, see Figure 2.22, is a Java applet which visualises the jobs currently running and displays all the jobs that are submitted through the resource brokers which it displays. These resource brokers are being monitored directly through a mySQL database connection. Since this information is continually updated, job movement around the world map is seen in real time, along with changes in status. The application displays all jobs currently submitted to LCG2 via the resource brokers monitored, with jobs having finished but not being cleared automatically ceasing to be monitored after 2 hours.

The Real Time Monitor also has a 3-d visualisation application which uses satellite imagery from NASA and displays running and scheduled jobs, job transfers and detailed information on resource brokers and computing elements for each site on the Grid. The program allows you to move around the globe, zoom in on a location, or isolate a particular virtual organisation or resource broker.

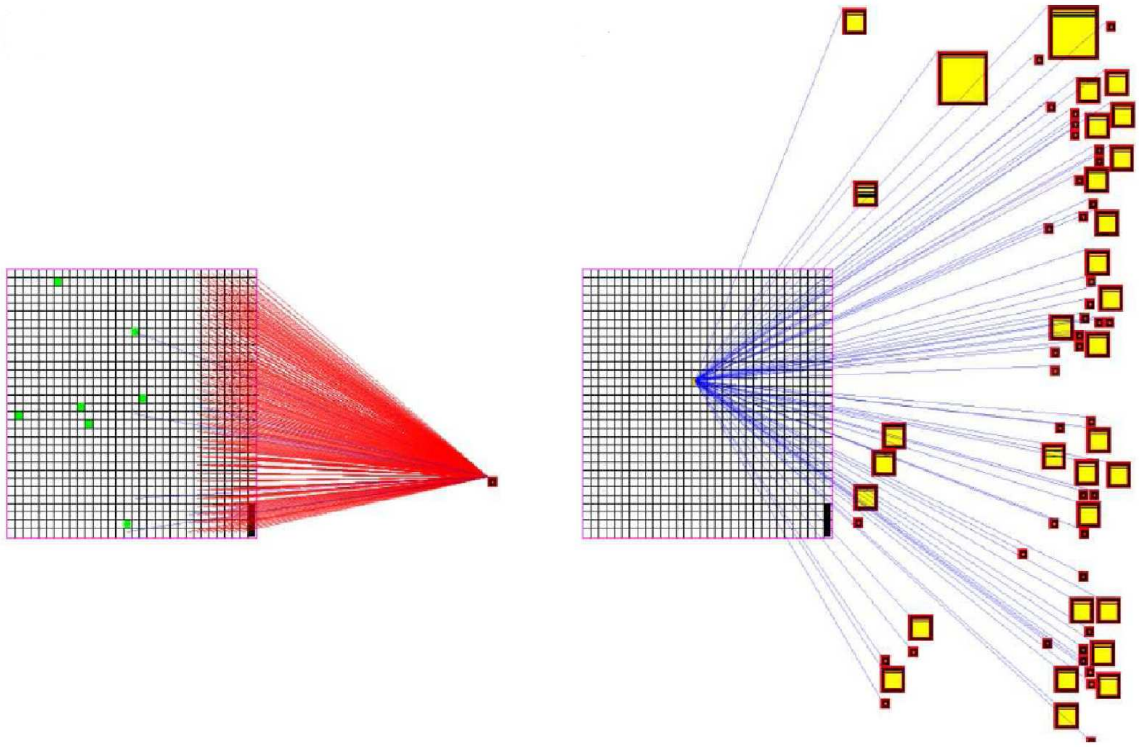


Figure 2.16: VISUAL: A 2D visualisation of 80 hours of network data on a home network of 1020 hosts. Image taken from *Home-centric visualisation of network traffic for security administration* [135]

Dr. Stuart Kenny of the author's host research group has utilised this 3-d Real Time Monitor for displaying the topology of network intrusion alerts as part of the Active Security Infrastructure [142], see Figure 2.23.

A second application which visualises grids has been developed by Dr. Keith Rochford, also from the author's host research group, and is part of the I4C grid service monitoring system [143]. This application is in 2-d and is based on the presentation tool from the Nagios [144] host and network monitoring system. A snapshot example of this application is shown in Figure 2.24. Finally, the GridICE grid monitoring tool incorporates a 2-d geographical monitor. This monitor is a basic 2-d map of Europe and Asia displaying the location of grid sites within the framework of the EGEE project.

A third application, written by the author, incorporates a more complex virtual adaptive grid visualisation, and is described in Chapter 3.

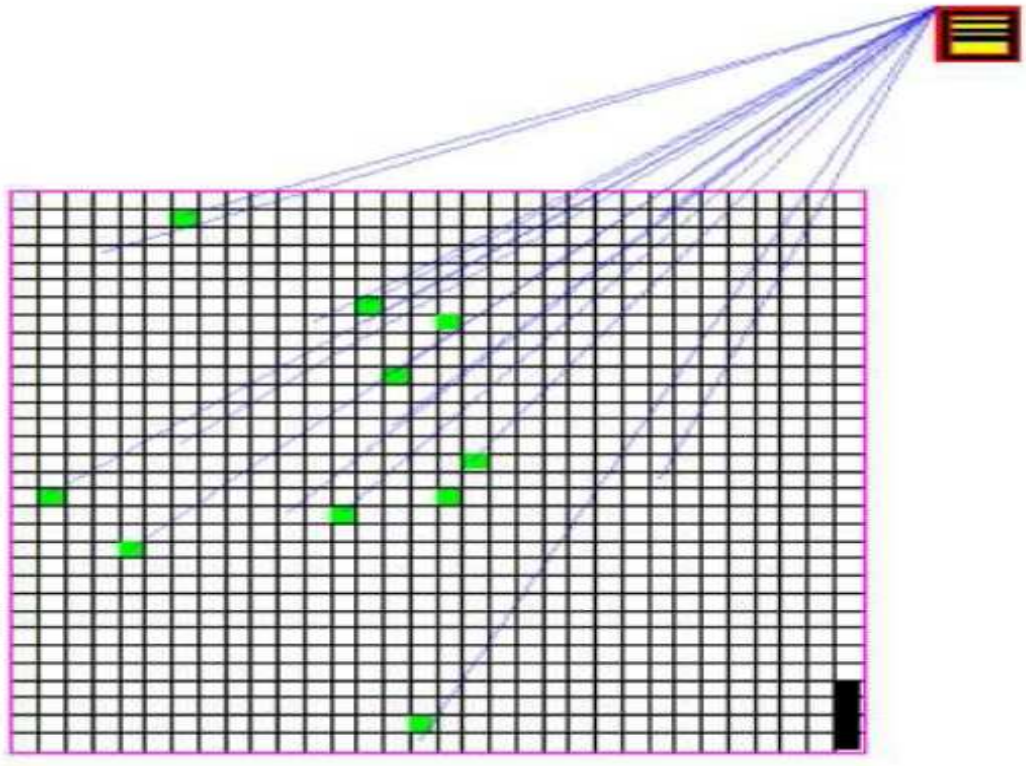


Figure 2.17: VISUAL: A 2D visualisation of 80 hours of network data on a home network of 1020 hosts. Image taken from *Home-centric visualisation of network traffic for security administration* [135]

2.8 Summary

This chapter has covered the current research into distributed computing, workflows, real-time visualisation and interactive visualisation on the Grid. It is evident from this overview that there are numerous visualisation resources available to the Grid user, both with interactive capabilities and computational steering. However, with the increase in choices for the grid user, comes the problem of appropriate or best-fit resource selection. This thesis will discuss possible mathematical models for determining visualisation resource selection and provide a model that will aid the user in this decision based on the benchmarking and analysis of the available visualisation resources of the Grid.

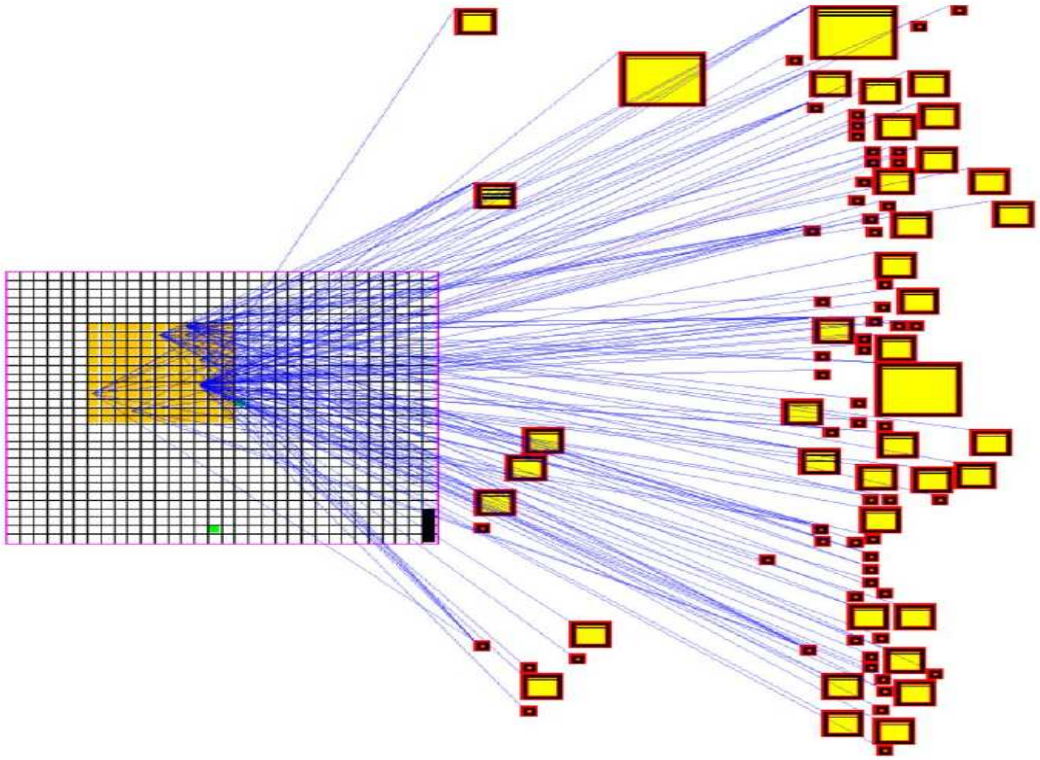


Figure 2.18: VISUAL: A 2D visualisation of 80 hours of network data on a home network of 1020 hosts. Image taken from *Home-centric visualisation of network traffic for security administration* [135]

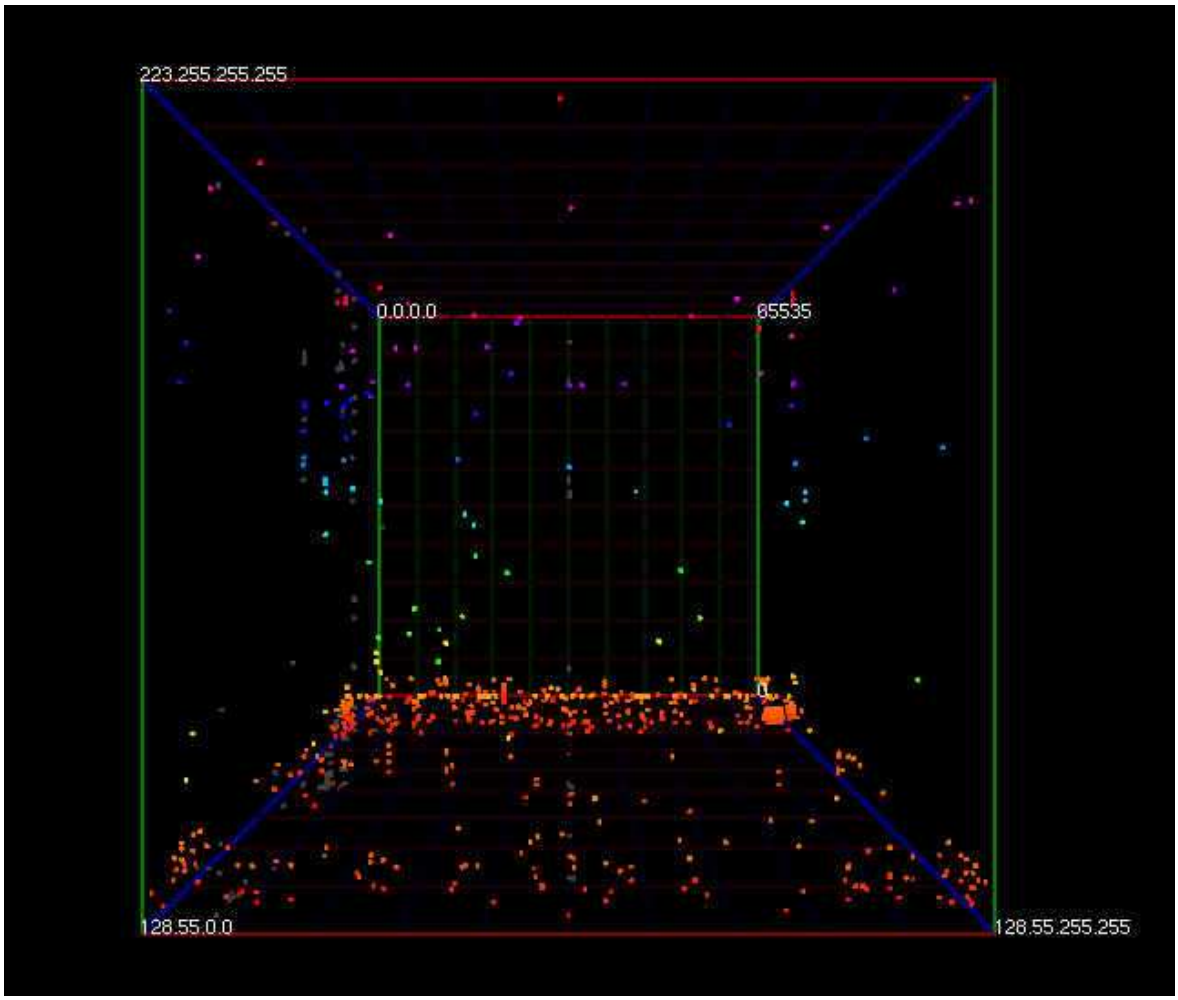
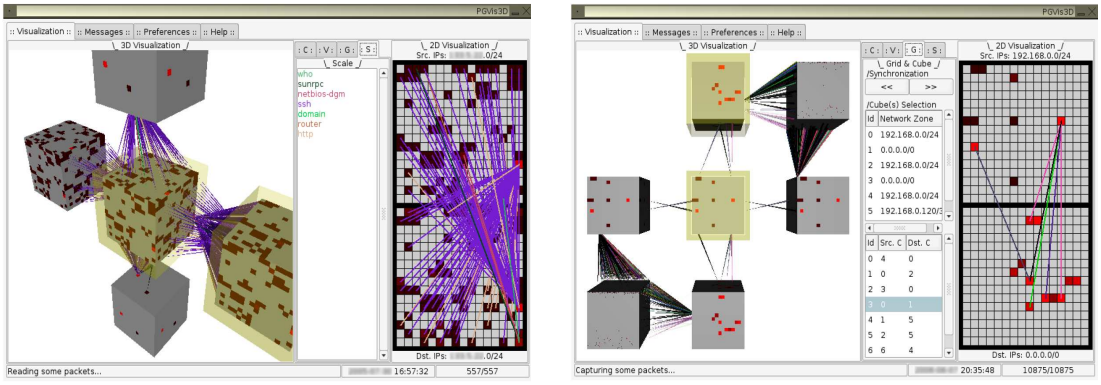


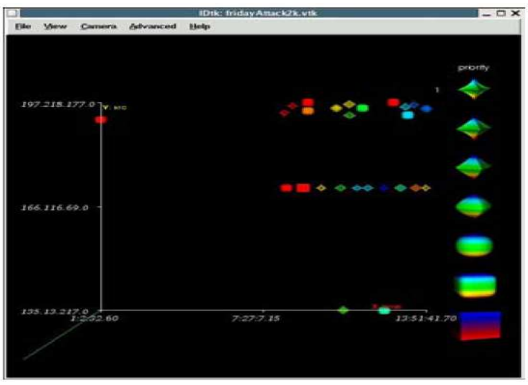
Figure 2.19: The Spinning Cube of Potential Doom displaying port activity. Image taken from *Interactively combining 2D and 3D visualisation for network traffic monitoring* [136]



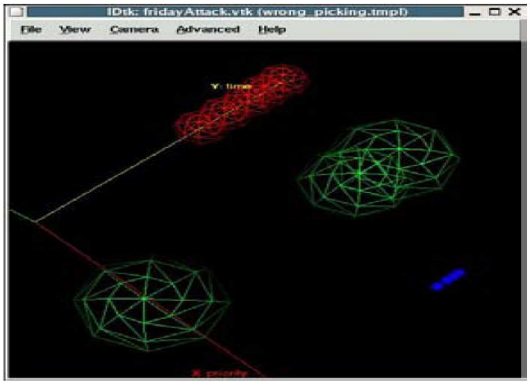
(a) A network scan from an internal host.

(b) Display of portscan of a local host.

Figure 2.20: Image taken from *Interactively combining 2D and 3D visualisation for network traffic monitoring* [140]



(a) Visualisation with mappings: time - x-axis, source ip - y-axis, ip datalen - colour, ip datalen - size



(b) Wireframe visualisation

Figure 2.21: IDtk visualisation of network traffic. Image taken from *A User-centered Look at Glyph-based Security Visualisation* [139]

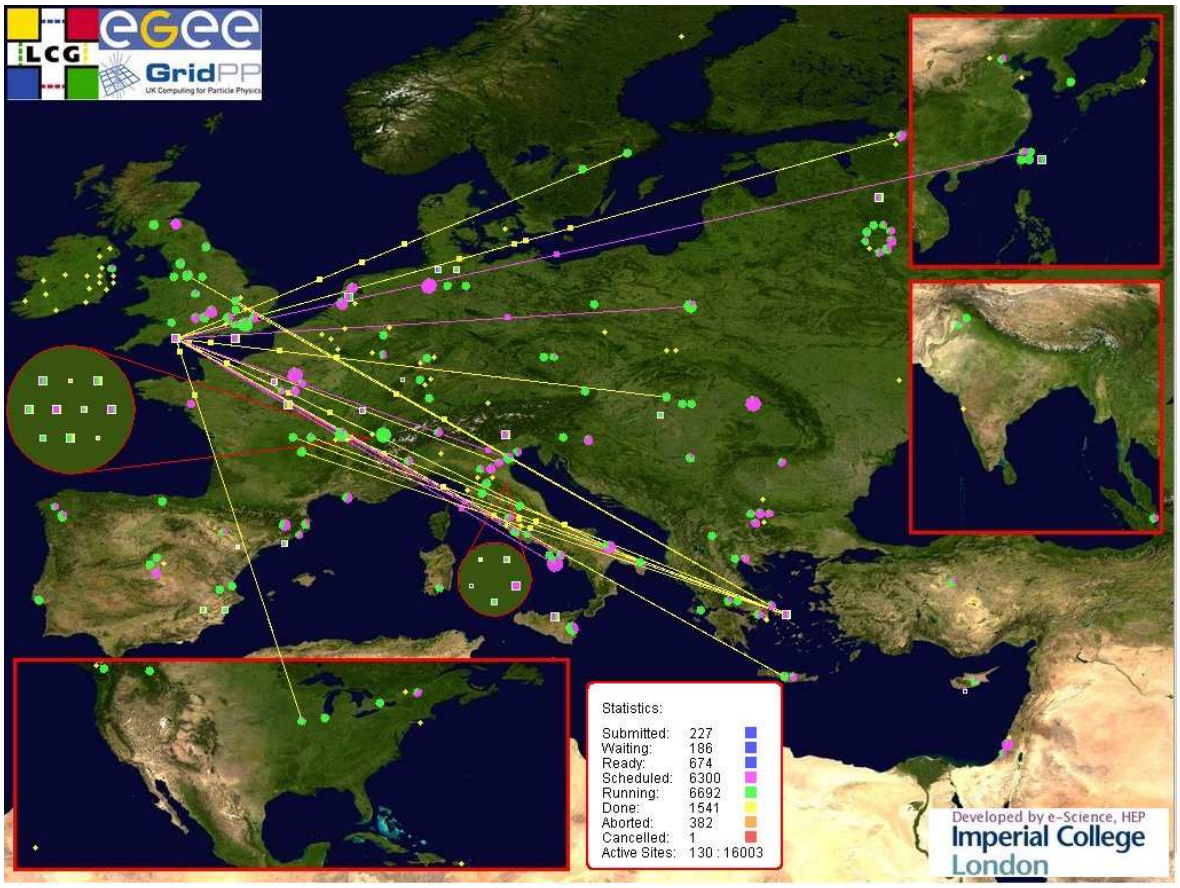


Figure 2.22: The 2-d Real Time Monitor application. Image take from *Real Time Monitor* [141]

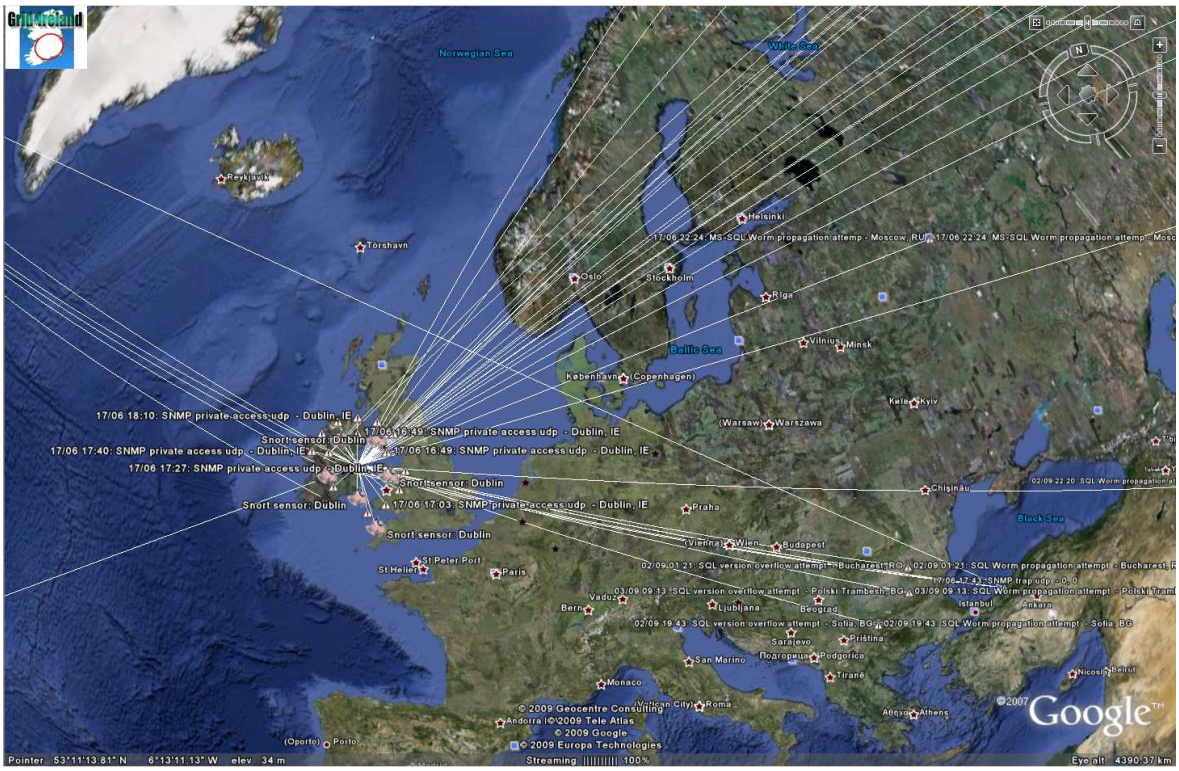


Figure 2.23: Real Time Monitor from the Active Security Infrastructure. Image courtesy of Dr. Stuart Kenny of the author's host research group

Figure 2.24: Example Nagios display derived from I4C. Image take from *An Agent-Based Approach to Grid Service Monitoring* [143], courtesy of Dr. Keith Rochford.

Chapter 3

Visualisation Framework for a Grid

3.1 Introduction

The potential to move from a test bed to a production infrastructure is critical to the success of the grid. This can be achieved when the Grid infrastructure is robust enough to support various compute and data intensive application domains. Moving towards a productivity phase will put the Grid in a leading position with respect to its peer technologies and infrastructures. User interfaces, including visualisation, will play a major role in this transition.

Here the author describes a principal output of this thesis, a framework for multiscale multimodal grid visualisation that has the potential to assist the penetration of the Grid into the domain of advanced interactive 3-d visualisation and geographical rendering. While the existing tools and techniques for interactive visualisation are of a general purpose nature and offer limited compute and data intensive graphical visualisation and interactivity, this framework leverages the power of the grid to offer: 1) more advanced visualisation 2) real time interactivity with the rendered content; 3) and integration with key data intensive graphical applications that would benefit from the use of the grid.

3.2 Multiscale Multimodal Grid Visualisation Framework

In *Integrating a Common Visualisation Service into a Metagrid* (see Appendix A), the author of this thesis presented the visualisation architecture of the framework and argued that it addresses the major limitations of existing grid visualisation solutions. These limitations were identified as lack of multimodality, interactivity, and timeli-

ness, as well as the high re-engineering cost for grid-enabling visual applications. The Multimodal Grid Visualisation framework utilises the grid visualisation architecture described in [5], which couples grid resources for high end visualisation. The grid resources developed so far within this framework include a time-shared Visualisation Engine, coarse and mid-scale simulation resources, a fine-scale rendering resource and interactivity resources. These grid resources, see Figure 3.1 are described in the following sub-sections. They have been implemented for a Linux platform.

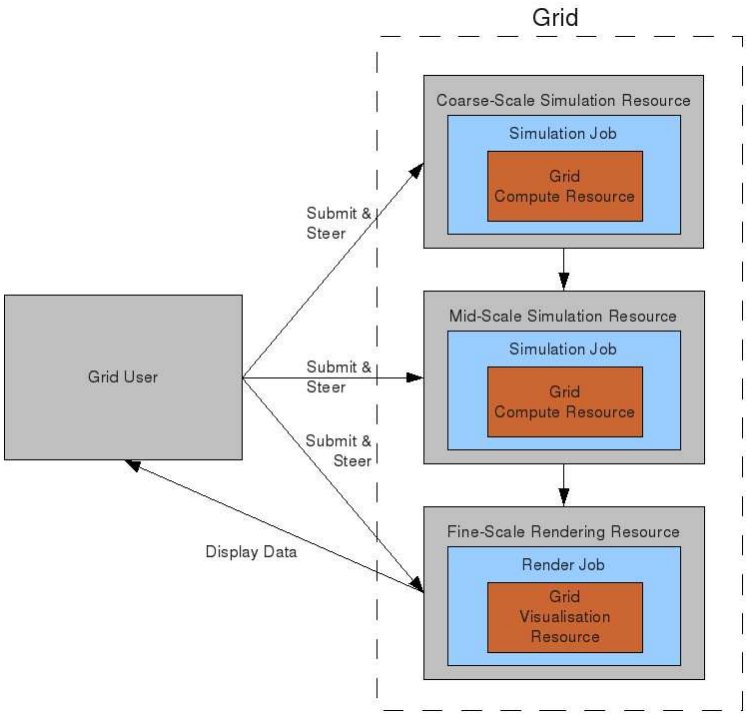


Figure 3.1: Diagram showing the coarse-scale, mid-scale and fine-scale division of grid resources.

3.2.1 The Coarse-Scale Simulation Resource example

This resource is described in detail later in this chapter. It integrates Grid and non-Grid tools and functionalities to extract, in real time, geographical information about the existing global grid infrastructure. Grid tools integrated in this coarse-scale simulation resource are *Site Functional Tests* (SFTs) [145][1], the *Grid Information System* and *Google Earth*.

Google Earth [148] is a free-of-charge, downloadable virtual globe program, a non-Grid tool, which provides a way of positioning placemarks onto a virtual earth model.

[1]It is interesting to note that the successor to SFTs, the Service Availability Monitoring tests (SAMs) [146] are evolving into the core status information source for the European Grid Infrastructure (EGI) [42], using the Apache MQ messaging system [147], so this approach will have longevity.

It runs on most platforms including Linux. Given its stability and wide usability, it is used in the simulation to give 3-d geographical information about the grid sites. This simulation also contains various parsers and file servers developed by the author to suit specific applications developed within the multimodal visualisation framework.

The simulation integrates placemarks, 3-d buildings or other features of Google Earth by creating a file with the required information stored in a special format called Keyhole Markup Language (KML), an XML-like grammar that is used for modelling and storing geographic features (points, lines, images, polygons).

3.2.2 The Mid-Scale Simulation Resource example

In a similar way, a mid-scale simulation resource integrates Grid and non-Grid tools and functionalities to add multimodal patterns of navigation to a geographical content. At present it utilises just one tool, *OGRE* [149][150].

Ogre is an Object-Oriented Graphics Rendering Engine. It is a scene-oriented, 3-d engine, written in C++, that is designed to produce applications utilising hardware-accelerated 3-d graphics. It is supported by a strong research community. The Ogre API abstracts all the details of using the underlying system libraries of OpenGL and provides an interface based on world objects and other intuitive class supports. It also has exporter plugins for most common modelling software.

A VirtualGrid application has been developed by the author, using Ogre, which allows for loading of a .vge file containing site-specific information gathered from grid information systems. This .vge file format was also developed by the author. These files are created by the 3d interaction resource and are stored on a Virtual Grid Engine (VGE) server.

3.2.3 The Fine-Scale Rendering Resource

Chromium [103] is a system for manipulating streams of OpenGL graphics commands on clusters of workstations. Chromium's stream filters can be arranged to create sort-first and sort-last parallel graphics architectures that, in many cases, support the same applications while using only commodity graphics accelerators. Sort-first or tiled rendering is where the frame buffer is subdivided into rectangular tiles which may be rendered in parallel by the nodes of a rendering cluster. Sort-last or Z-compositing rendering is where the 3d dataset is broken into N parts which are rendered in parallel by N processors. The resulting images are composited together according to their Z buffers to form the final image. In addition, these stream filters can be extended programmatically, allowing the user to customise the stream transformations performed

by nodes in a cluster.

3.2.4 The Interaction Resources

For the presentation to the user (i.e. the display) it is important that streams of rendered images be conveyed from the visualisation engine to the (remote) user via widely used protocols that have good supporting display software. The Crossgrid/int.eu.grid GVid [120] streaming protocols have been adopted for this purpose. The intent is also to use Access Grid [101] streaming protocols as an alternative, as this has the distinct advantage that the visualisation output stream can be incorporated into Access Grid video conferences. Thus the streams of rendered images are transferred by the GVid or Access Grid unidirectional datapaths.

For the control by the user, both GVid and Access Grid have control paths that can be used to steer the simulation and rendering resources from the user's workstation.

3.3 The Visual Pipeline

In developing grid-enabled visualisation solutions, the grid community has attempted to break the description of a visual application into a visual pipeline in order to facilitate implementation and frame the description of the application domain. Here, the visual pipeline described by the author in [5] as computation, rendering, display and interaction will be referred to in order to convey the experimental configuration and the implementation of a use case application developed within the multimodal framework. The use case is an application that provides a real time view of the worldwide grid infrastructure, described in more detail later in this chapter. The following sub-sections describe each stage of the visual pipeline corresponding to the use case.

3.3.1 Computation

In the context of the use case, computation refers to data collection (gathering) and manipulation of information about the global grid infrastructure. This computation is in turn subdivided into a sequence of tasks including:

- running site functional tests [145] and using the get-GOC-sites-map command to extract information about active sites and their corresponding gridgates[2] The

[2]In the Grid-Ireland context, for gLite middleware a gridgate is a computing element (CE), a storage element (SE), a user interface (UI) and a local resource management system (LRMS). For GT4 middleware a gridgate refers to a Globus gatekeeper and a LRMS.

use of these tests to identify active sites ensures near real-time information is obtained.

- parsing the results of site functional tests to identify the gridgates of various grid sites around the world.
- using the grid information system (LDAP search) [151] to query for information about the gridgates.
- parsing the results of the ldap queries to extract longitude and latitude geographical coordinates of gridgates of various grid sites around the world.
- using the longitude and latitude information about gridgates to construct a KML file describing geographical locations of various gridgates.
- adding generic site building models and placemark locations to the KML file.
- uploading the KML file to the KML file server communicating with the Google Earth rendering on the coarse-scale simulation resource.

These coarse-scale simulation tasks are gathered together to form one grid job which is submitted to the Grid. In Figure 3.2 labels 1-4 show the path of this job from the user's terminal to the worker node where the job is executed. The CE/WN are not specified, therefore this job may run anywhere on the Grid.

There are no data limitations to these simulations as all the information gathered describing a grid site, from coarse-scale to fine-scale, is less than one megabyte of storage at present. This information, however, is only a snapshot of the Grid at the execution time of these information gathering grid jobs. A limit or expiration date may be required on the information collected so that grid storage resources are not overloaded.

The mid-scale simulation resource is used to associate 3-d navigable worlds (.vge files) to the geographical description of the various locations of active gridgates around the world. A VGE server is used to continuously host the .vge files, where these files are created using one or more grid jobs, shown traversing the paths 5-7 on Figure 3.2. These CEs/WNs are also not pre-selected and so the jobs may run on any grid sites. Once at these sites they gather information specific to that site that will be explorable by using the Virtual Grid Engine.

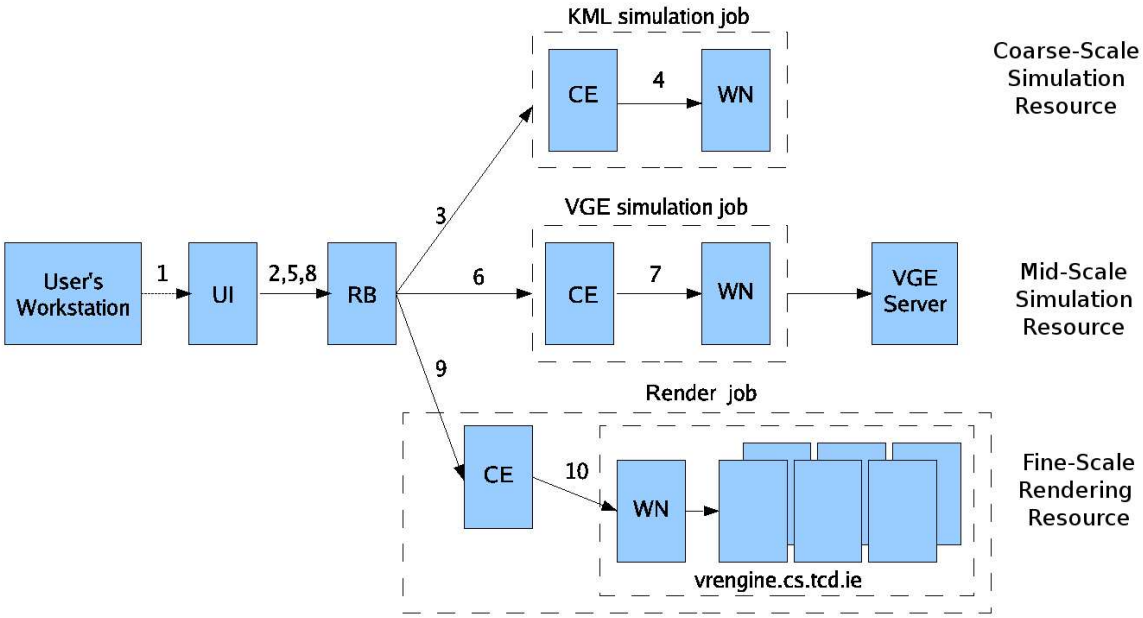


Figure 3.2: The Visual Pipeline of the Multiscale Multimodal Grid Visualisation Application .

3.3.2 Rendering

The KML file server together with the VGE file server communicates with the fine-scale rendering resource (the Visualisation Engine) to supply the necessary information for interactive real time rendering. The rendering is performed by Chromium, see Section 3.2.3, again shown traversing paths 8-10 on Figure 3.2.

3.3.3 Display

The interaction resources use either GVid [120] or Access Grid [101] data paths to transfer the visualisation stream to the user's terminal, shown as path 16 on Figure 3.3. Only an efficient and highly compressed video stream is transferred through the network and displayed on one or more thin video clients.

3.3.4 Interaction

Interaction takes place on a fine-scale by steering the Chromium process as shown on Figure 3.3 (17), on a mid-scale by steering the VGE simulation as shown on Figure 3.3 (18) and on a coarse-scale by steering the Google Earth simulation as shown on Figure 3.3 (19). The multi-scale steering can be considered as a form of hierarchical

steering that limits the fine scale to within a locality of reference to allow significant data caching. The steering is done using the interaction resources (GVid or Access Grid) controlpaths, from the user's workstation.

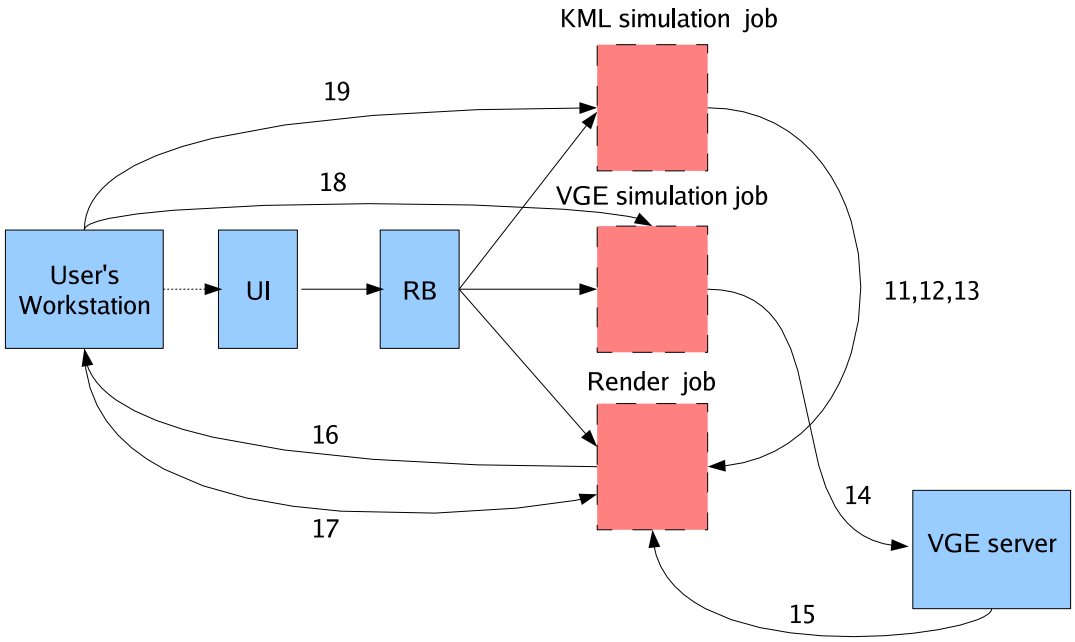


Figure 3.3: Other actions of the Multiscale Multimodal Grid Visualisation Application.

3.3.5 Other Activities

Figure 3.2 traces the flow of three jobs through the framework, each handling a stage in the visual pipeline. These jobs are the KML simulation job (2-4), the VGE simulation job (5-7), and the rendering job (8-10).

Synchronisation between the jobs corresponding to the different stages of the visual pipeline is handled via lock files, where these actions can be seen in Figure 3.3 (11,13).

The other actions in the visual pipeline are as follows:

- Transfer of the KML file from the KML simulation job to the render job (12)
- Transfer of .vge files to the VGE server (14)
- Fetching of .vge files by the render job from VGE server (15)

- Interactive streaming of the render job to the user's workstation via GVid or Access Grid streaming protocols (16)
- Steering of rendering and simulations (17-19)

These actions will be somewhat similar for other use cases, as they involve synchronisation, the flow of simulation results, and computational steering on coarse, mid and fine scales.

3.4 VirtualGrid

The author has developed a concrete example of the use of the visualisation framework of Section 3.2 to visualise the Grid itself; an application called *VirtualGrid* [8]. This allows visual exploration of a world-wide grid infrastructure for learning, monitoring, management, and application development. In this thesis it is used as a concrete test vehicle and a target for benchmarking. The visualisation of distributed environments like this is a highly challenging task. The main challenges to be addressed are complexity, scalability, and relevance to the user. The latter motivates a bigger research agenda of profiling the user (his expectations and taste) and adapting the visualisation to the user profile.

This visualisation of the grid consists of developing an environment abstracting elements of the EGEE[1] grid infrastructure using 3-d animated metaphors coping with the problems of complexity, scalability and adaptability to the user profile. Using information gathered from gLite[30] Berkeley Database Information Indexes (BDIIs[152]) to create a navigable world that contains representations of Compute Elements, Storage Elements, Worker Nodes, etc, a navigable world has been developed using Ogre[149].

Since users have differing levels of knowledge about grid infrastructures and their components, the aim is to take into account the user profile in adapting the navigable world. At this initial stage it is a very simple user profile consisting of the user taste in navigation, their expected level of complexity of the visualisation, and their technical background. The user profile is selected by the user at start-up from two pre-configured profiles. A profile for a novice grid user, and another profile for an advanced grid user. The navigable world is adapted according to the users profile. In response to the user profile, varying patterns of interaction are offered with the navigable world, different geometric metaphors and animations of grid elements suitable to the user taste, and a level of complexity adapted to the technical background of the user.

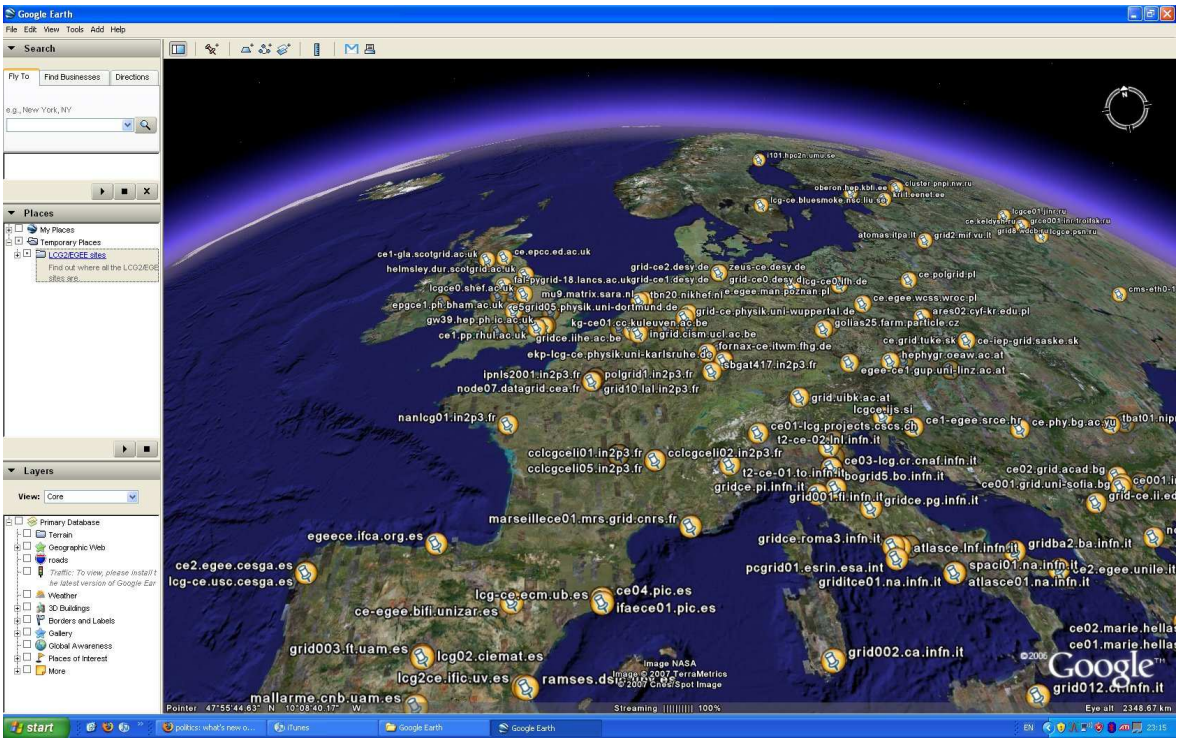


Figure 3.4: Google Earth visualisation of active gLite sites created by the author.

3.4.1 Multimodal Visualisation

Figure 3.4 is a snapshot of an adapted visualisation, showing multiple sites located across Europe, created by the framework described in Section 3.2. This adaptation is for a user who wants a basic overview of the geographical locations of functional gLite sites. Figures 3.5 and 3.6 are snapshots of the immersive grid world, which show the visualisation of a single grid site and multiple grid sites respectively, i.e. visualisations of the grid at a finer scale. The user in this case prefers to know more about the architecture of the grid infrastructure, as compute nodes and their resources are visually represented using differing orbital speed, colours and distances from a central node. Currently these multimodal adaptations are statically selected from the evaluation of a *.vge_profile* configuration file, but the work of Maad et al[153] shows how this could be defined in a more generic fashion such as with ClassAds[154], and selected according to more dynamic criteria such as market-driven economic models.

3.4.2 VirtualGrid Simulation Architecture

Figure 3.7 shows the complete multiscale multimodal grid visualisation framework: the *.kml* file generation and Google Earth application composing the coarse-scale simulation resource; the *.vge* file generation and immersive world making up the mid-

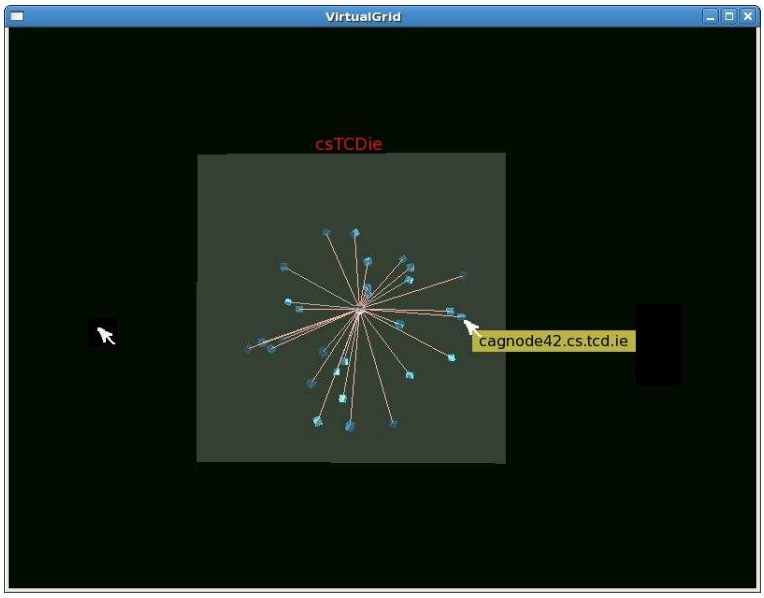


Figure 3.5: Selected active gLite sites in an immersive grid world created by the author.

scale simulation resource; and finally, the fine-scale simulation resource, composed of Chromium and VirtualGL.

.

3.4.3 Mid-scale Simulation Architecture

Ogre [149], the Object-Oriented Rendering Engine, is an open source rendering engine which is used as the basis for the VirtualGrid mid-scale simulation that has been developed. This engine is solely a rendering engine, not a game engine, and therefore does not contain any collision detection and only some basic level-of-detail techniques. Ogre is implemented in C++ and is aimed at making it easier and more intuitive for users to produce applications that utilise the 3-d acceleration available on graphics hardware. It is relatively simple to get Ogre and its example programs up and running and its API then allows users to add their own functionality. The mid-scale simulation has been implemented using the Ogre API to create the immersive grid world, taking advantage of the simple creation, organisation and control of 3-d objects in Ogre. Grid resources like CEs, WNs and RBs have been modelled in C++ and mapped to 3-d objects. Figure 3.7 shows the basic flow of the mid-scale simulation, where XML input files are read in, turned into OpenGL models, and the output produced is OpenGL rendering commands for images of these models.

The selection then of which models to use is made by the mid-scale simulation after parsing the VirtualGrid .vge XML file to examine the description of the grid from the information gathering by the grid jobs.

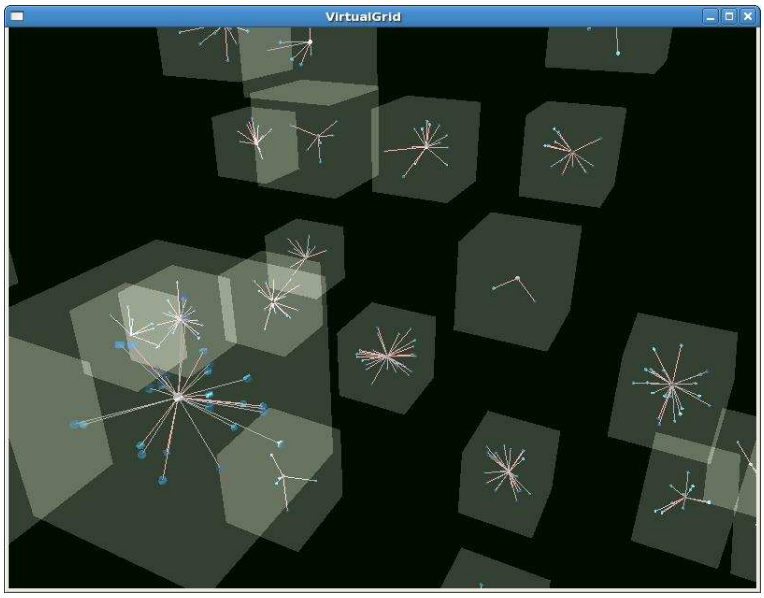


Figure 3.6: Multiple active gLite sites in the immersive grid world.

VirtualGrid XML File Type

This section describes the VirtualGrid .vge filetype, a custom format defined by the author, which describes a grid in XML. The Xerces-C++ validating XML parser, which is written in a portable subset of C++ DOM, has been utilised by the author of this thesis to help parse the .vge file and create C++ objects. The layout of the file is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<vgworld>
        <sites>
                <site siteName = "gridgate.cs.tcd.ie">
                        <ce>cpu_load =
                            uptime =
                            disk_size=
                            disk_util=
                            mem_size=
                            mem_util=
                            network_stats=
                            process_list=
                        </ce>
                        <se>
                        ...
                        </se>
                        <rb>
                        ...
                        </rb>
                </site>
        </sites>
</vgworld>
```

A full example .vge file can be viewed in Appendix C.1.

3.4.4 VirtualGrid Run-Time Operation

For debugging and informational purposes it is desirable for the system to have a means of supplying information to the user. In this case consoles are made use of to relay system information. A normal text console window is created at run-time by the VirtualGrid client program to which all standard error and print statements are redirected. Alternatively logfiles can be generated, especially in the case of debugging system errors. A system class handles the creation and writing to the log file. A log is kept of important system events during the initialisation phase of the application to help diagnose the cause of problems that can prevent the system from starting correctly. Less serious run-time errors can usually be handled and diagnosed by using the console classes. The base hardware requirements (independent of rendering API) for running VirtualGrid are:

- Nvidia: Geforce2 or higher required, Geforce 4(non-mx) or higher recommended
- ATI: Radeon 7500 or higher required, Radeon 9600 or higher recommended
- Silicon Integrated Systems (SiS), Intel and S3 cards might or might not be supported.

The use of the latest drivers provided by the graphics card vendor is always advisable. An OpenGL driver exposes a certain core version and a set of extensions. The OpenGL core version defines the set of base capabilities, while the extensions supply external features that can be used by applications and games to improve graphical quality or increase performance. To work at all, OGRE requires a base OpenGL version of 1.2.1. This is a very relaxed requirement as most graphics adapters, from the Nvidia TNT2 series and the ATI radeon series, are able to provide this. In addition to that, there is a long list of extensions that Ogre can and will use if they are available. These are listed in Appendix D. With regards to portability issues, VirtualGrid has been developed and is deployed on the X Window system (X11) using the Eclipse development environment. The development language was standard C++, with very few platform-specific portions. While development was done on a Linux machine, there are no reasons to prevent the porting of VirtualGrid to other architectures. Data acquisition for the construction of the model is gained through submission of infomation-gathering gLite jobs. These jobs are sent to all available grid sites and they collect site, node and machine specific data which is then converted to an XML file for input to the VirtualGrid application.

3.4.5 Model Construction

Models are created for Ogre in many ways but must eventually be represented by .mesh and .material files, which are the Ogre object and material filetypes. Ogre is then able to read each object and material and produce a 3-d rendering of it. Many 3-d modelling software tools will export their model files into .mesh and .material files, and one of those tools is Blender. Blender is an open-source 3-d modelling package which was used as the primary off-line model creation tool for VirtualGrid. The Blender exporter, see Figure 3.8, supports full Ogre mesh, material and animation exporting, and is kept in synchronism with the Blender versions as they are released.

Ogre Mesh Object Files

This class holds the data used to represent a discrete 3-d object. Mesh data usually contains more than just vertices and triangle information; it also includes references to materials (and the faces which use them), level-of-detail reduction information, convex hull definition, skeleton/bones information, keyframe animation etc. However, it is important to note the emphasis on the word 'discrete' here. This class does not cover the large-scale sprawling geometry found in level/landscape data.

Multiple world objects can be created from a single mesh object. The mesh object will have it's own default material properties, but potentially each world instance may wish to customise the materials from the original. When the object is instantiated into a scene node, the mesh material properties will be taken by default but may be changed. These properties are actually held at the SubMesh level since a single mesh may have parts with different materials.

As described above, because the mesh may have sections of differing material properties, a mesh is inherently a compound construct, consisting of one or more SubMesh objects. However, it strongly 'owns' its SubMeshes such that they are loaded/unloaded at the same time. This is contrary to the approach taken to hierarchically related scene nodes, where data is loaded/unloaded separately.

3-d Text

3-d text has been created for identifying specific sites, compute elements, etc., in VirtualGrid. Some of the important features are:

- The text is attached to a node, and gets smaller when far away.
- The text is also always facing the camera (like a Billboard).

- The text can be set to a horizontal or vertical position (centre, left, etc...).
- The text can also be translated along each axis, i.e. x, y and z.

Materials

In Ogre, the material defines how an object reflects the light, but not how that reflected light interacts with other objects in the scene. Thus, objects do not become additional lighting sources in the scene as they reflect or emit light. This is often referred to as *global illumination* and is used in raytracing and producing photo realistic images. This form of lighting algorithm is not suitable for real-time 3-d rendering and in Ogre a simplified version of global illumination is used. Ogre supports four different types of colour descriptions for a material: ambient, diffuse, emissive and specular.

3.5 Summary

This chapter outlined a framework for multiscale multimodal grid visualisation. This framework takes advantage of breaking up the visualisation pipeline and distributing the workload to grid resources that are more suited to certain pieces of the pipeline. This means that these resources are utilised to their fullest potential thus making the visualisation more efficient. The rest of the chapter described the development of VirtualGrid, a concrete example of the use of the visualisation framework, and the components of this application.

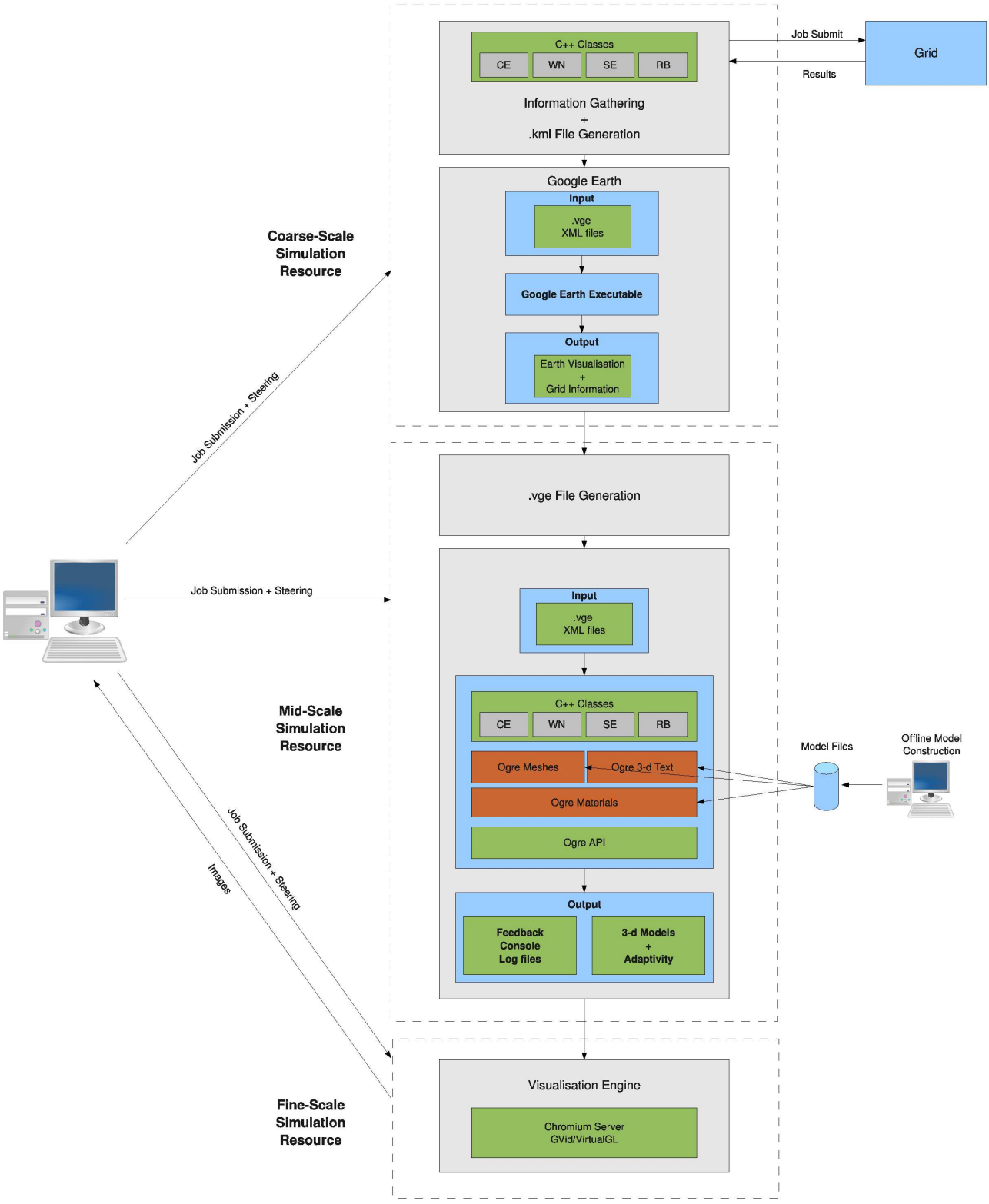


Figure 3.7: VirtualGrid, an example use of the multiscale multimodal visualisation framework.

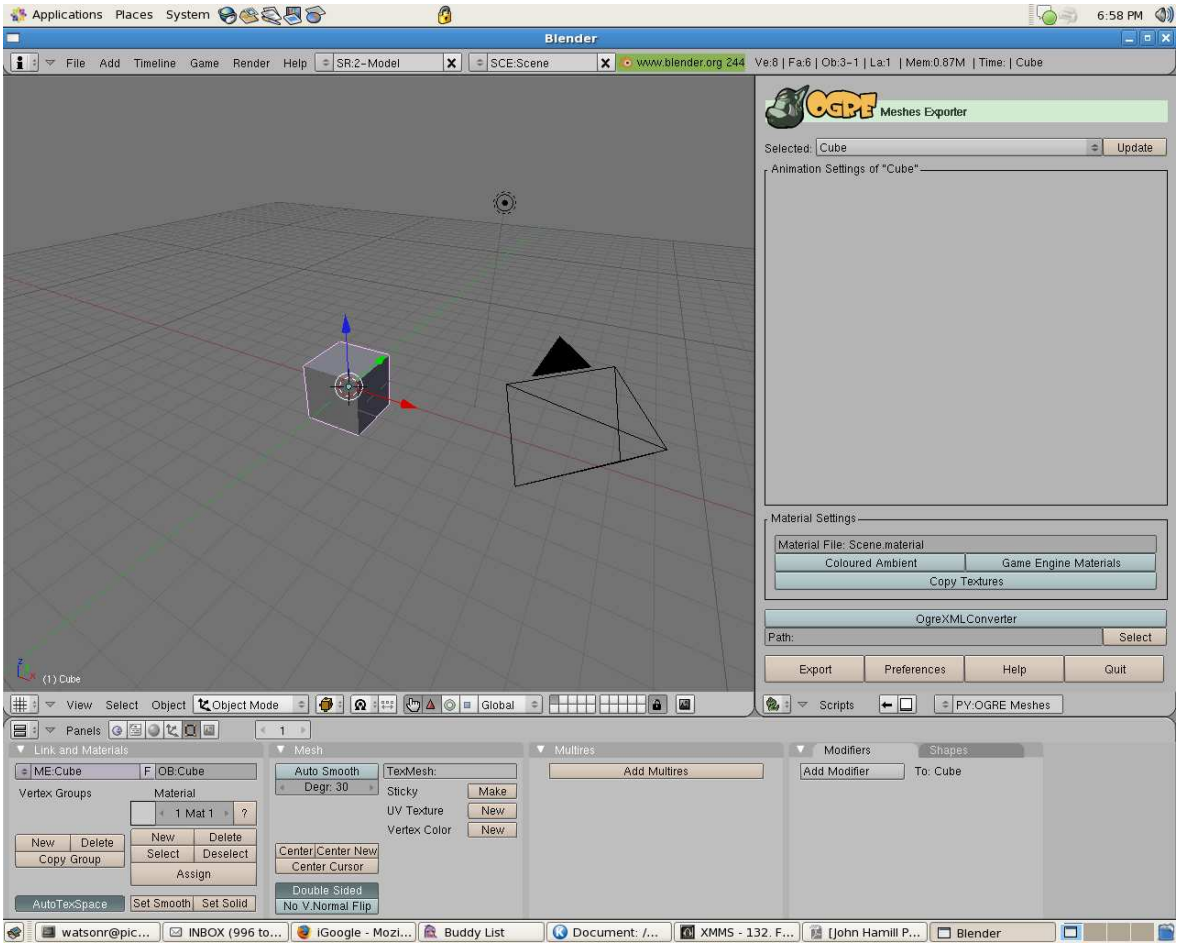


Figure 3.8: Blender modelling package with Ogre mesh exporter. Snapshot by the author.

Chapter 4

Mathematical Model and Benchmarks for Visualisation Framework Characterisation

4.1 Introduction

To enable the characterisation of frameworks for grid-enabled visualisation and computational steering, a mathematical model is required to help identify the complicated relationships between the variables of the architectures and systems used. This chapter will present a discussion of statistical models with their benefits and example applications of their use, along with the decision and reasons why one particular model is chosen. The use of this model will then be to help users make decisions on where their compute and/or visualisation jobs are to be sent on a grid. Users will approach a grid with varying computational loads and model complexities and, after the creation of this mathematical model, be able to use the model to establish which is the best resource for each type of job. Some use cases are examined in Chapter 5.

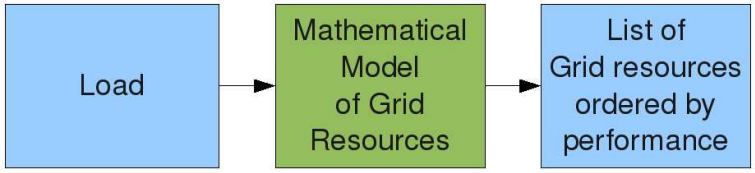


Figure 4.1: Mathematical model producing a list of grid resources most suitable to the user's job type and load.

4.2 Statistical Modelling

Statistical modelling is about finding general laws from observed data and is aimed at learning rules and restrictions based on that set of observed data. In mathematical terms, a statistical model is frequently thought of as a pair (Y,P) where Y is the set of possible observations and P the set of possible probability distributions on Y. It is assumed that there is a distinct element of P which generates the observed data. Statistical inference then enables statements to be made about which element(s) of this set are likely to be the true one.

Statistical methods have two distinct branches: **Descriptive** methods and **Inferential** methods.

- **Descriptive methods:** have three subgroups which are
  - Univariate - the examination of the distribution of cases on only one variable at a time (e.g., college graduation)
  - Bivariate - the examination of two variables simultaneously (e.g., the relation between gender and college graduation)
  - Multivariate - the examination of more than two variables simultaneously (e.g., the relationship between gender, race, and college graduation)

  Descriptive methods are used to examine data that has been collected and summarise it. Descriptive statistics are just descriptive and they do not involve generalising beyond the data at hand. They can be used to obtain the following information from data: mean, median, sum, variance and the standard deviation.

- **Inferential methods:** Inferential statistics are used to make generalisations or inferences about a population based on findings from a sample. With inferential statistics, the aim is to try and reach conclusions that extend beyond the immediate data alone. For instance, they are used to try to infer from a small collection of sample data, what a larger collection might think. Or, are used to make judgements of the probability that an observed difference between groups is a dependable one, or one that might have happened by chance in a study. Thus, inferential statistics are used to make inferences from the data to more general conditions.

To characterise a visualisation framework, a model is needed so that inferences can be made from sample data collected. Two possible inferential methods which could allow for this are Linear Regression and Markov Chains.

4.2.1 Linear Regression Analysis

Linear regression models [155] are extremely powerful, and help to explain very complicated relationships between variables. Generally speaking, the technique is useful, among other applications, in helping explain observations of a dependent variable, usually denoted y, with observed values of one or more independent variables, usually denoted $x_1, x_2, ...$

Regression analysis is most often used for prediction. The goal in regression analysis is to create a mathematical model that can be used to predict the values of a dependent variable based upon the values of an independent variable. In other words, the model is used to predict the value of y when the value of x is known. (The dependent variable is the one to be predicted). Correlation analysis is often used with regression analysis because correlation analysis is used to measure the strength of association between the two variables x and y.

In regression analysis involving one independent variable and one dependent variable the values are frequently plotted in two dimensions as a scatter plot. The scatter plot allows us to visually inspect the data prior to running a regression analysis. Often this step allows us to see if the relationship between the two variables is increasing or decreasing and gives only a rough idea of the relationship.

Regression analysis traces the distribution of a dependent variable Y, as a function of one or more independent variables $(X_1, ..., X_k)$:

$$p(y|x_1, ..., x_k) = f(x_1, ..., x_k)$$

Here, $p(y|x_1, ..., x_k)$ represents the probability of observing the specific value y of the dependent variable, conditional upon a set of specific values $(x_1, ..., x_k)$ of the independent variables; $p(Y|x_1, ..., x_k)$ is the probability distribution of Y for these specific values of X's

Applications of linear regression models

Linear regression is widely used in social sciences, behavioural sciences and in computing science to describe possible relationships between variables. It is one of the most important tools used in these disciplines. Some example applications of the linear regression model are as follows:

- **Finance** - The capital asset pricing model uses linear regression as well as the concept of Beta for analysing and quantifying the systematic risk of an investment. The beta coefficient, in terms of finance and investing, describes how the

expected return of a stock or portfolio is correlated to the return of the financial market as a whole. This comes directly from the Beta coefficient of the linear regression model that relates the return on the investment to the return on all risky assets. Regression may not be the appropriate way to estimate beta in finance given that it is supposed to provide the volatility of an investment relative to the volatility of the market as a whole. This would require that both these variables be treated in the same way when estimating the slope. Whereas regression treats all variability as being in the investment returns variable, i.e. it only considers residuals in the dependent variable.

- **Trend Line** - A trend line represents a trend, the long-term movement in time series data after other components have been accounted for. It tells whether a particular data set (e.g. GDP, oil prices or stock prices) have increased or decreased over the period of time. A trend line could simply be drawn by eye through a set of data points, but more properly their position and slope is calculated using statistical techniques like linear regression. Trend lines typically are straight lines, although some variations use higher degree polynomials depending on the degree of curvature desired in the line. Trend lines are sometimes used in business analytics to show changes in data over time. This has the advantage of being simple. Trend lines are often used to argue that a particular action or event (such as training, or an advertising campaign) caused observed changes at a point in time. This is a simple technique, and does not require a control group, experimental design, or a sophisticated analysis technique. However, it suffers from a lack of scientific validity in cases where other potential changes can affect the data.

4.2.2 Markov Chains

Let $X = \{X_1, X_2, ...\}$ be a random process in the discrete state space ε. It is called a Markov chain [156] if the conditional probabilities between the outcomes at different times satisfy the Markov property, which is explained as follows.

Consider a time t and the event

$$\{X_t, = x_t, X_{t-1} = x_{t-1}, ..., X_1 = x_1\}$$

for some sequence of states $x_t, x_{t-1}, ..., x_t$. This is a record of the entire history of the process up to and including the time t. It is written in reverse-time order because t should be thought as the present time and to express the event starting from the

present and moving back into the past. The conditional probability

$$\mathbb{P}(X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1}, ..., X_1 = x_1)$$

thus represents the probability of an event one step into the future beyond time t, conditioned on the entire past of the process up to t. On the other hand

$$\mathbb{P}(X_{t+1} = x_{t+1} | X_t = x_t)$$

is the conditional probability of the future event given just the present. The Markov property is satisfied when these two conditional probabilities are equal.

Applications of model

- **Queueing theory** - Markov chains can also be used to model various processes in queueing theory and statistics. The concept of entropy [157] derived by Markov modelling of the English language represents an idealised model and such idealised models can capture many of the statistical regularities of systems. Even without describing the full structure of the system perfectly, such signal models can make possible very effective data compression through entropy coding techniques such as arithmetic coding. They also allow effective state estimation and pattern recognition. The world's mobile telephone systems depend on the Viterbi algorithm [158], for error-correction, while hidden Markov models [159] are extensively used in speech recognition and also in bioinformatics, for instance for coding region/gene prediction. Markov chains also play an important role in reinforcement learning.

- **Internet applications** - The PageRank of a web page as used by Google is defined by a Markov chain [160]. It is the probability to be at page i in the stationary distribution on the following Markov chain on all (known) web pages. If N is the number of known web pages, and a page i has k_i links then it has transition probability $\frac{1-q}{k_i} + \frac{q}{N}$ for all pages that are linked to and $\frac{q}{N}$ for all pages that are not linked to. The parameter q is taken to be about 0.15.

  Markov models have also been used to analyse web navigation behaviour of users. A user's web link transition on a particular website can be modeled using first- or second-order Markov models and can be used to make predictions regarding future navigation and to personalise the web page for an individual user.

- **Physics** - Markovian systems appear extensively in physics, particularly statis-

tical mechanics, whenever probabilities are used to represent unknown or unmodelled details of the system, if it can be assumed that the dynamics are time-invariant, and that no relevant history need be considered which is not already included in the state description.

Markov chain methods have also become very important for generating sequences of random numbers to accurately reflect very complicated desired probability distributions, via a process called Markov chain Monte Carlo (MCMC) [161]. In recent years this has revolutionised the practicability of Bayesian inference methods [162], allowing a wide range of posterior distributions to be simulated and their parameters found numerically.

4.3 Selection of the Statistical Model

Having examined the two possible models to help characterise visualisation frameworks, the Linear Regression Model fits best. The reasons for this are:

- The analysis of a linear regression model can be extended to cover situations in which the dependent variable is affected by several controlled variables, and uncontrolled variables. When this occurs a Multiple Linear Regression model [155] can be constructed.
- Grid resources that are available to user are typically 'black box' machines. A 'Black box' is a device or system that is viewed in terms of its input, output and transfer characteristics, without any knowledge of its internal workings. A Multiple Linear Regression model maps very well to this, as it has uncontrolled inputs $x_1, x_2, ..$, outputs y and is modeled by some function f.

Further aspects of linear regression models are described in Appendix E.

Multiple regression

The analysis of the linear regression model can be extended to cover situations where the dependant variable is influenced by more than one other variable. An example of this would be as follows, where there are three variables x_1, x_2, and x_3. A linear regression equation would be of the form

$$y = a_0 + a_1x_1 + a_2x_2 + a_3x_3.$$

Given n sets of measurements, $(y_1, x_{11}, x_{21}, x_{31})$,...,$(y_n, x_{1n}, x_{2n}, x_{3n})$, the least squared estimates of a_0, a_1, a_2 and a_3 can be obtained in a similar way to that described in

Section E.3, and the sum of squared deviations of the observed values of y from the predicted values is given by

$$S = \sum (y_i - a_0 - a_1 x_{1i} - a_2 x_{2i} - a_3 x_{3i})^2.$$

This quantity can be minimised to obtain four simultaneous equations in $\hat{a}_0$, $\hat{a}_1$, $\hat{a}_2$ and $\hat{a}_3$ and these equations, called the normal equations, can be solved to give the least squares estimates of a_0, a_1, a_2 and a_3. Most statistical software available can calculate these values, and that is the case with R [63], the software package the author of this thesis has made use of for the statistical analysis of gathered data.

There are some special cases of multiple regression, for example:

- **Polynomial Regression:** Given that the dependant variable is a polynomial function of a single variable, in cubic regression, the equation is given by

  $$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3.$$

  In this case substitutions can be made as follows, with $x_1 = x$, $x_2 = x^2$ and $x_3 = x^3$, and a_0, a_1, a_2 and a_3 can be solved as before.

- **Mixtures:** Some models involve a combination of multiple and curvilinear regression with an example of this given by

  $$y = a_0 + a_1 x + a_2 x^2 + a_3 z,$$

  $$y = a_0 + a_1 x + a_2 z + a_3 xz.$$

  Again this can be solved using substitution with $x_1 = x$, $x_2 = x^2$ and $x_3 = z$; and in the second case with $x_1 = x$, $x_2 = z$ and $x_3 = xz$.

- **Transformations:** Theoretical considerations may lead to a regression model which depends on a transformation of the controlled variables. A possible regression with variable transformation could be of the form

  $$y = a_0 + a_1 logx + a_2 logz,$$

  where x and z and the variables and the estimates of a_0, a_1 and a_2 can be obtained by setting $x_1 = logx$ and $x_2 = logz$.

4.4 Visualisation Framework Characterisation

In relation to this thesis, it is proposed to construct two types of regression models. For the first type y would denote the completion time of a computation, where $x_1, x_2, ..$ would be the resulting measured variables of benchmarks run on the grid and the input parameters to the computation. For the second type y would denote the frames-per-second (fps) output from a visualisation benchmark running on a grid resource. These models can be considered to represent the quality-of-service (QoS) of the resource.

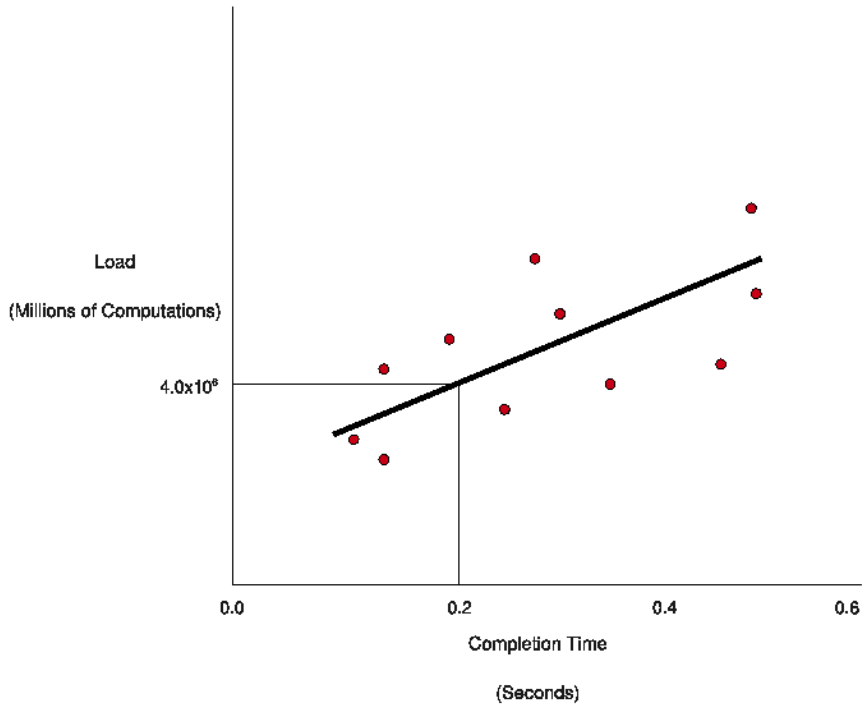


Figure 4.2: Example linear regression graph showing the completion time achieved at increasing numbers of computations.

The visualisation framework proposed in this thesis has several architectural blocks, e.g. coarse-scale, mid-scale and fine-scale resources. Therefore to populate the model, firstly a set of experimental benchmarks need to be performed on the individual architectural blocks with respect to known influential variables such as CPU speed, memory performance, graphical rendering capability and hard disk read and write speeds. These benchmarks need to be run specifically to test one variable at a time so that a detailed profile (i.e. a set of performance estimates) of the architectural blocks is built.

An example linear regression model for a benchmark is shown in Figure 4.2. This linear model would allow for the completion times of a given benchmark to be predicted

within a statistical confidence interval The figure shows the predicted completion time of an application as 0.2 seconds for a load of 4 million computations.

With the profiles established, the model can then be constructed at a later date (by others who do not have access to the raw benchmarking data) for each of the architectural blocks that have been characterised, but this time inferring an overall metric of the block for a specified range of input variables. As an example Figure 4.3 shows a linear regression model of how one architectural block performed in frames per second against an increasing load. We can infer from this graph that subjecting this block to a load of over 4 million polygons would result in a very poor frames per second output.

Figure 4.4 shows two architectural blocks and their frames per second performances and in this case Arch2 performs better than Arch1 with an increasing load. With this linear regression graph it is possible to infer that, if a user requires real-time rendering, 30 fps, on a load of over 2 millions polygons, then Arch2 is the block to use. With Arch2 profiled from the experimental benchmarks the user knows what type of system requirements are needed to obtain real-time rendering at that load.

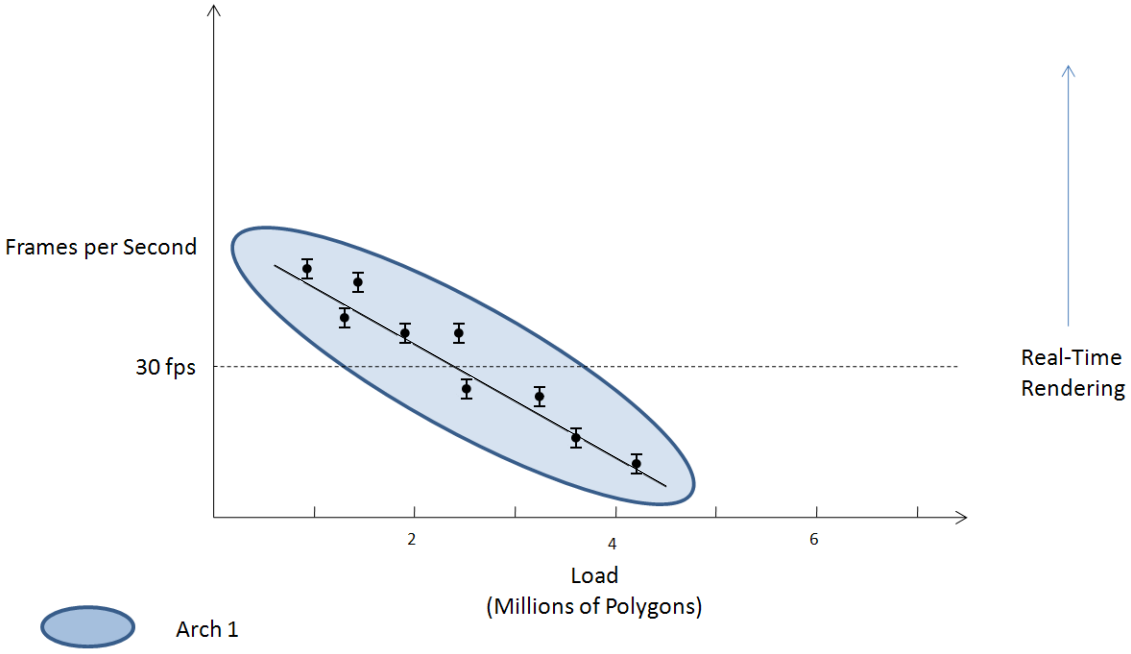


Figure 4.3: Example linear regression graph showing the frames per second achieved at increasing numbers of polygons

Whilst this is a simple concept, the statistical subtleties are less so, the burden of characterisation (although once-off) is high and the conditions for characterisation must be strictly controlled if the model is to remain valid. To the author's knowledge this has not been done, or at least reported in the literature, before. The reason for

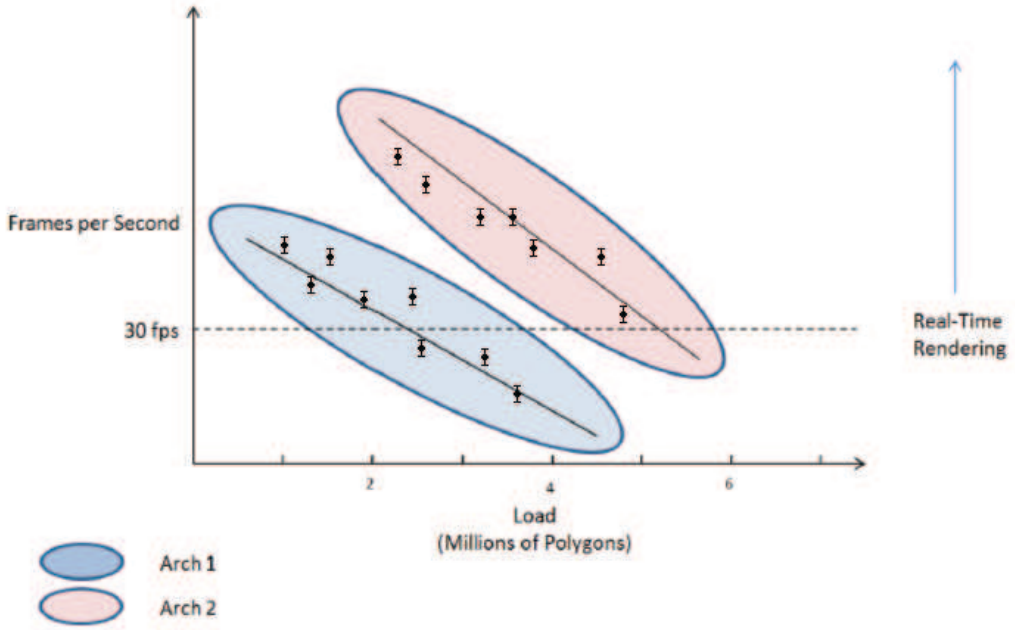


Figure 4.4: Example linear regression graph showing the frames per second achieved at increasing numbers of polygons from 2 different architecture types.

this could be that obtaining a large enough sample size to be statistically significant is difficult.

In Tsouloupas el al. [163] and Georgatos et al. [164] benchmarking and characterisation of grid resources is undertaken by low-level benchmarks. This characterisation, however, only takes the form of observations made of the results, with no mathematical analysis performed.

4.4.1 Resource Benchmarking

In the currently evolving European context there are more than 30 national grid providers (National Grid Initiatives, or NGIs). For a grid provider to be able to publish information about their grid in a way that can be used in the construction of a performance model, it is necessary to run a set of specifically selected benchmarks on the resources of their grid, which will gather detailed performance information.

Figure 4.5 shows the generalised description of the resource benchmarking of such grids. For a $grid[N]$, with B benchmarks, where $grid[N]$ has j resource providers, $j \times B$ jobs are run on $grid[N]$. This yields performance estimates $E[N,1], ..., E[N,j]$ for each benchmark on every resource on that grid.

Alternatively one could take a collective view. For the past ten years a series of EU projects (EDG, Crossgrid, Int.EU.Grid, EGEE, EGEE-II, EGEE-III) have aggregated

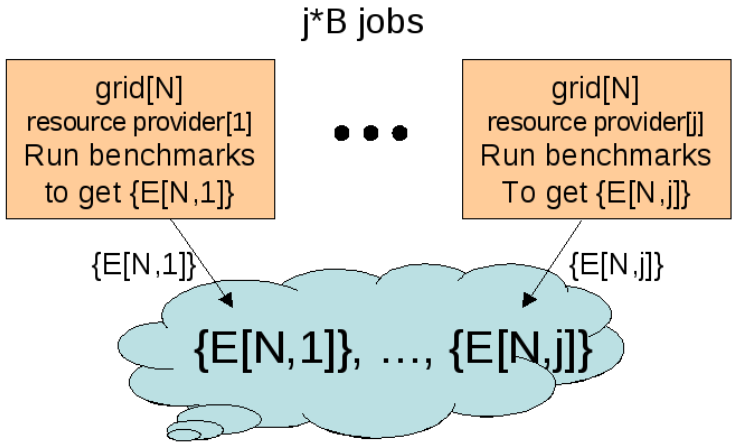


Figure 4.5: Resource benchmarking: For the grid[N], with B benchmarks, where grid[N] has j resource providers. This yields performance estimates E for each benchmark on every resource on that grid.

grid site resources into a single large grid (EGEE-III, which ends in April 2010, consists of 270 sites in over 50 countries). From May 2010, the national grid initiatives (NGIs) are establishing a permanent federated European Grid Infrastructure (EGI), which also could be considered as a single large grid. In each case the aggregated site resources could be benchmarked as one Grid.

4.4.2 Model Construction

Figure 4.6 describes the method for an application developer to construct a performance model for their application. Assuming the set of resources that might best run that application have been benchmarked, then the application model can be constructed by testing the application on a smaller subset of the resources. In fact, this is its value - one does not have to test the application on *all* the resources in order to infer its properties within a certain degree of confidence.

For an application k on $grid[N]$, where $grid[N]$ has j resource providers, h jobs need to be run on $grid[N]$, where $h < j$. This yields a performance model $M[N, k]$, using linear regression and correlation analysis, for $application[k]$ on $grid[N]$.

4.5 Benchmarks for Framework Characterisation

In today's modern world of computing there are many different types of benchmarks that are used to present information about a system's performance to current and potential users. Due to the large scope of benchmarks this chapter will firstly describe

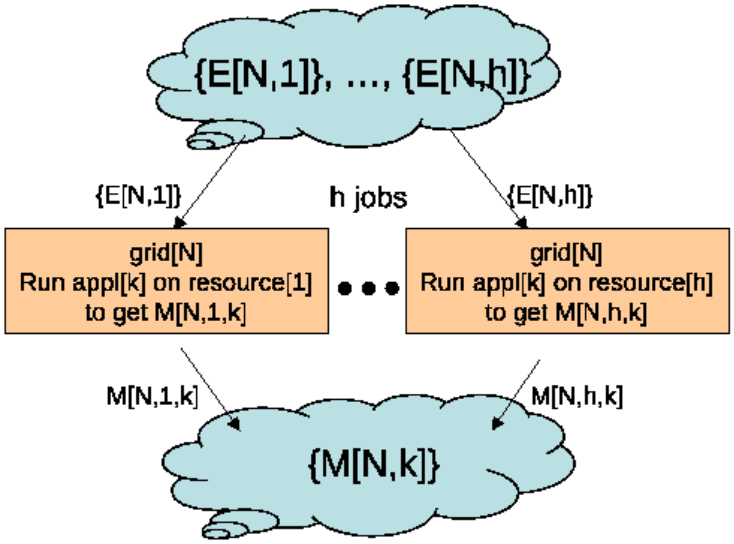


Figure 4.6: Model construction: For the application k on grid[N], where grid[N] has j resource providers, only h of which are tested. This yields a performance model for application[k] on grid[N].

an ontology for benchmarks which will help define the relationships that exist between entities of a grid. It will then introduce and explain the different types of benchmarks that will be used to characterise a framework for compute and visualisation grid jobs. Thirdly, the results from running these benchmarks are presented and analysed with respect to the mathematical models discussed in Chapter 4. Lastly a summary of the chapter is made. Table 4.1 outlines the sections that discuss resource benchmarking and model construction in Grid-Ireland and EGEE.

4.6 Benchmarking of Heterogeneous Resources

In *Grid benchmarking: vision, challenges and current status*[165] a simple Grid performance ontology is suggested. It describes the metrics used to evaluate the performance of compute/storage nodes, grid sites and whole grids themselves. As this thesis intends to characterise a framework for compute and visualisation jobs the author of this thesis has extended this ontology to include visualisation nodes and their metrics for performance evaluation, see Figure 4.7. This new ontology considers visualisation nodes as part of the Grid and the metrics that are associated with benchmarking such nodes. The Grid Infrastructure is at the highest level with Site Performance being at the next level. The sites are broken down into Computing, Storage and Visualisation nodes and CPU, Memory, I/O and Graphics performance metrics.

Methodology	Coarse and Mid-scale (simulation)	Section	Fine-scale (rendering)	Section
Resource Benchmarking	Grid-Ireland + EGEE	5.2 5.2	Grid-Ireland	5.4
Model Construction	Grid-Ireland + EGEE	5.3.2 5.3.3	Grid-Ireland	5.5.1

Table 4.1: Sections that discuss resource benchmarking and model construction in Grid-Ireland and EGEE

4.6.1 Benchmarks

As the Grid is composed of many heterogeneous complicated systems working together, the benchmarks are not trivial. They need to be separated into different types. There are 3 different types that this thesis will examine and apply in order to obtain performance measurements. The 3 types are defined by Tsouloupas and Dikaiakos as follows [166]:

- **Micro-benchmarks:** are small benchmarks, each of which measures a particular aspect of the system under controlled conditions. These benchmarks can estimate the performance of disk access speeds, or cache read/write times. By collecting the results of micro-benchmarks a clear picture of a system's overall performance can be gained.

- **Micro-kernels:** are benchmarks that test the performance of multiple aspects of a system while under a realistic synthetic workload. This workload tries to simulate working conditions of that entity while working in a grid. These benchmarks can obtain performance measurements of a grid site or just a visualisation node within a site.

- **Application kernels and Grid applications:** are used to investigate the performance of some Grid entity (node, site, infrastructure) under realistic workload conditions.

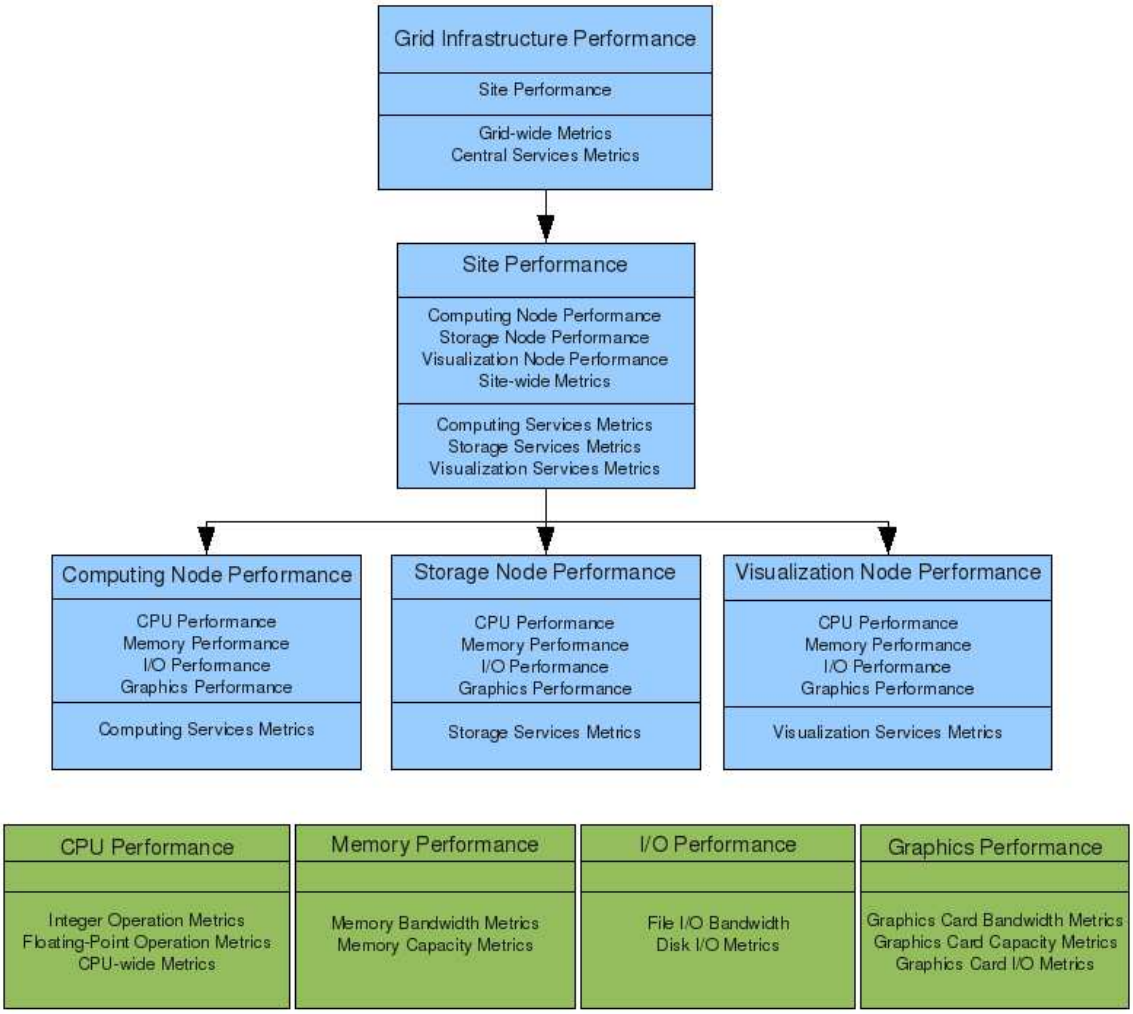


Figure 4.7: Performance Ontology

These benchmarks are designed to investigate the performance of Grid entities belonging to different layers of the Grid architecture as seen in Figure 4.7. These entities range from complete Grids, to Grid Sites, to single CPUs available on the Grid.

4.6.2 Micro-benchmarks

In [163] a description of low-level benchmarks are provided and these correspond to the concept of micro-benchmarks. The specific system attributes that are considered for benchmarking in this thesis are based on CPU, Memory, Interconnect and Graphics cards of the heterogeneous systems.

Metrics associated with CPU benchmarking are measuring Floating Point and Integer operations per second while Memory performance metrics include measuring the

memory bandwidth in MBs (copy, add, multiply) and Physical Memory. Interconnect metrics measure the Latency and Bandwidth, and a Graphics Card's performance is based on Bandwidth and frame per second (fps) measurements on selected applications.

The following subsections describe the micro-benchmarks used. The benchmarks are part of a software package called GridBench. GridBench [166] is a tool for evaluating the performance of grids and grid resources through benchmarking.

EPWhetstone

EPWhetstone is a C program which is an altered version of the original Whetstone program [167]. The original Whetstone program is designed to be run as single CPU benchmark whereas EPWhetstone has been modified so that it can run in parallel across multiple CPUs using MPI. It measures operations per second of a mixture of floating point and integer calculations taking an everage rate at which these operations were performed, but not taking into account the communication time across CPUs.

EPDhrystone

EPDhrystone is another adaptation of a program called Dhrystone [168] which is a benchmark for CPUs with integer operations per second as it's metric. It has also written in C and has been modified to run across multiple CPUs using MPI.

EPFlops

EPFlops is a benchmark for investigating the CPU's power for floating-point calculations, adapted from the flops benchmark [169] and written in C. Again it is modified so that it runs across multiple CPUs using MPI and attempts to estimate a system's floating-point 'MFLOPS' rating for the FADD, FSUB, FMUL, and FDIV operations based on certain 'instruction mixes'. The program provides an estimate of peak MFLOPS performance by making maximal use of register variables with minimal interaction with main memory. The execution loops are all small so that they will fit in any cache.

glxspheres

glxspheres is a C program that produces a number of continuously translating graphical objects (mostly spheres) on screen using OpenGL. It can be used as a benchmarking tool as it is possible to vary the number of polygons that are drawn, and the author of this thesis has altered the program so that it can also be possible to change the reso-

lution of the output e.g 320x240, 640x480,800x600 and 1024x768. The FPS achieved by the program while it is running is echoed to 'standard out'.

4.7 Summary

This chapter presented a discussion of statistical models with their benefits and example applications of their use, along with the decision and reasons why linear regression is most suited to model the performance predictions of grid resources. In the context of the visualisation framework, the modelling requires two steps, firstly resource benchmarking, and secondly model construction. Each of these has been described.

A modified performance ontology was then described that included visualisation nodes and their associated metrics. This ontology helped classify grid resources so that they could be benchmarked according to appropriate benchmarks. A set of micro-benchmarks was then presented along with the details of their specific area of examination.

Chapter 5

Grid Framework Characterisation

5.1 Introduction

The micro-benchmarks presented in Section 4.6.2 when included in grid jobs and run a statistically significant amount of times, produce estimates of the performance of a resource in relation to that micro-benchmark. The main purpose of these estimates is to build a profile of a grid resource and have a basic way of describing the performance of a resource with respect to its computational power and/or rendering capabilities. These estimates can then be used to help explain the performance variance of an application across other grid resources.

The following sections describe the coarse-scale/mid-scale/fine-scale benchmarking and model construction of the Grid-Ireland and EGEE Grid resources.

5.2 Example Coarse-scale/Mid-scale Resource Benchmarking

With benchmarks selected to help define grid resources, two examples of benchmarking the coarse-scale/mid-scale resources of a grid are given in the following sections. These grids are Grid-Ireland and the EGEE grid.

Grid-Ireland

Figure 5.1, which is derived from Figure 4.5, describes the method with which to benchmark Grid-Ireland's resources and obtain performance estimates. Three micro-benchmarks, EPDhrystone, EPWhetstone and EPFlops, are sent, as non-MPI (single core) grid jobs, to each Grid-Ireland resource. They are run on each resource 60 times to get an estimate and the results are sent back as a job output file.

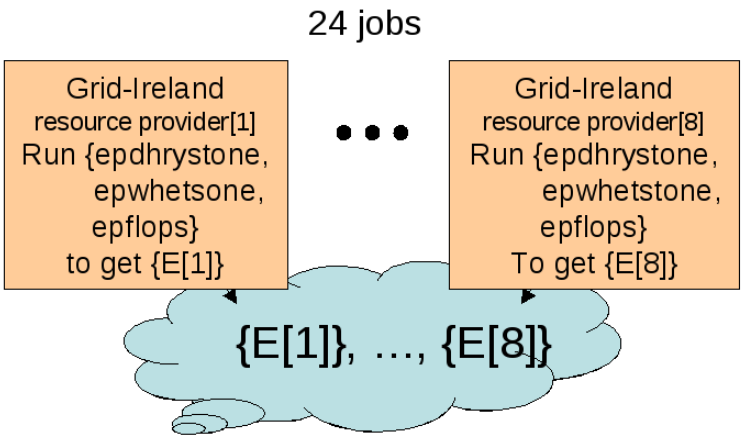


Figure 5.1: Resource benchmarking: For Grid-Ireland, with three (B=3) micro-benchmarks epdhrystone, epwhetstone, epflops, where Grid-Ireland has 8 resource providers. This yields performance estimates E for each micro-benchmark on every resource on Grid-Ireland.

Figures F.1, F.2 and F.3 show the results of the jobs run on Grid-Ireland. *ps001.grid.cs.tcd.ie* is a PlayStation3 machine which performs poorly across all micro-benchmarks. This is due to the fact that the micro-benchmarks have not been ported, so far, to make use of the six Synergistic Processing Elements (SPE) that are available while running under a Linux operating system. Later in this chapter it will be shown that running a program compiled to execute on a PS3, which takes advantage of all six SPEs, performs better than other machines in Grid-Ireland.

EGEE Grid

Figure 5.2, which is derived from Figure 4.6, describes the method with which to benchmark EGEE's resources and obtain performance estimates. The micro-benchmarks, EPDhrystone, EPWhetstone and EPFlops, are sent, as non-MPI (single core) grid jobs, to each EGEE resource. They are run on each resource 60 times to get an estimate and the results are sent back as a job output file. Figures F.4, F.5 and F.6 show the results of the jobs run on EGEE.

5.3 Example Coarse-scale/Mid-scale Model Construction

With performance estimates obtained an example application is selected to test the linear regression model construction. The FFTW [170][171] application, described in more detail in Section 5.3.1, is a CPU intensive application which computes the discrete

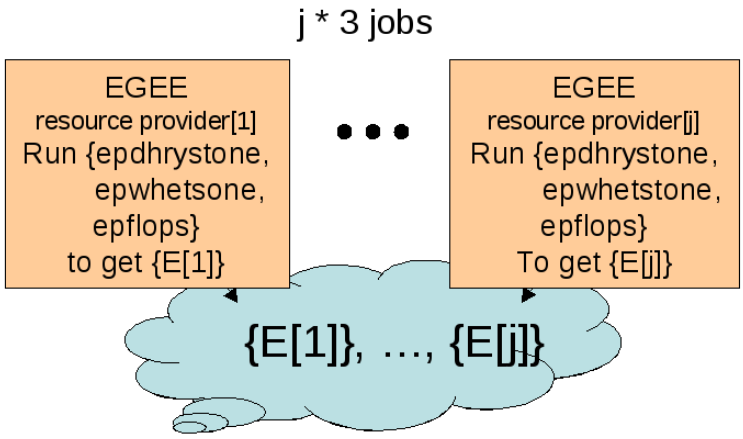


Figure 5.2: Resource benchmarking: For the EGEE Grid, with three benchmarks epdhrystone, epwhetstone, epflops, where EGEE has 270 resource providers. This yields performance estimates E for each benchmark on every resource on EGEE.

Fourier transform in one or more dimensions. The completion time of this application is what is of interest here, and with the construction of a linear regression model, the author hopes to show that it is possible to predict the completion time of a FFTW job submission. Figure 5.3 shows the grid jobs that need to be run in order to obtain a performance model for the FFTW application. These are run as non-MPI (single core) jobs.

5.3.1 Fastest Fourier Transform in the West (FFTW)

FFTW [170][171] is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST).

Data Formats

- **Complex Transforms:** Most FFT routines in the benchmark accept complex data in interleaved format, i.e., as arrays of complex numbers. Some routines accept complex data in split format, i.e., as two separate arrays containing the real and imaginary part of the data. Some other routines accept either format.

  Routines that accept a single format were benchmarked with that format. Routines that accept either format were benchmarked with interleaved data, which seems to be the most commonly used.

  In interpreting the benchmark results, please notice that different formats are

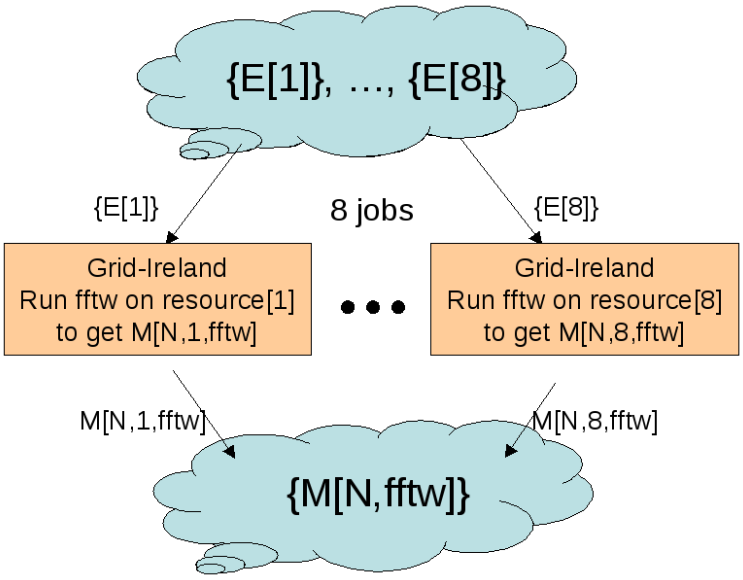


Figure 5.3: Model Construction: For the FFTW application on Grid-Ireland, where Grid Ireland has 8 resource providers. Normally only a subset need testing, but with so few providers, all need testing. This yields a performance model for FFTW on every resource on Grid-Ireland.

not strictly comparable.

- **Real Transforms:** Each FFT routine seems to have its own way of storing the conjugate-symmetric output of real transforms, especially for multidimensional transforms. Each routine was benchmarked using whatever format the routine chose to implement.

  Again, routines that use different formats are not strictly comparable. Consequently, the plots can only be interpreted as a rough description of the relative merits of different codes. A faster code is of no use if it produces the output in a format that is not wanted.

Timing

The FFT timing measurement is intended to reflect the common case where many FFTs of the same size, indeed of the same array, are required. Thus, the measurement is broken into two parts:

- **Initialisation:**

  First, any separate initialisation/setup routines provided by the code are called. This step may include calling the code's FFT once, if it performs initialisation on the first call. This setup time is measured separately from the FFT performance

below, but only as a rough indicator; no attempt is made to perform repeated measurements or to make the initialisation preparations as efficient as possible.

- **Performance:**

  Second, the FFT performance is measured by performing repeated FFTs of the same zero-initialised array.

  The input array is initialised to zero to prevent divergences from repeated FFTs of the same array. No FFT code or any floating-point hardware in the benchmark has a speed that depends on the input data (except for floating-point exceptions).

  The timing procedure consists of two loops. Firstly enough repeated FFTs are computed so that the total time is sufficient for accurate timing, and divided by the number of iterations to obtain the average time. Secondly, this averaging process is repeated eight times, and the minimum average time (to avoid fluctuations due to system interrupts, cache priming) is reported.

Reporting

To report FFT performance, the "mflops" of each FFT is plotted, which is a scaled version of the speed, defined by:

$$mflops = \frac{5Nlog2(N)}{(timeforoneFFTinmicroseconds)}$$

for complex transforms, and

$$mflops = \frac{2.5Nlog2(N)}{(timeforoneFFTinmicroseconds)}$$

for real transforms, where N is number of data points (the product of the FFT dimensions). This is not an actual flop count; it is simply a convenient scaling, based on the fact that the radix-2 Cooley-Tukey algorithm asymptotically requires five $Nlog2(N)$ floating-point operations. It allows the performance for many different sizes to be compared on the same graph, to get a sense of the cache effects, and provide a rough measure of "efficiency" relative to the clock speed.

In Figure 5.4 the completion time for FFTW complex double precision 1D transforms is plotted against the exponential input of 2^x where $x = 1$ to 21. Regression lines are fitted to the data using the *method of least squares* described in Section E.3.

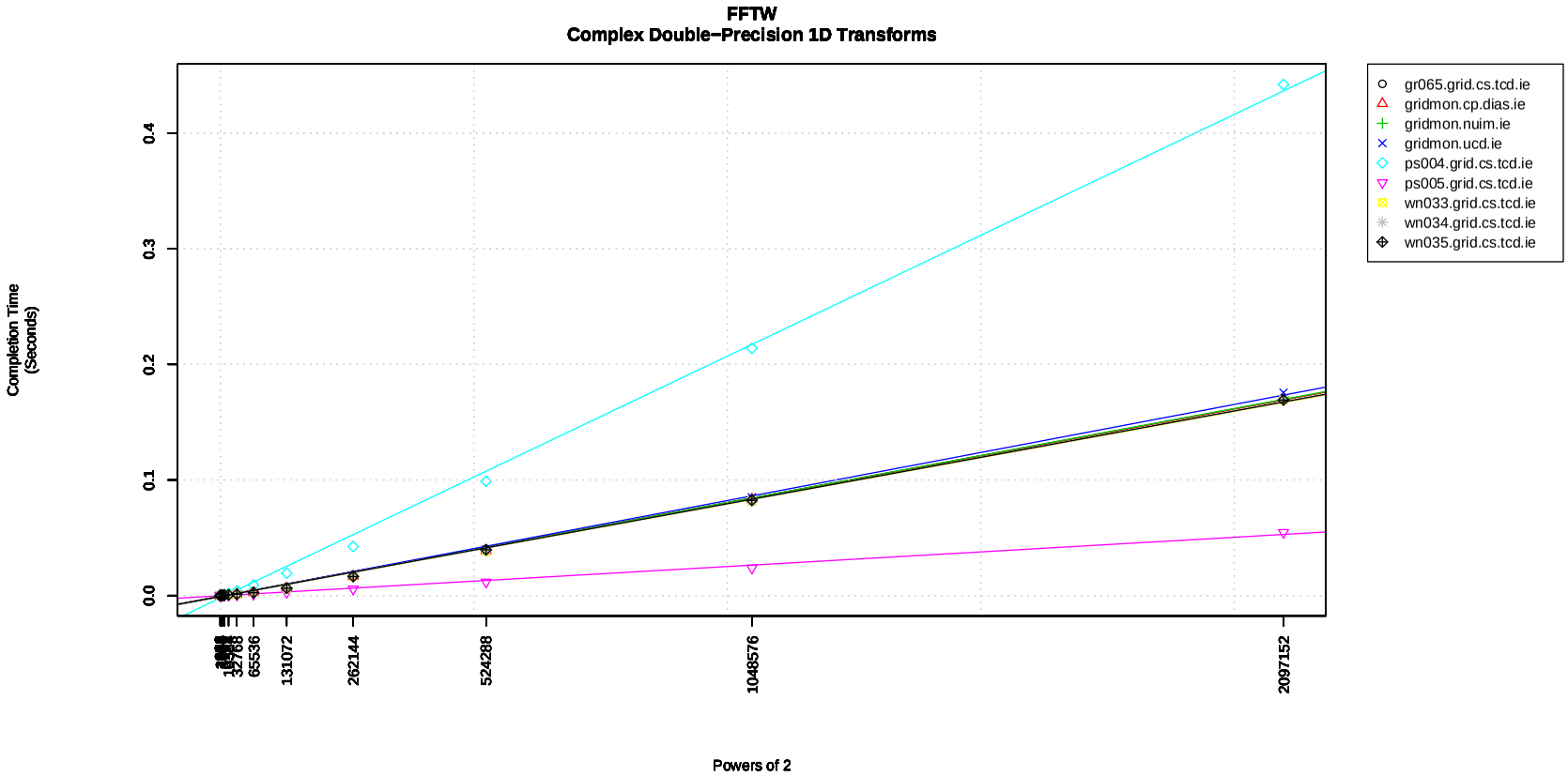


Figure 5.4: Completion Time for FFTW Complex Double Precision 1D Transforms with Powers of 2 Inputs for resources on Grid-Ireland. *ps004.grid.cs.tcd.ie* runs a FFTW program which is specifically compiled for the CellBE architecture.

5.3.2 FFTW Model Construction (for/on) the Grid-Ireland Coarse-scale/Mid-scale Resources

Let us examine if there is a relationship between the results for the FFTW program (particularly the measured completion time of the FFTW program) and the results for the benchmarks. From the examination of linear regression analysis in the previous chapter the first step in finding out if the variables are interdependant is to make scatter plots of the data gathered. Figure 5.5 shows the estimates obtained from running 60 executions of EPFlops, plotted against the completion time of FFTW for the largest input size parameter of 2097152, on each available resource in Grid-Ireland. A significant number of available resources on Grid-Ireland are of the same architecture which means that a lot of the obtained results are very similar. This can be observed in the bottom left part of Figure 5.5. While these results show that the estimates from running the EPFlops benchmark are very stable, the effect on the construction of the FFTW model, if these results are used, is to skew the model to how the FFTW application would run on Grid-Ireland alone. While this is another possible use of the model, where grids can be compared to each other, this section will only deal with constructing a linear model to describe the FFTW application performance on a generalised architecture. Therefore, all but one of the estimates obtained from the same architectures are considered resulting in five estimates of EPDhrystone, EPWhetstone, EPFlops and FFTW that are used.

The resulting Figure 5.6 shows the EPFlops results plotted against the FFTW completion time without the skew, and one can note that the slope of the linear relationship has altered. Figures 5.7 and 5.8 also show the estimates of 60 runs of EPDhrystone and EPWhetstone on each of the five machines.

From examining these figures, it would appear that there is a relationship between the completion time of the FFTW results and the EPDhrystone and EPWhetstone benchmarks, namely that the shorter the completion time the higher results gained in the benchmarks. The scatterplot of EPFlops, Figure 5.6, shows that the EPFlops would appear to have a very low correlation coefficient for the sample size of 5. Of course, given a larger sample size this correlation might become more significant, but in this instance the EPFlops estimate will be discarded.

For a multiple linear model of the form $y = a_0 + a_1x_1 + a_2x_2$, the variables x_1 and x_2 must be independent of each other. A scatterplot of x_1, the EPDhrystone estimates, and x_2, the EPWhetstone estimates is drawn and can be seen in Figure 5.9. This figure shows that there appears to be a strong linear relationship between these two estimates, making them not independent of each other, and therefore only

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	5.424×10^{-01}	$7.942\times10^{-}02$	6.829	0.00642
a_1	-3.509×10^{-08}	1.043×10^{-08}	-3.366	0.04355

Table 5.1: Linear regression summary of complex data for 1-d powers of 2 FFTW application executions, with an input size parameter of 2097152 and the EPDhrystone estimates as the independent variable.

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	5.486×10^{-01}	5.709×10^{-02}	9.609	0.00239
a_1	-2.444×10^{-04}	5.043×10^{-05}	-4.845	0.01678

Table 5.2: Linear regression summary of complex data for 1-d powers of 2 FFTW application executions, with an input size parameter of 2097152 and the EPWhetstone estimates as the independent variable.

a linear regression model can be constructed. To decide on which linear regression model, either $y = a_0 + a_1x_1$ or $y = a_0 + a_2x_2$, the P values for each model must be calculated.

Solving for a_0 and a_1 in both equations the resulting coefficients, standard errors, t values and corresponding (two-tailed) p-values are shown in Tables 5.1 and 5.2 for complex data of 1-d FFTW application executions with an input size parameter of 2097152. x_1 and x_2 are both deemed statistically significant to the 5 percent level, but the regression model using the EPWhetstone estimates is slightly more significant and would therefore model the FFTW completion time more accurately.

In regression, the r-squared coefficient of determination is a statistical measure of how well the regression line approximates the real data points. An r-squared value of 1.0 indicates that the regression line perfectly fits the data. An adjusted r-squared value is a modification of r-squared that adjusts for the number of explanatory terms in a model. Unlike r-squared, the adjusted r-squared value increases only if the new term improves the model more than would be expected by chance. The adjusted r-squared value for this model is 0.8489 which means that the regression line is not a perfect fit to the real data points.

With the values for a_0 and a_1 calculated, a prediction of FFTW completion time for 1-d complex data with an input size parameter of 2097152, can be made given a machine's EPWhetstone(M_w) benchmark estimates. The full equation is as follows:

$$\begin{aligned} y &= 5.486 \times 10^{-01} \\ &\quad +(-2.444 \times 10^{-04}) \times (M_w) \end{aligned}$$

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	$-1.827\times10^{+01}$	1.377×10^{-01}	-132.673	2.0×10^{-16}
a_1	-3.898×10^{-04}	8.952×10^{-05}	-4.355	3.18×10^{-05}
a_2	$1.200\times10^{+00}$	1.223×10^{-02}	98.128	2.0×10^{-16}

Table 5.3: Multiple linear regression summary of complex data for 1-d powers of 2 FFTW application executions using the EPWhetstone estimates and the input size parameter as the independent variables.

Figure F.7 shows the completion times of the complex data 1-d FFTW runs on the available Grid-Ireland machines, and it shows that the result is exponential. The results need to be converted using logarithms by taking the natural logarithm of the completion time and the input size parameter the resulting plot is shown in Figure F.8. This shows clearly that there is a linear relationship between the input size parameter and the completion time, which means a regression model can be constructed. The variations in completion time that are seen at each input size parameter measurement need to be examined statistically to see if they can be explained by the EPWhetstone estimates.

As the input size parameters and the EPWhetstone estimates are independent variables a multiple linear regression equation can be constructed of the form

$$log(y) = a_0 + a_1x_1 + a_2logx_2,$$

where x_1 is the EPWhetstone estimates and x_2 is the input size parameter.

Solving for a_0, a_1 and a_2 the resulting coefficients, standard error, t value and corresponding (two-tailed) p-value are shown in Table 5.3 for complex data for 1-d FFTW application executions.

With the values for a_0, a_1 and a_2 calculated and determined to be significant by the t tests and the adjusted r-squared value, a prediction of FFTW completion time for 1-d complex data can be made given a machine's EPWhetstone(M_w) benchmark estimates and input size parameters.

$$\begin{aligned} logy &= -1.827 \times 10^{+01} \\ &\quad +(-3.898 \times 10^{-04}) \times (M_w) \\ &\quad +(1.2) \times log(InputSize) \end{aligned}$$

As an example of how this equation could be used to predict the performance of other coarse-scale resources that are not on Grid-Ireland, access was granted to

resources on the UK NGS, thanks to Dr. Andy Richards, Director of UK NGS, to allow benchmarking of their machines. Grid jobs which contained the micro-benchmarks, EPDhrystone, EPWhetstone and EPFlops, were sent to an available node in NGS called *node034.ngs.oerc.ox.ac.uk*. Before the FFTW application was sent, a prediction of the completion time for 1-d complex data at an input size of 2097152 and with an obtained EPWhetstone estimate of 2076, was made using the equation above, and evaluated as follows:

$$\begin{aligned}
logy &= -1.827 \times 10^{+01} \\
&\quad +(-3.898 \times 10^{-04}) \times (2076) \\
&\quad +(1.2) \times log(2097152) \\
logy &= -1.827 \times 10^{+01} \\
&\quad -0.8092 \\
&\quad +17.4673 \\
logy &= -1.611 \\
e^{logy} &= e^{-1.611} \\
y &= 0.1996
\end{aligned}$$

The FFTW program was then run on *node034.ngs.oerc.ox.ac.uk* and the results were collected. The measured completion time for the FFTW program with a 2097152 input parameter size was 0.202827 seconds, which shows that the multiple linear regression equation is a good predictor for coarse-scale and mid-scale grid resources of other grid sites.

Regression models have been constructed for the other eleven FFTW benchmarks, i.e. for 1-d, 2-d and 3-d real data, for non-powers of 2 of the input size, and combinations thereof.

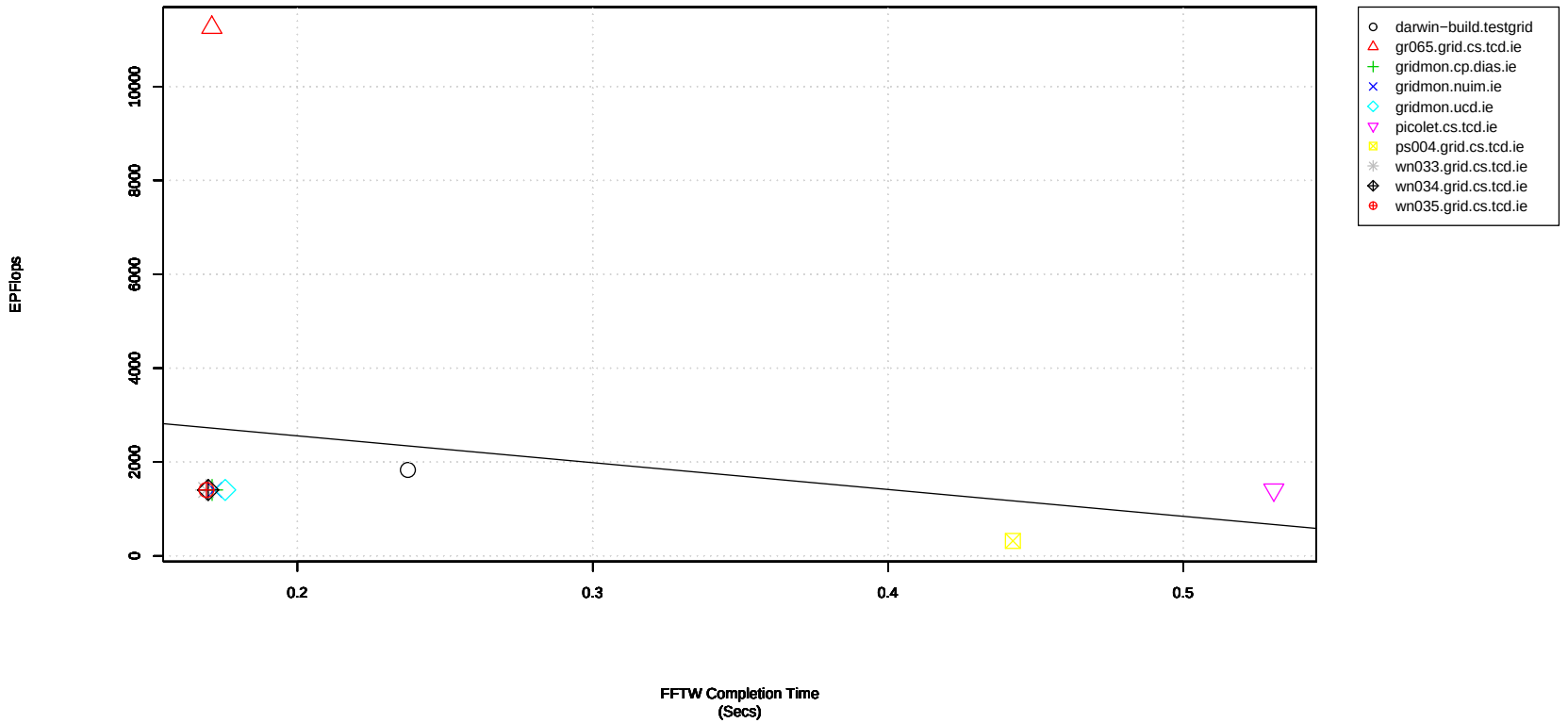


Figure 5.5: Estimates of EPFlops for each machine with architecture skew.

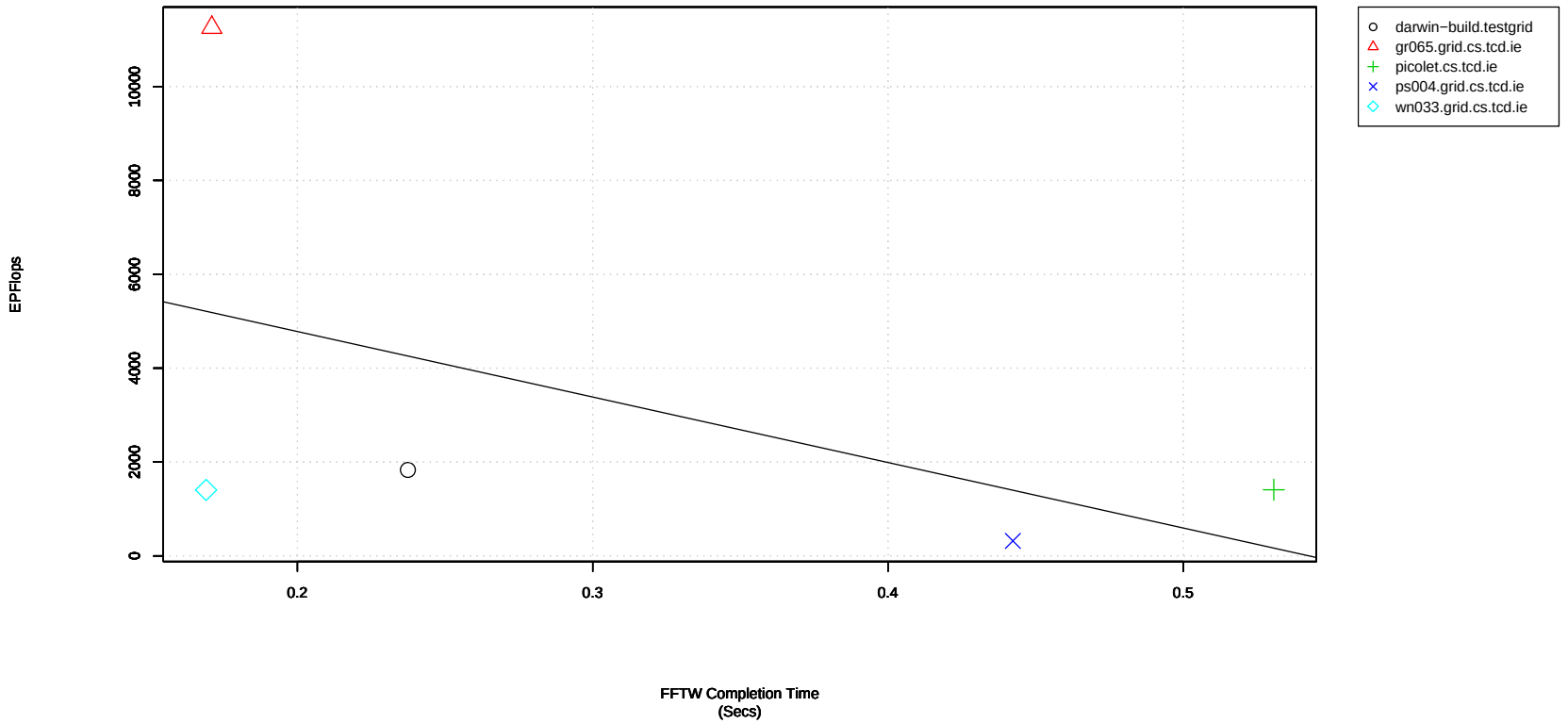


Figure 5.6: Estimates of EPFlops for each machine without architecture skew.

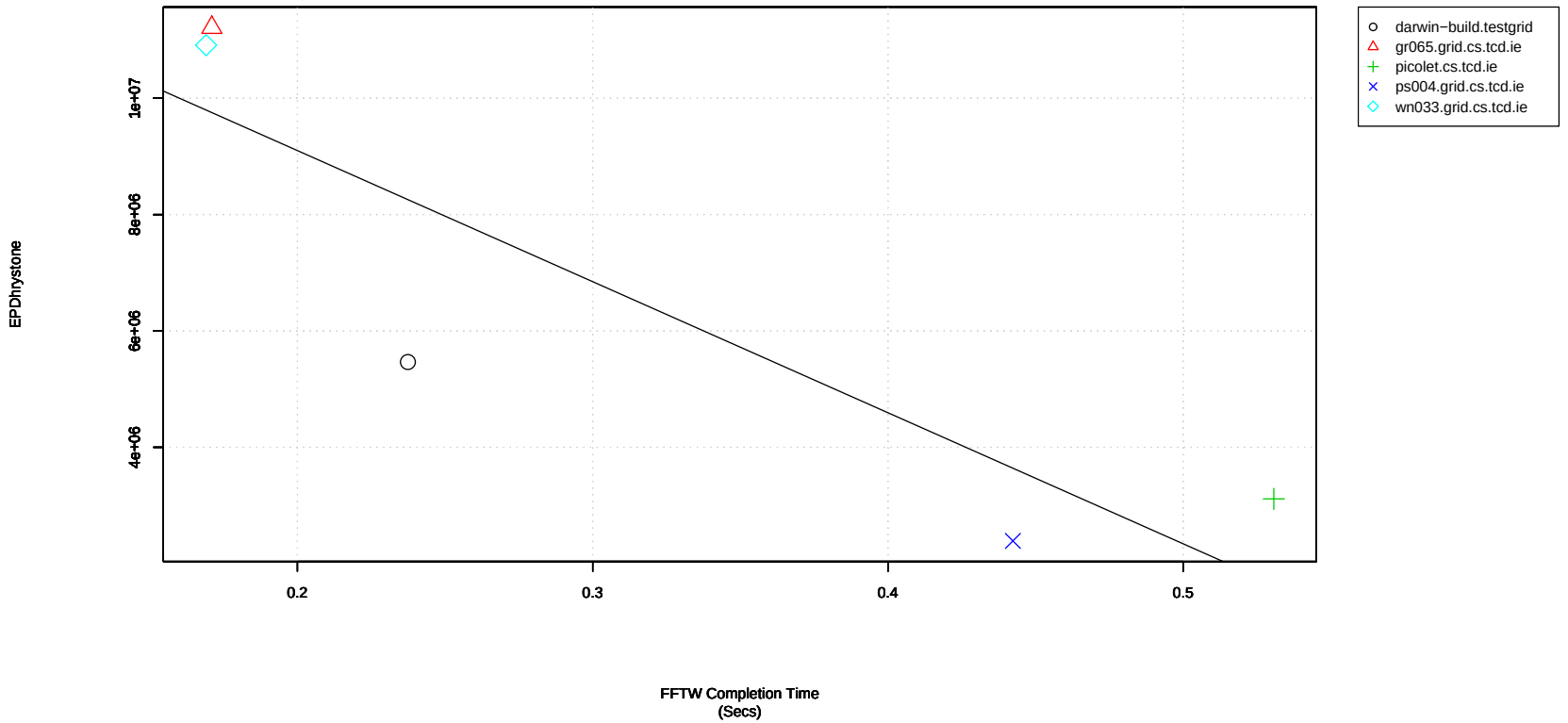


Figure 5.7: Estimates of Dhrystones from each machine

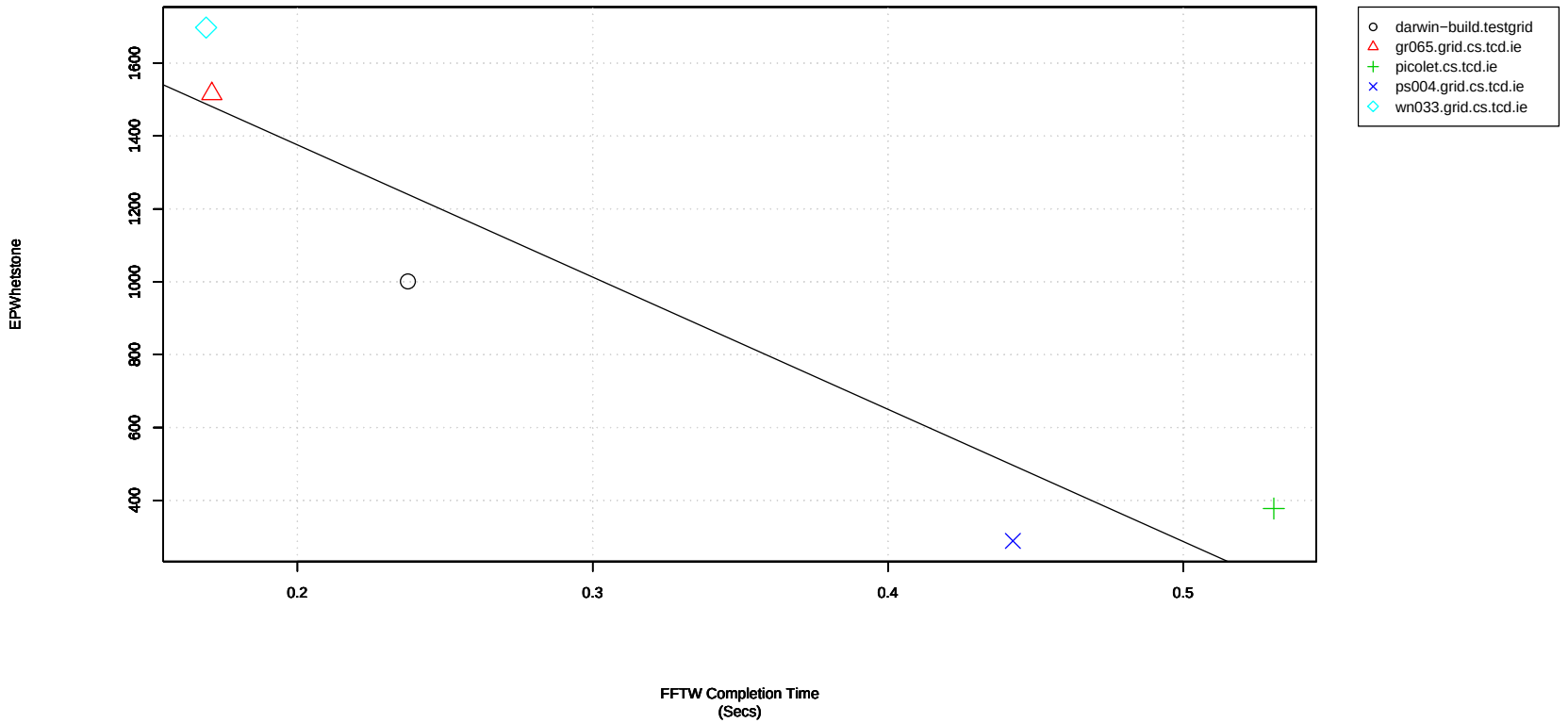


Figure 5.8: Estimates of EPWhetstone from each machine

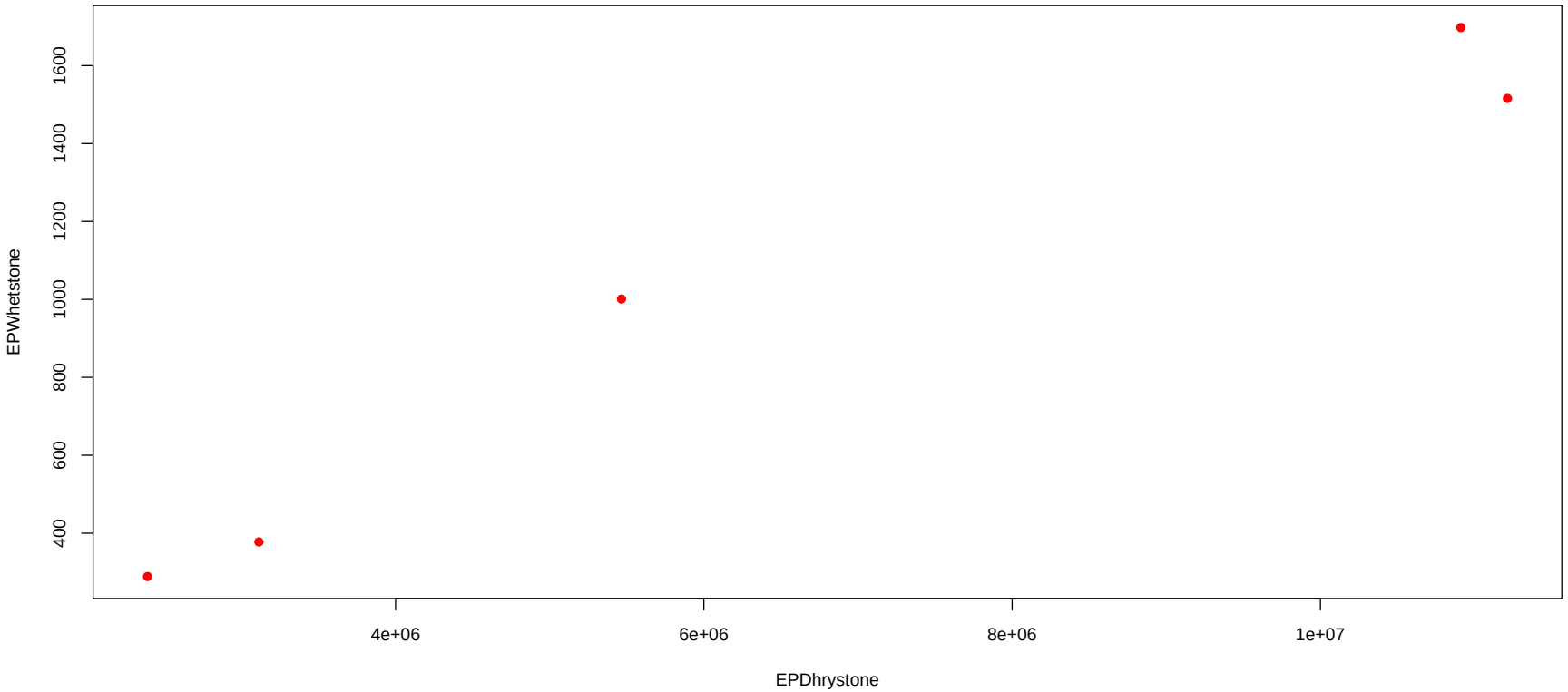


Figure 5.9: Scatterplot of the EPDhrystone estimates against the EPWhetstone estimates, showing a linear relationship.

5.3.3 FFTW Model Construction for the EGEE Grid Coarse-scale/Mid-scale Resources

The author was granted a request for permission to benchmark all resources on the EGEE Grid. This allows the construction of a model for FFTW using a larger sample set, based on a very large set of resource benchmarks. It also allows examination of the influence of sample set size. Moreover, by testing FFTW on *all* EGEE resources it allows validation of the model, i.e. comparison of predictions versus reality.

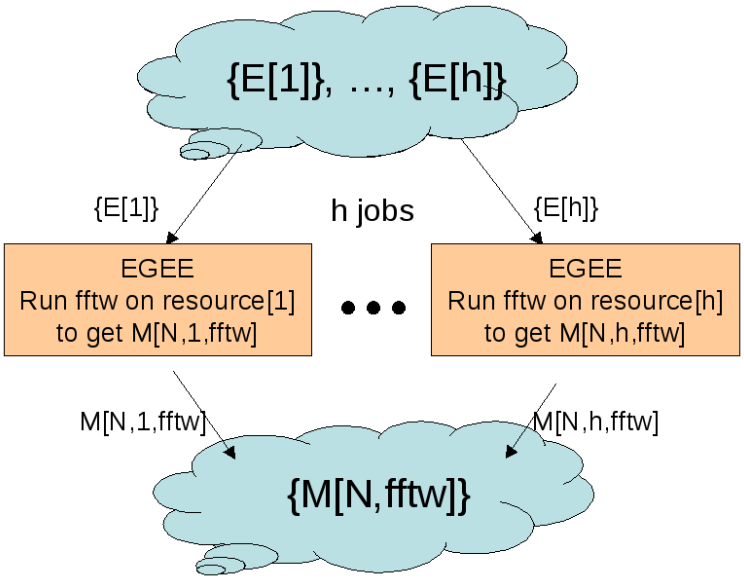


Figure 5.10: Model Construction: For the FFTW application on EGEE, where h is a subset of all of EGEE's resource providers. This yields a performance model for FFTW on every resource on EGEE.

Let us examine if there is a relationship between the results for the FFTW program (particularly the measured completion time of the FFTW program) and the results for the benchmarks. From examining these figures, it would appear that there is a relationship between the completion time of the FFTW results and the EPDhrystone, EPWhetstone and EPFlops benchmarks, namely that the shorter the completion time the higher results gained in the benchmarks. As mentioned in Section 5.3.2, with a larger sample size, the correlation between the EPFlops benchmarks and the corresponding FFTW completion time does appear more significant. However, the scatterplot of EPFlops, Figure 5.11, shows that the EPFlops results have a lower correlation coefficient than the EPDhrystone and EPWhetstone results, with 11 results significantly differing from the trend. On further examination of these results, there are no geographical, architectural or system-based similarities of the machines benchmarked in question. The machines are located in Italy, Portugal, Korea, Russia, and the UK,

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	3.725×10^{-01}	1.073×10^{-02}	34.72	2.0×10^{-16}
a_1	-1.580×10^{-08}	8.526×10^{-10}	-18.54	2.0×10^{-16}

Table 5.4: Linear regression summary of complex data for 1-d powers of 2 FFTW application executions, with an input size parameter of 2097152 and the EPDhrystone estimates as the independent variable.

with 32 bit and 64 bit Intel CPUs at varying speeds. They also vary with the number of CPUs available in each machine, but as the benchmarks only work on single cores this would not have had any affect.

A possible reason for these benchmark results lying so far from the other results could be due to some extra load on the machines at the same time as running the FFTW benchmarks. This would account for a good performance in the EPFlops benchmarking but slower FFTW completions times. A solution to this would be to gather more information about the system while running the benchmarks, e.g. CPU and memory loads, to check if other programs or services are using these resources at the same time as the benchmarks or application. This information could then be used to either ignore the results from machines where other programs are utilising the machine's resources, or to wait until such a time as the machine is free.

For a multiple linear model of the form $y = a_0 + a_1x_1 + a_2x_2 + a_3x_3$, the variables x_1, x_2 and x_3 must be independent of each other. A scatterplot of x_1, the EPDhrystone estimates, and x_2, the EPWhetstone estimates is drawn and can be seen in Figure F.9. A scatterplot of x_2, the EPWhetstone estimates, and x_3, the EPFlops estimates can be seen in Figure F.10 and a scatterplot of x_1, the EPDhrystone estimates, and x_3, the EPFlops estimates is drawn and can be seen in Figure F.11.

These figures show that there appears to be a linear relationship between each of the three estimates, making them not independent of each other..

To decide on which linear regression model to choose, either $y = a_0 + a_1x_1$, $y = a_0 + a_2x_2$ or $y = a_0 + a_3x_3$, the P values for each model must be calculated.

Solving for a_0 and a_1 in the three equations the resulting coefficients, standard errors, t values and corresponding (two-tailed) p-values are shown in Tables 5.4, 5.5 and 5.6 for complex data of 1-d FFTW application executions with an input size parameter of 2097152.

x_1, x_2 and x_3 are deemed statistically significant, but the adjusted r-squared values for each model are 0.74, 0.73 and 0.55 respectively. This means that the EPDhrystone linear model is the best fit to the data points and this regression model, using the EPDhrystone estimates, is slightly more accurate and would therefore model the FFTW

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	3.979×10^{-01}	1.217×10^{-02}	32.69	2.0×10^{-16}
a_1	-1.125×10^{-04}	6.129×10^{-06}	-18.36	2.0×10^{-16}

Table 5.5: Linear regression summary of complex data for 1-d powers of 2 FFTW application executions, with an input size parameter of 2097152 and the EPWhetstone estimates as the independent variable.

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	8.561×10^{-01}	5.567×10^{-02}	15.38	2.0×10^{-16}
a_1	-4.622×10^{-04}	3.081×10^{-05}	-12.16	2.0×10^{-16}

Table 5.6: Linear regression summary of complex data for 1-d powers of 2 FFTW application executions, with an input size parameter of 2097152 and the EPFlops estimates as the independent variable.

completion time better than the other two models.

With the values for a_0 and a_1 calculated, a prediction of FFTW completion time for 1-d complex data with an input size parameter of 2097152, can be made given a machine's EPDhrystone(M_d) benchmark estimates. The full equation is as follows:

$$\begin{aligned} y &= 3.725 \times 10^{-01} \\ &\quad +(-1.580 \times 10^{-08}) \times (M_d) \end{aligned}$$

By extending these benchmarks to all of the available EGEE grid resources and plotting them against the FFTW completion time results, there is a clear linear relationship shown between the two. With the increase in sample size the EPDhrystone benchmark is shown to be of greater significance in predicting the FFTW completion time. The decrease in the adjusted R-squared value of 0.8489 in the Grid-Ireland model to 0.74 for the EGEE grid results shows that, again with the increase in sample size, a greater variation from the linear model occurs. This variation may be due to other aspects of grid resources not being captured by the benchmarks that have been run. A multiple linear regression model including the benchmarking of more system resources could capture these variations, making the model more accurate.

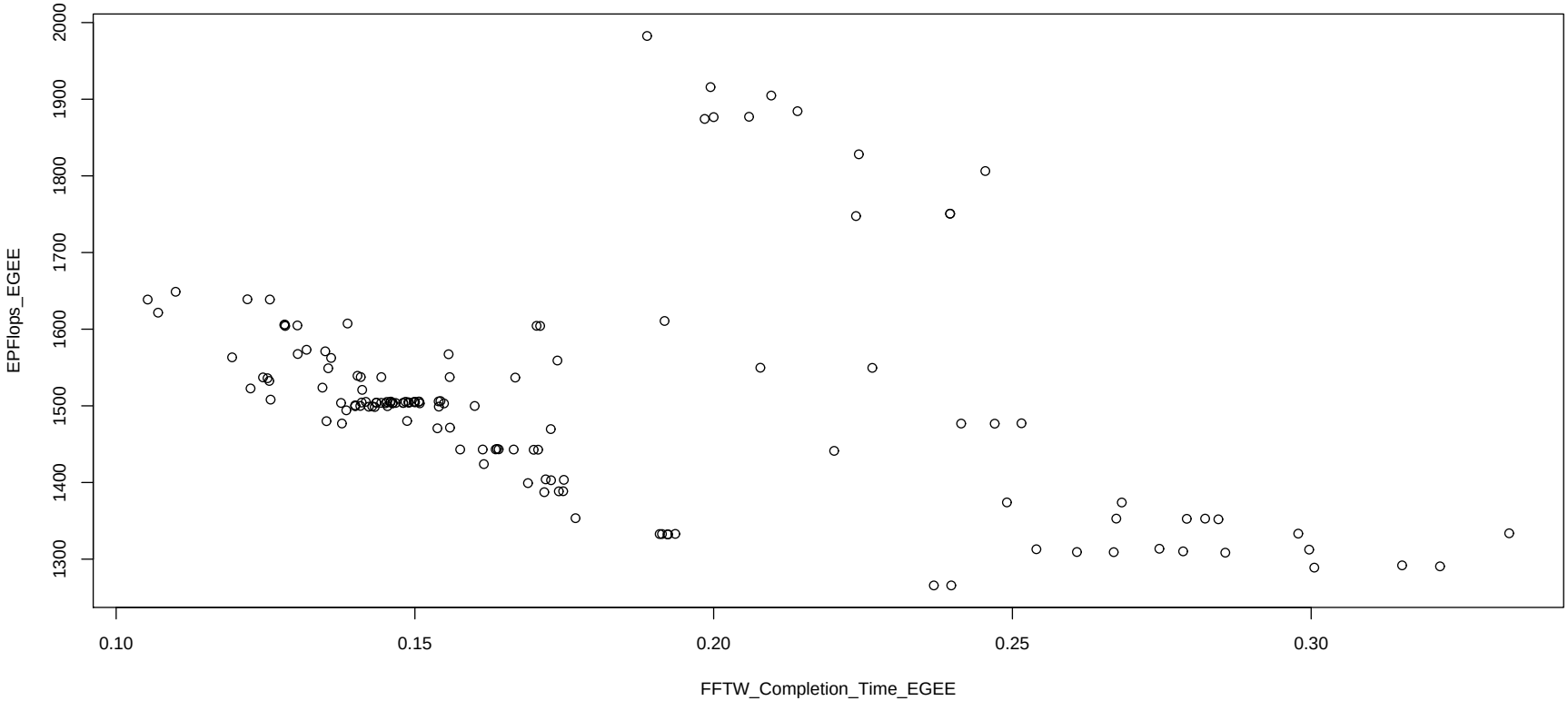


Figure 5.11: Estimates of EPFlops for each machine plotted against FFTW completion time.

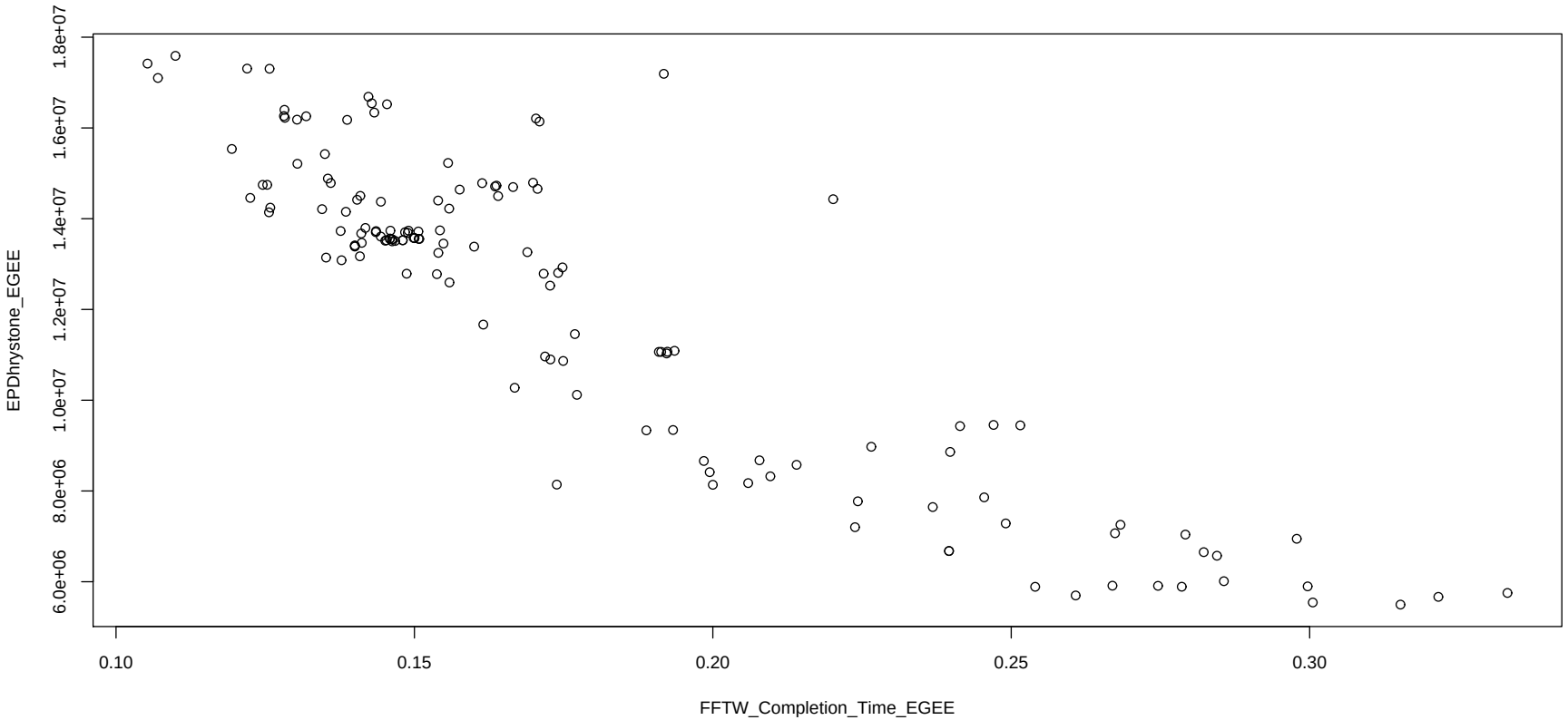


Figure 5.12: Estimates of Dhrystones from each machine against FFTW completion time.

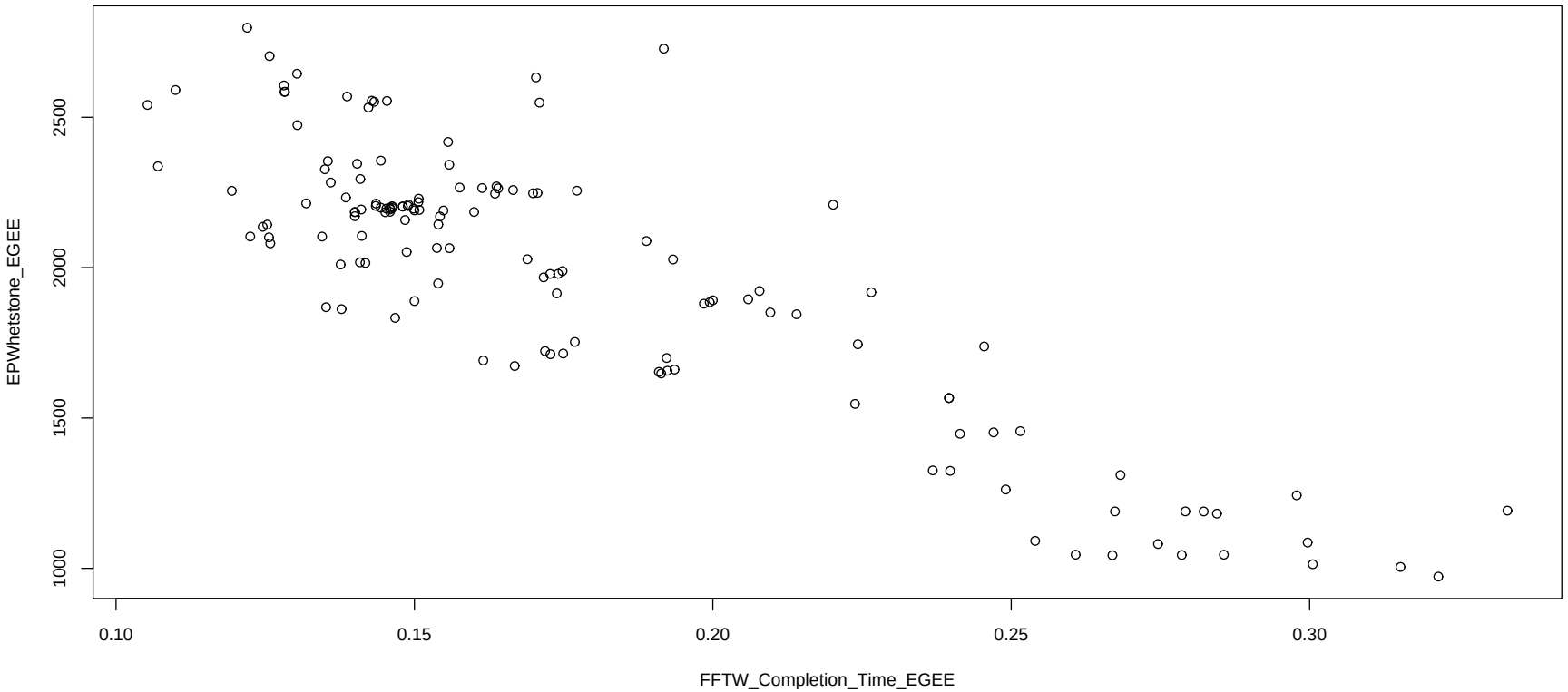


Figure 5.13: Estimates of EPWhetstone from each machine against FFTW completion time.

5.4 Example Fine-scale Resource Benchmarking

Grid-Ireland

At present Grid-Ireland has three clusters of dedicated visualisation resources:

- [gr065-gr68].grid.cs.tcd.ie: A cluster of 4 machines with dual-core AMD64 Opteron processor, 4GB of RAM and a GTX280 Nvidia PCI Express graphics card.
- vrengine: A cluster of 9 machines consisting of dual-core Pentium 4 3.20GHz processors, 1GB of RAM and a GeForce 6200 Nvidia PCI graphics card.
- [ps001-ps007].grid.cs.tcd.ie: A cluster of 7 PlayStation3 machines with PowerPC-base core 3.2GHz processor, 7 x SPEs 3.2GHz, 256MB of RAM and a 1.8TFLOPS floating point performance capable GPU.

For the purposes of the example fine-scale benchmarking and regression model construction only one of each machine in two clusters (gr065.grid.cs.tcd.ie and vrengine.cs.tcd.ie) is used. Possible ways of using more machines in a cluster is discussed in the future work section of Chapter 6.

Figure 5.14, which is derived from Figure 4.5, describes the way with which to benchmark Grid-Ireland's visualisation resources and obtain performance estimates.

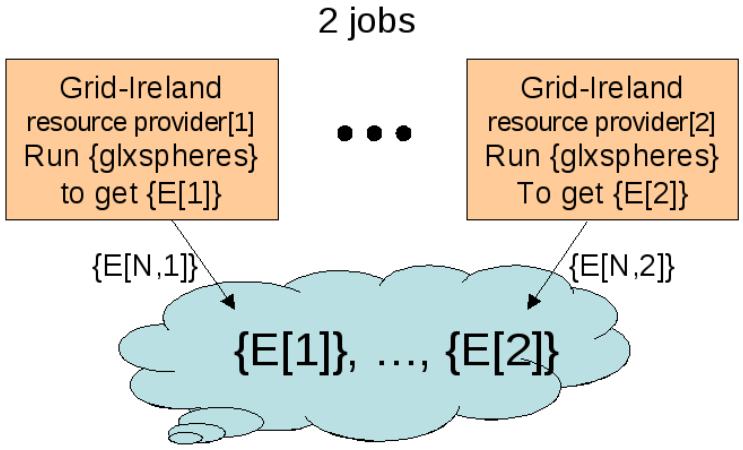


Figure 5.14: Resource benchmarking: For Grid-Ireland, with three (B=1) micro-benchmarks glxspheres, where Grid-Ireland has two fine-scale visualisation resource providers. This yields performance estimates E for the micro-benchmark on every fine-scale visualisation resource on Grid-Ireland.

The *glxspheres* benchmark is sent, along with a number of *bash* scripts written by the author, as a grid job to each of Grid-Ireland's visualisation resources. The scripts provide the staging mechanism with which to run *glxspheres* with a number of different inputs. The program is run with a combination of varying numbers of polygons and screen resolutions on a dynamic scene. As these jobs are intended to benchmark only the visualisation resource without any interference from other factors, such as network bandwidth, the *DISPLAY* environment variable is set to the machine's own screen, *localhost:0.0*.

Figure 5.15 shows the results of running the benchmark on *gr065.grid.cs.tcd.ie* and *vrengine.cs.tcd.ie*. As can be seen *gr065.grid.cs.tcd.ie* outperforms *vrengine.cs.tcd.ie* considerably. This is to be expected as the GTX280 Nvidia graphics card is state of the art at present.

Figure 5.16 has the same results as Figure 5.15 but is focused in towards the smaller number of polygon input sizes. It is interesting to observe the gaps between the start of the curves for different resolutions for each machine, and that *vrengine.cs.tcd.ie* has the wider gaps. This can be explained by the fact that the Nvidia graphics card on that machine is only a PCI card, so as the resolution increases, the slower bus reduces the speed at which it can output the images.

Most OpenGL applications are thought of as interactive but some are used to produce photorealistic images, for example ray tracing applications [172][173], over a longer period of time. These programs produce single frames at non-interactive rates which can be gathered together after execution to produce a high quality video.

Once the results from a benchmarked resource have returned from the grid it is possible to construct a linear regression model which can be used to infer the frames per second performance of such OpenGL applications running **locally** on this visualisation resource.

The relationship between polygon input size and frames per second output is exponential, therefore logarithms of both sides of the regression equation are needed, giving an equation of the form:

$$logy = a_0 + a_1 log(x_1)$$

where y is the frames per second output values and x_1 is the polygon input size. Before solving for a_0 and a_1, a linear relationship between the polygon input size and the frames per second output is expected. As increasing the work load for a graphics card, by increasing the number of polygons to be drawn, should result in a poorer frames per second output. Solving for a_0 and a_1 the resulting coefficients, standard error, t value and corresponding (two-tailed) p-value are shown in Table 5.7 for a resolution of 1024x768 on *gr065.grid.cs.tcd.ie*. The adjusted r-squared value of this equation is

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	6.18241	0.22662	27.28	1.87×10^{-11}
a_1	-1.74709	0.08674	-20.14	4.96×10^{-10}

Table 5.7: Linear regression summary of *glxspheres* running on *gr065.grid.cs.tcd.ie* at a resolution of 1024x768.

0.9712 which is close to 1.0, meaning that statistically the regression line fits well with the data. Regression models have been constructed for the other resolution sizes, 320x240, 640x480 and 800x600, and are plotted and shown in Figure 5.17.

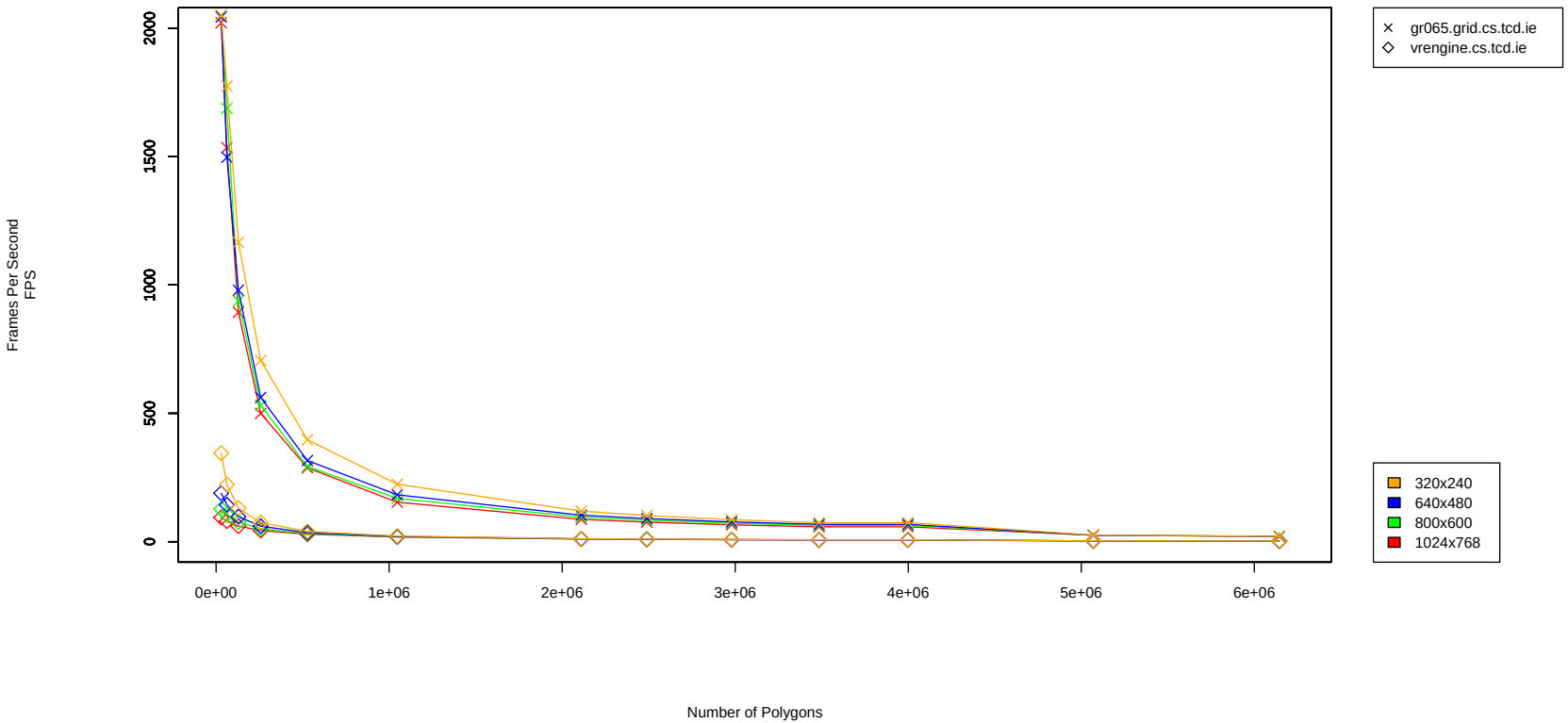


Figure 5.15: Frames per second achieved by glxspheres with varying polygon input size and resolution running on *gr065.grid.cs.tcd.ie* and *vrengine.cs.tcd.ie*.

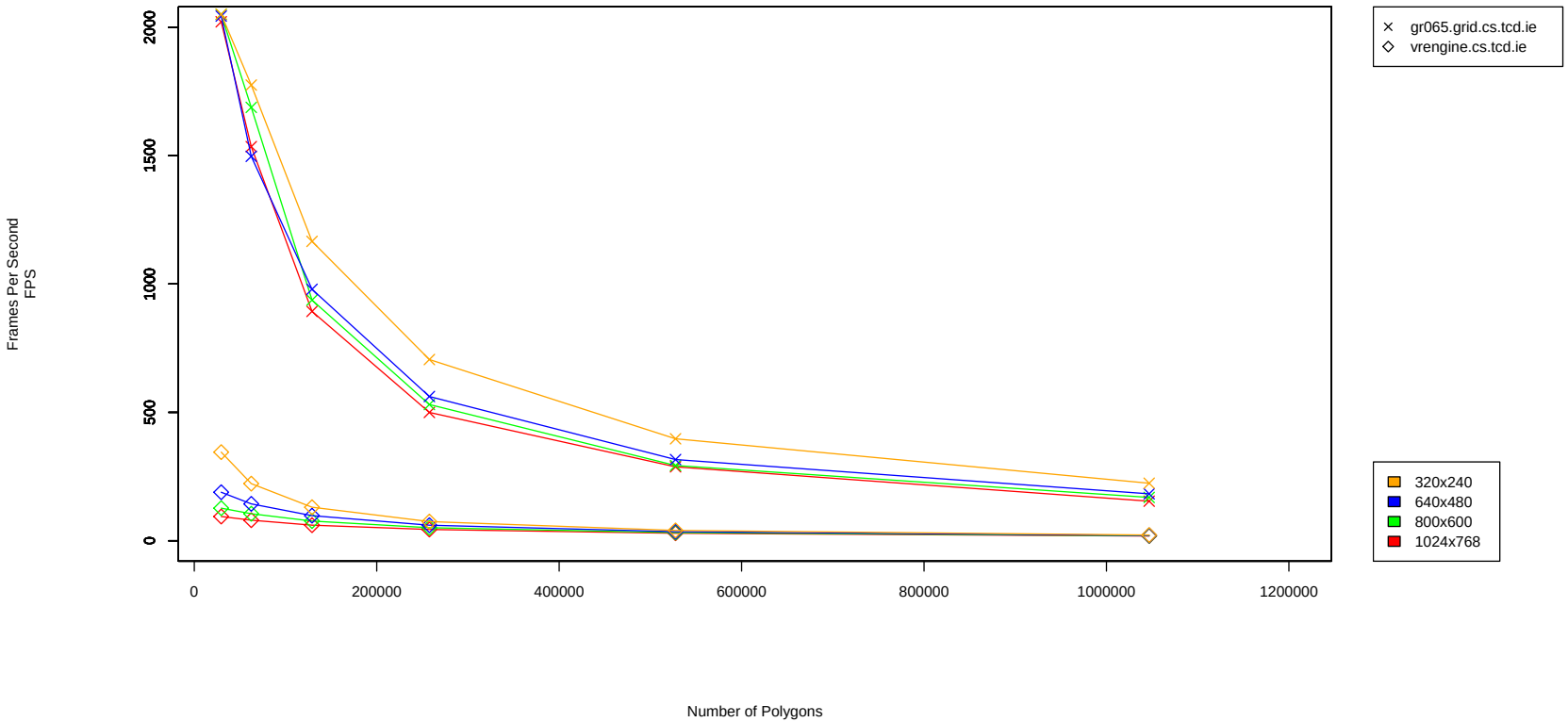


Figure 5.16: Frames per second achieved by glxspheres with varying polygon input size and resolution running on *gr065.grid.cs.tcd.ie* and *vrengine.cs.tcd.ie*.

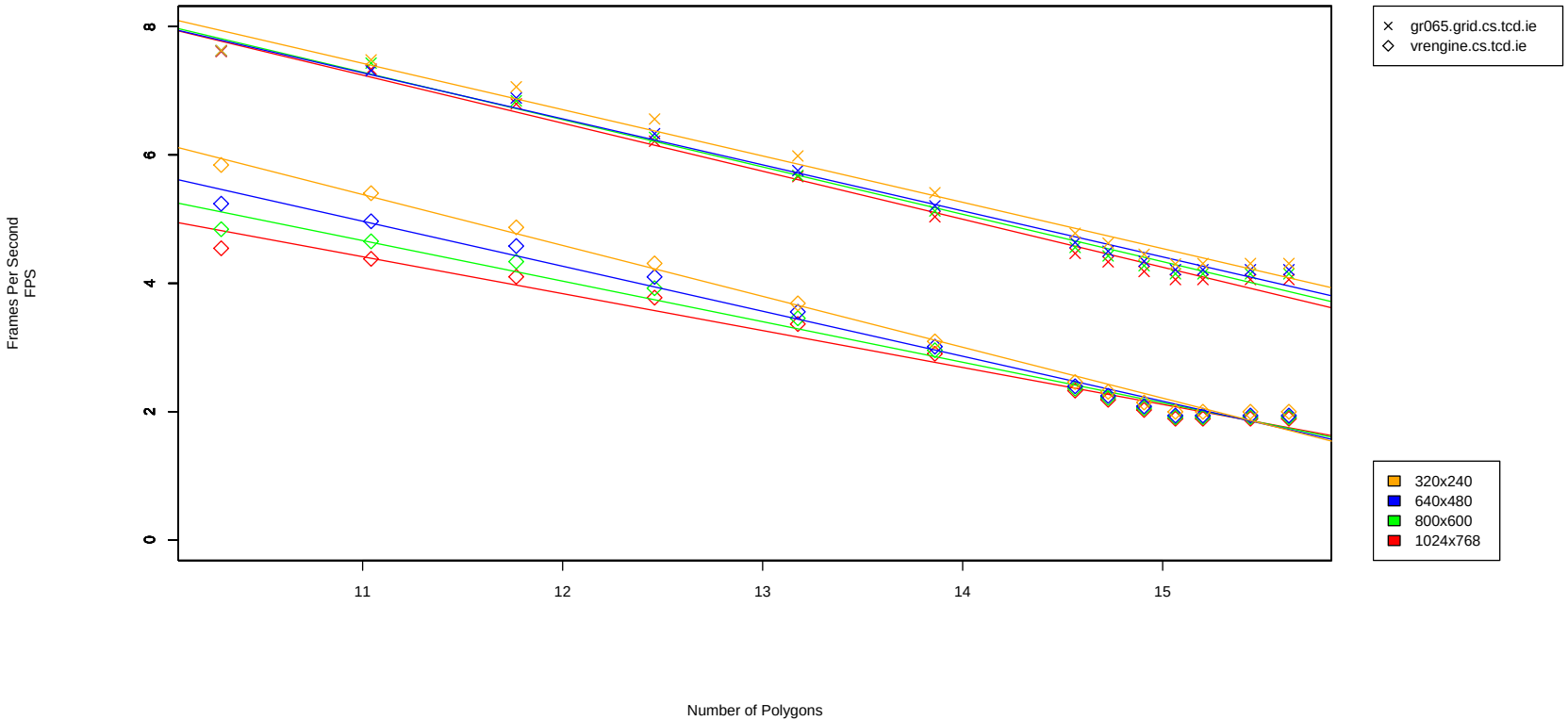


Figure 5.17: Linear regression lines of *glxspheres* running *gr065.grid.cs.tcd.ie* and *vrengine.cs.tcd.ie* with varying polygon input size and resolution.

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	15.67633	0.47226	33.19	2.0×10^{-16}
a_1	-0.77599	0.03369	-23.03	2.0×10^{-16}
a_2	-0.30846	0.02159	-14.29	2.0×10^{-16}

Table 5.8: Multiple linear regression summary of glxspheres running on *gr065.grid.cs.tcd.ie* at a resolution of 1024x768 with multiple users.

As the visualisation resource is a grid resource shared by more than one person an examination must be made of the effect of multiple simultaneous users on the regression model. Figure F.12 shows the frames per second achieved by *glxspheres* with varying polygon input size, at a resolution of 1024x768 and 2, 4 and 8 simultaneous users respectively. As this data appears to be of an exponential shape, the logarithm of both input and output values is calculated, and a scatter plot of the results is shown in Figure F.13. From this figure it is clear that there is a linear relationship between the frames per second achieved, the polygon input size and the number of users. The multiple linear regression equation will be of the form:

$$logy = a_0 + a_1 log(x_1) + a_2(x_2)$$

where y is the frames per second output value, x_1 is the polygon input size and x_2 is the number of users. Solving for a_0, a_1 and a_2 the resulting coefficients, standard error, t value and corresponding (two-tailed) p-value are shown in Table 5.7 for a resolution of 1024x768 on *gr065.grid.cs.tcd.ie*. The adjusted r-squared value of this equation is 0.9349 meaning that statistically the regression line fits well with the data.

The resulting multi-linear regression equation for glxspheres running on *gr065.grid.cs.tcd.ie* with multiple users is as follows:

$$\begin{aligned} log(y) &= 15.67633 \\ &\quad +(-0.77599) \times log(PolygonInputSize) \\ &\quad +(-0.30846) \times (NumberofUsers) \end{aligned}$$

These coefficients can now be published and used by grid users to determine which is the best visualisation resource for the render jobs.

5.5 Example Fine-scale Model Construction

With a multiple linear regression model constructed for each visualisation grid resource at varying resolutions it is necessary to examine how an interactive visualisation application performs. To allow for an interactive session between the user's desktop and the visualisation resource, VirtualGL [174] is used. VirtualGL is an open source package which gives any Unix or Linux remote display software the ability to run OpenGL applications with full 3-d hardware acceleration by redirecting the OpenGL commands and 3-d data to the 3-d graphics card on the visualisation resource, so only the rendered 3-d images are sent to the client machine. The images and interaction are tunnelled, using *ssh*, through the grid User Interface (UI) to the user's desktop.

For interactive applications VirtualGL has an option to allow frame spoiling to occur. By default, VirtualGL will only send a frame to the client if the client is ready to receive it. If a rendered frame arrives at the server's queue and a previous frame is still being processed, the new frame is dropped ('spoiled'). This prevents a backlog of frames on the server, which would cause a perceptible delay in the responsiveness of interactive applications. But when running non-interactive applications it is desirable to disable frame spoiling. With frame spoiling disabled, the server will render frames only as quickly as VirtualGL can send those frames to the client, which will conserve server resources as well as allow OpenGL benchmarks to accurately measure the frame rate of the VirtualGL system.

The choice of an example application to construct a fine-scale model is again *glxspheres*, as it has all the necessary attributes of an interactive OpenGL application. Figure 5.18 shows the grid jobs that need to be run in order to obtain a performance model for the *glxspheres* application.

5.5.1 glxspheres Model Construction for the Grid-Ireland Fine-scale Resources

To examine how *glxspheres* performs when the rendered image output has to be transported to the user's desktop a number of issues have to be considered. Firstly, enabling/disabling frame spoiling will have an effect; secondly, the fine-scale (rendering) resources were designed to support multiple users; and thirdly, the network bandwidth between the user's desktop and the visualisation resource might affect the achievable frames per second relative to what would be achieved on the user's desktop alone. Two desktop machines, *picolet.cs.tcd.ie* and *bacchus.cs.tcd.ie*, were used to test this.

Figure F.14 shows the resulting performance when running *glxspheres* with varying polygon input size and resolution running on *picolet.cs.tcd.ie* and *bacchus.cs.tcd.ie*.

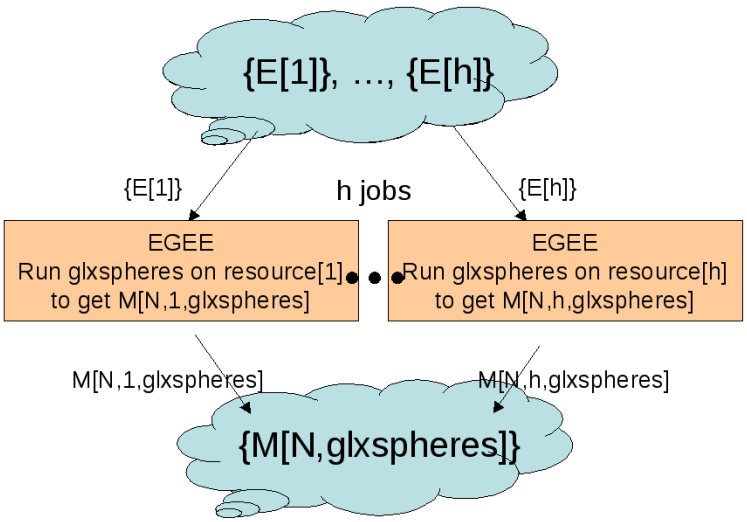


Figure 5.18: Model Construction: For the glxspheres application on Grid-Ireland, where Grid Ireland has 2 fine-scale visualisation resource providers. This yields a performance model for glxspheres on every fine-scale visualisation resource on Grid-Ireland.

The performance of both machines is very poor with only *bacchus.cs.tcd.ie* managing a real time frame rate of over 30fps at the lowest resolution and then only at the smallest polygon input size of 29,524.

Figures F.15 and F.16 show the resulting frames per second performances of *glxspheres* with varying polygon input size and resolution running on *gr065.grid.cs.tcd.ie* and *vrengine.cs.tcd.ie*, with the frame-spoiling VirtualGL option enabled and the rendered output sent to *picolet.cs.tcd.ie* and *bacchus.cs.tcd.ie*.

As can be seen from these figures, the performance gained when running *glxspheres* on the resources available on the grid is greatly improved relative to that of the desktop machines. To construct a multiple linear regression model for an interactive application of a visualisation resource that is shared by multiple users, grid jobs for a number of users must be executed. Figures F.17 and F.18 show the results from running *glxspheres* on the *gr065.grid.cs.tcd.ie* visualisation resource for two and three users. Clearly the performance levels, as expected, decrease as more users run the application on the visualisation resource, but with the construction of a multiple linear regression equation for an application, an application developer can tell how many users can run the application simultaneously on a visualisation resource and still achieve a real time frame rate of over 30fps.

Taking the results from these grid jobs a regression equation, which will be able to model the relationship between the frames per second output of *glxspheres*, the polygon

Coefficient	Estimate	Standard Error	T Value	P Value
a_0	11.14842	0.42433	26.27	$\times 10^{-16}$
a_1	-0.43813	0.02939	-14.91	2.0×10^{-16}
a_2	-0.71616	0.06184	-11.58	1.07×10^{-13}

Table 5.9: Multiple linear regression summary of glxspheres running on *gr065.grid.cs.tcd.ie* at a resolution of 1024x768 with multiple users, and with the rendered images being interactively sent to the grid user's desktop.

input size and the number of users, will take the form:

$$logy = a_0 + a_1 log(x_1) + a_2(x_2)$$

where y is the frames per second output values, x_1 is the polygon input size and x_2 is the number of users. Solving for a_0, a_1 and a_2 the resulting coefficients, standard error, t value and corresponding (two-tailed) p-value are shown in Table 5.9 for a resolution of 1024x768 on *gr065.grid.cs.tcd.ie*. The adjusted r-squared value of this equation is 0.9031 meaning that statistically the regression line fits well with the data.

The resulting multi-linear regression equation for *glxspheres* running interactively on *gr065.grid.cs.tcd.ie* with multiple users and the rendered images being sent to the user's desktop is as follows:

$$\begin{aligned} log(y) &= 11.14842 \\ &\quad +(-0.43813) \times log(PolygonInputSize) \\ &\quad +(-0.71616) \times (NumberofUsers) \end{aligned}$$

These coefficients can be published and employed by grid users to determine which is the best available visualisation resource for running *glxspheres*.

For example, if the user wanted to run the *glxspheres* application at a resolution of 1024x768, with a real time frame-rate of 30fps or above, and with a maximum of 2 simultaneous users, the equation could be evaluated to find the maximum size polygon input size to achieve this. It would be evaluated as follows:

$$
\begin{aligned}
log(30) &= 11.14842 \\
&\quad +(-0.43813) \times log(PolygonInputSize) \\
&\quad +(-0.71616) \times (2) \\
6.3149 &\geq (0.43813) \times log(PolygonInputSize) \\
PolygonInputSize &\leq e^{14.414} \\
PolygonInputSize &\leq 1,812,105
\end{aligned}
$$

Therefore to achieve a real time frame rate of 30fps or above for two simultaneous users, the polygon count of the application must stay less than or equal to 1,812,105.

The multiple regression equation can be rearranged to give:

$$P_{RealTime} \leq e^{17.6826-(1.634 \times N_{Users})}$$

where $P_{RealTime}$ is the maximum number of polygons, for N_{Users}, that will achieve a real time frame rate running *glxspheres* as an interactive grid job on *gr065.grid.cs.tcd.ie*.

The above methods can be used by application developers to produce a similar set of coefficients which will describe how their specific application will run on grid visualisation resources, and if this published then users can clearly exploit this information.

The network bandwidth between the grid user's desktop and the visualisation resource can be represented by another regression coefficient in the multiple linear regression model constructed above. This becomes very apparent when analysing the results of running *glxspheres* with the frame-spoiling option disabled. Disabling this option means that the visualisation resource will wait for the client (i.e. user's desktop machine) to receive and decode each frame until it sends another one. This means that the network becomes a possible limitation on the frames per second that can be achieved. Figures F.19, F.20 and F.21 show the results of running *glxspheres*, by multiple users, with the frame-spoiling option disabled. It can be seen from these figures that there is a bottleneck in the performance output of the application as the resolution size increases. Due to the fact that the visualisation resources that were used in this example fine-scale model construction were located in the Grid-Ireland OpsCentre machine room, only 30 metres away on a 1Gbps Ethernet connection, it was not possible to observe the effect varying network bandwidths would have, thus making it impossible to obtain measurements that could help construct a multiple linear regression model.

5.6 Summary

An example application was chosen to demonstrate the construction of a model for a coarse-scale/mid-scale visualisation resource. This application was the FFTW program which computes the discrete Fourier transform in one or more dimensions. This is a CPU intensive application and by executing it on a number of coarse-scale/mid-scale resources on Grid-Ireland, a multiple linear regression model was constructed, with the use of the statistically significant EPWhetstone estimates. This model was then used to enable the author to accurately predict the completion time of a FFTW program execution on a grid resource outside of Grid-Ireland.

By extending these benchmarks to all of the available EGEE grid resources and plotting them against the FFTW completion time results, there is a clear linear relationship shown between the two. The decrease in the adjusted R-squared value from the Grid-Ireland model to the EGEE grid model shows that with the increase in sample size, a greater variation from the linear model occurs. A multiple linear regression model that includes the benchmarking of other aspects of a system could capture these variations, making the predictions more accurate.

Another example application, *glxspheres*,was chosen to benchmark and help construct a multiple linear regression model that predicted the frames per second output of a fine-scale visualisation resource. It has also been shown that these multiple linear regression equations, if rearranged, can be used to help grid users and application developers in their parameter selections.

Finally, my thanks to all those who gave permission and assistance that enabled benchmarking on grid resources in Grid-Ireland, UK NGS, IFCA, NIKHEF, PSNC and EGEE: Dr.Brian Coghlan (Director of the Grid-Ireland OpsCentre) and John Walsh (Grid Manager of Grid-Ireland), Dr.Andy Richards (Director of UK NGS), Prof. Jesus Marco (Director of GRID-CSIC), Dr.Isabel Campos Plasencia and Pablo Orviz of IFCA in Spain, Dr.David Groep (Leader of Grid Security Middleware Development) of NIKHEF in the Netherlands, Dr.Norbert Meyer (RINGrid project coordinator) and Marcin Pospieszny of PSNC in Poland, Dr.Bob Jones (Director, EGEE) and Dr.Maite Barroso Lopez (SA1 Manager, EGEE-III), and all the EGEE Regional Operations Centre (ROC) managers for permission to use their site resources. Especial thanks to John Walsh for vetting the benchmarking code, etc, as a prelude to requesting these permissions.

Chapter 6

Conclusions and Future Work

6.1 Introduction

This final chapter presents a number of possible extensions to the vision and the work of this thesis, the uses of the model and a discussion of future work to the contributions of the work.

6.2 Conclusions

The motivation of this thesis was to consider a framework for visualisation that gives a greater performance than a desktop computer can provide, achieved at a cost much cheaper than a CAVE, while allowing simultaneous usage by multiple users, and to do so on a mathematical basis.

This thesis presented a methodology that has been developed to share these visualisation facilities on demand in the same way other resources are shared for batch submission style problems, described in Chapter 3, and to do so on a mathematical basis as described in Chapter 5. A comprehensive application, VirtualGrid, which tests this framework thoroughly was also described within this thesis. This application also provides a useful example for grid users and developers to help understand better the architecture and components of the framework and its models.

The model does not distinguish resources at all, e.g. by architecture or physical attributes like memory or the number of cores on a CPU, but instead tries to build its own profile of a resource by running appropriate benchmarks aimed at specific types of grid resources, i.e. *glxspheres* benchmarking a fine-scale visualisation resource. By benchmarking a large enough sample size of differing grid resources, and running an application on a subset of these resources, the model can be used to provide a rough

estimate of the performance of all grid resources running that application.

This is not an ideal solution to providing grid resource performance information, as for example, it is not possible to predict how a given architecture of a coarse-scale grid resource, with a particular amount of memory, will perform running a simulation. But once the given coarse-scale resource has run a set of micro-benchmarks, the model can give a rough prediction of the performance outputs of a resource.

Ideally, future models would be based on architectures and physical attributes by running grid jobs that collect information from the resource, like the amount of memory and number of CPUs available. However for this to be possible, not only does the sample size of grid resources from which the information is gathered have to be large, but it also must contain a significant variation of resource specifications to allow a model to be constructed. The problem with this is that a lot of grid sites are built with types of machines which have similar, if not identical, specifications. An example of the problem this can cause can be seen in Section 5.3.2 where the linear regression model is skewed due to similar estimates obtained from a number of Grid-Ireland machines. An answer to this would be to perform some offline benchmarking of machines where it would be possible to vary the amount of memory, type of graphics card and the CPU of a given machine. It this way an exact profile could be built which would allow for the performance prediction of various architectures. Such future models might also make use of polynomials to profile more accurately coarse-scale, mid-scale and fine-scale resources.

This thesis has focussed on models at the coarse-scale/mid-scale and fine-scale levels, without considering the combination of all these. It should be possible to construct a model of the interaction loop that occurs when interactive applications are run on a grid. This model would need to consider, among other variables, the network bandwidths involved in modelling an interaction loop. An example of such a loop is shown in 3.3, where model data and steering instructions need to be communicated between coarse-scale, mid-scale and fine-scale resources. The combinations of interactive visualisation models running on fine-scale resources and computation models running on coarse-scale/mid-scale resources would have to be explored. Figure 6.1 shows an example set of combinations between the user's desktop, coarse-scale/mid-scale and fine-scale resources.

These extensions would build upon the work of this thesis and more accurately help describe the complicated interaction between the heterogeneous resources harnessed by the visualisation framework.

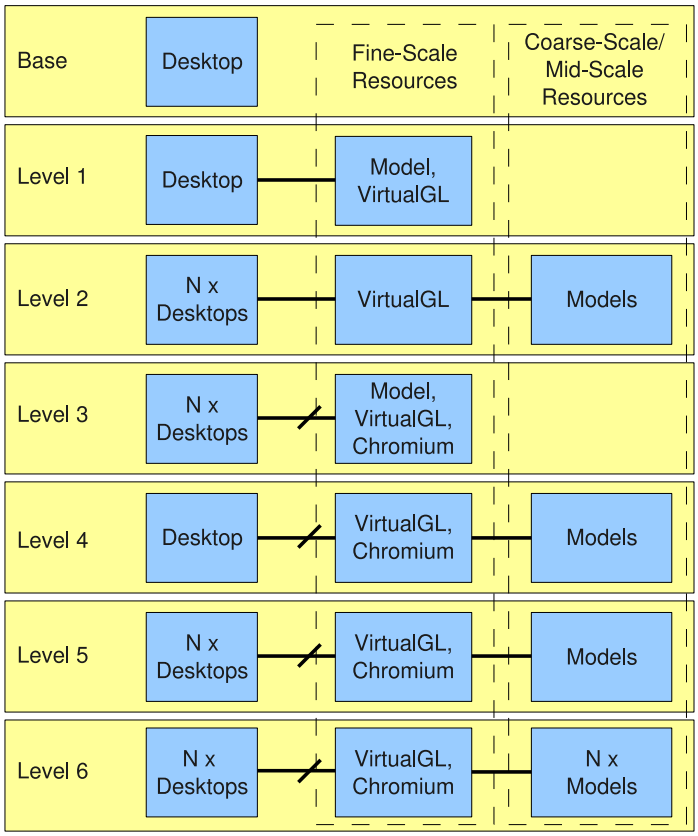


Figure 6.1: A possible set of combinations of simulation and rendering models running on coarse-scale, mid-scale and fine-scale resources.

6.3 Uses of the Model

Given a model that has been populated with experimentally measured results for specific combinations of useful architectural blocks, this may be employed in at least the following ways.

6.3.1 Specification of a Visualisation Resource Provision

The model can be used to assist in the specification of a visualisation resource, e.g. rendering facility, say for procurement purposes. Whether the facility is local versus remote (relative to the user) and single-user versus multi-user has the following effects:

- If it is a local single-user facility then both the interaction loop and the render engine model are quite deterministic.

- If it is a local multi-user facility then the interaction loop is quite deterministic but the render engine model is quite non-deterministic.
- If it is a remote multi-user facility then both the interaction loop and the render engine model are quite non-deterministic.

A resource provider that wishes to provide a new rendering facility can exhaustively evaluate the model for all combinations of architectural blocks to yield a fully populated quality-of-service (QoS) space. They can then employ this to enumerate their design in two ways:

- If they already have very definite specifications for the black-box QoS requirements of the proposed facility then they can utilise the QoS space to identify the feasible combinations of architectural blocks that would meet those requirements.
- If they do not have pre-defined QoS requirements, then they make a value judgement on what QoS would be useful within their business model, and identify the feasible combinations of architectural blocks.

In both cases this might then enable them to choose the optimal combination based on other criteria like cost, energy use, space needs, etc. The argument equally applies to coarse-scale or mid-scale simulation resources.

6.3.2 User-selection of a Remote Rendering Resource

The model could be explicitly or implicitly employed by a user to either manually or programmatically select a remote shared coarse-scale, mid-scale or fine-scale visualisation resource based on their needs:

- If the the resource providers publish the composite QoS characteristics of their resources, i.e. the coefficients of the multiple linear model, then the user may select an appropriate resource and subsequently target that resource. In this case the user is implicitly employing the model, where for their specific combination of architectural blocks the resource providers have either evaluated the model or have directly measured the metrics defined by the model.
- If the the resource provider publishes the specific combination of architectural blocks for their resources then the user may explicitly evaluate the model based on the published information and select an appropriate resource and subsequently target that resource.

The GLUE schema [175][176], which provides a way of publishing information about resources on a grid, could be extended to allow for the benchmark estimates of coarse-scale, mid-scale or fine-scale resources to be made available for a grid user or application developer to use in either deciding which available resource is best for their job, or for an application developer to build a model for their specific program. These estimates could be added to the attributes of a Compute Element (CE) with suggested GLUE schema identifiers as follows:

- *GlueCEInfoEPDhrystoneEstimate*: estimate for the EPDhrystone micro-benchmark.
- *GlueCEInfoEPWhetstoneEstimate*: estimate for the EPWhetstone micro-benchmark.
- *GlueCEInfoEPFlopsEstimate*: estimate for the EPFlops micro-benchmark.
- *GlueCEPolicyMaxRunningVisualisationJobs*: maximum allowed number of running visualisation jobs.
- *GlueCEStateFineScaleUsers*: number of current interactive visualisation users
- *GlueCEStateMaxFreePolygonsRealTime*: the current maximum number of polygons that the resource can render with a real time frame rate. This attribute is the result of evaluating the multiple linear equation of Section 5.5.1 with the *GlueCEStateFineScaleUsers* attribute as an input argument.

According to Michel Jouvin of IN2P3 in France: "The main use of benchmarks is to compute/assess/pledge the relative contribution of each site for a particular community. Current Glue v1 doesn't allow to do this but we can hope that glue v2 will add this flexibility."[177]

6.3.3 Basic Brokerage of Visualisation Resources

Distributed Computing infrastructures, like EGEE or EGI, can employ automatic services (brokers) that choose the best available resources for a job (i.e. do resource allocation). There are many criteria that can be used to guide the service towards an optimal allocation. The models in the thesis are expected to be especially useful in this regard, for example:

- **Architecture-based:** The GLUE schema has an attribute, *GlueHostArchitecturePlatformType*, which states the architecture of a site and so, in future, if an architecture based model is constructed, then this attribute could be used by the gLite Workload Management System (WMS) to determine the best available resource for a coarse or mid-scale simulation.

- **QoS-based:** An 'extended' JDL could guide selection of a grid resource based on the above extended GLUE schema. This would allow resources to be selected by an enhanced gLite WMS. An example 'extended' JDL file, shown as follows, specifies as a requirement that the visualisation job be sent to a fine-scale resource that can take an extra load of 200,000 polygons and still maintain a real time frame rate.

```
[   Executable = "run_visualisation.sh";
    Arguments  = "-polygons 200000";
    StdOutput  = "std.out";
    StdError   = "std.err";
    InputSandbox = {
        "./run_visualisation.sh",
        "./glxspheres1024x768"
    };
    OutputSandbox = {
        "std.out",
        "std.err",
    };
    Requirements = other.GlueCEStateMaxFreePolygonsRealTime > 200000
    RetryCount = 3;
    JobType = "normal";
    Type = "Job";]
```

- **Economic Market-based:** An economic market model could also be constructed based on these benchmarks which could determine:
  - **Value:** The value of a multi-user fine-scale (rendering) resource is clearly the remaining available frames per second (which will reduce whenever an extra user employs the resource).
  - **Price:** The price is what the owner of the fine-scale resource puts on frames per second, for example, in an economic market the owner might set the price to zero for friends or charities, to discounted prices for like-minded consortia or Government-funded work, or full market value for commercial work.

  Economic market models for grids have been examined by Dr. Gabriele Pierantoni, of the author's research group, and he has examined them in his Ph.D. thesis [178].

In the case of coarse-scale and mid-scale simulation resources the model would be of value to all users with simulation jobs, whereas in the case of fine-scale (rendering) resources the model is specifically of value to users with visualisation jobs.

6.4 Contributions of the Work

The contributions of this work consist of:

- The approach to a generic visualisation framework that is amenable to characterisation with mathematical models.
- A multiscale multimodal multiuser grid-enabled visualisation framework that takes advantage of breaking up the visualisation pipeline and distributing the workload to grid resources that are more suited to certain pieces of the pipeline (so that these resources are utilised to their fullest potential), thus making the visualisation more efficient.
- The development of VirtualGrid which is an example of the use of the visualisation framework, and the components of this application.
- An analysis of statistical models for the framework and the use of multiple linear regression models to help predict the performance of coarse-scale, mid-scale and fine-scale grid resources.
- A method of resource benchmarking and application model construction on multiple grids.
- A modified performance ontology that includes visualisation nodes and their associated metrics which helps classify grid resources so that they can be benchmarked according to appropriate benchmarks.
- A method of running a set of micro-benchmarks that produces estimates of the performance of a resource in relation to that micro-benchmark.
- The building of a resource profile for grid resources that describe the performance of a resource with respect to its computational power and/or rendering capabilities. These estimates can then be used to help explain the performance variance of an application across other grid resources.
- A way of using multiple linear regression equations which can be used to help grid users and application developers in their parameter selection guidelines.
- A roadmap of future work that can drive forward the evolution of the framework and its models.

6.5 Summary

To summarise this thesis I would like to say that, even though it has been a long and arduous effort over the last five years, with its highs and lows, I feel that overall, the positives have far outweighed the negatives and that I have achieved and learned a number of important things about this subject and about myself along the way. I have achieved what I intended to in writing this thesis and I feel that I have gained a professional and personal sense of development. I look forward to the future evolution of this subject and, even though I may not continue following this subject as closely as I have done for the last number of years, I will still maintain my interest in the areas of visualisation, grids and the complex combination of both.

Bibliography

[1] F. Gagliardi and M. E. Begin, "EGEE - providing a production quality grid for e-science," *International Symposium on Mass Storage Systems and Technology*, vol. 0, pp. 88–92, 2005.

[2] B. Coghlan, E. Fernndez, E. Heymann, P. Heinzlreiter, S. Kenny, M. Owsiak, I. C. Plasencia, M. Piennik, H. Rosmanith, M. Senar, S. Stork, and R. Valles, "Int.eu.Grid Project Approach on Supporting Interactive Applications in the Grid Environment," pp. 435–445, 2009.

[3] J. Walsh and B. Coghlan, "The Grid-Ireland National Grid Infrastructure," in *4th Iberian Grid Infrastructure Conference (IBERGRID'2010)*, (Braga, Portugal), 2010.

[4] T. Akenine-Móller and E. Haines, *Real-Time Rendering, Third Edition*. A K Peters, 2008.

[5] R. Watson, S. Maad, and B. Coghlan, "Multiscale multimodal visualization on a grid," in *Cracow Grid Workshop*, (Cracow, Poland), 2006.

[6] J. Morrison, B. Coghlan, A. Shearer, S. Foley, D. Power, and R. Perrot, "Webcom-g: A condidate middleware for grid-ireland," *High Performance Computing Applications and Parallel Processing*, 2005.

[7] J. Gomes, M. David, J. Martins, L. Bernardo, A. Garcia, M. Hardt, H. Kornmayer, J. Marco, R. Marco, D. Rodriguez, I. Diaz, D. Cano, J. Salt, S. Gonzalez, J. Sanchez, F. Fassi, V. Lara, P. Nyczyk, P. Lason, A. Ozieblo, P. Wolniewicz, M. Bluj, K. Nawrocki, A. Padee, W. Wislicki, C. Fernandez, J. Fontan, Y. Cotronis, E. Floros, G. Tsouloupas, W. Xing, M. Dikaiakos, J. Astalos, B. Coghlan, E. Heymann, M. Senar, C. Kanellopoulos, A. Ramos, and D. Groen, "Experience with the International Testbed in the CrossGrid Project," in *Advances in Grid Computing - EGC 2005*, vol. 3470 of *Lecture Notes in Computer Science*, pp. 98–110, 2005.

[8] R. Watson, S. Maad, and B. Coghlan, "Virtual grid: Adaptive visualization of grids," in *Cracow Grid Workshop*, (Cracow, Poland), 2007.

[9] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, pp. 115–128, Summer 1997.

[10] I. Foster and C. Kesselman, "The grid," ch. The Globus toolkit, pp. 259–278, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[11] G. Allen, T. Dramlitsch, I. Foster, N. T. Karonis, M. Ripeanu, E. Seidel, and B. Toonen, "Supporting efficient execution in heterogeneous distributed computing environments with cactus and globus," in *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, Supercomputing '01, (New York, NY, USA), pp. 52–52, ACM, 2001.

[12] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," *Journal of Computer Science and Technology*, vol. 21, pp. 513–520, 2006. 10.1007/s11390-006-0513-y.

[13] "The Globus Project." http://www.globus.org/. Last accessed, October 2009.

[14] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A resource management architecture for metacomputing systems," *Lecture Notes in Computer Science*, vol. 1459, 1998.

[15] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," in *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, pp. 365–375, IEEE Computer Society Press, 1997.

[16] A. S. Grimshaw and W. A. Wulf, "The Legion vision of a worldwide virtual computer," *Commun. ACM*, vol. 40, no. 1, pp. 39–45, 1997.

[17] "Legion." http://legion.virginia.edu/. Last accessed, October 2009.

[18] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience.," *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[19] "The Condor Team at the University of WisconsinMadison." http://www.cs.wisc.edu/condor/. Last accessed, October 2009.

[20] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," *IEEE Symposium on High Performance Distributed Computing*, vol. 10, 2001.

[21] "The Condor-G Project." http://www.cs.wisc.edu/condor/condorg. Last accessed, October 2009.

[22] G. Staples, "TORQUE resource manager," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, (New York, NY, USA), ACM, 2006.

[23] "The Torque Resource Manager." http://www.clusterresources.com/products/torque-resource-manager.php. Last accessed, October 2009.

[24] J. P. Jones, "Beowulf Cluster Computing with Windows," ch. PBS: portable batch system, pp. 363–383, Cambridge, MA, USA: MIT Press, 2002.

[25] "Portable Batch Submission (PBS)." http://www.openpbs.org. Last accessed, October 2009.

[26] S. Zhou, "LSF: Load sharing in large-scale heterogeneous distributed systems," in *Workshop on Cluster Computing*, (Tallahassee, FL), December 1992.

[27] "Load Sharing Facility." http://www.platform.com/Products/platform-lsf. Last accessed, October 2009.

[28] "Unicore." http://www.unicore.eu/. Last accessed, October 2009.

[29] V. Huber, "UNICORE: A Grid Computing Environment for Distributed and Parallel Computing," in *Proceedings of the 6th International Conference on Parallel Computing Technologies*, PaCT '01, (London, UK), pp. 258–265, Springer-Verlag, 2001.

[30] C. Marco, C. Fabio, D. Alvise, G. Antonia, G. Francesco, M. Alessandro, M. Moreno, M. Salvatore, P. Fabrizio, P. Luca, and P. Francesco, "The gLite Workload Management System," in *Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing*, GPC '09, (Berlin, Heidelberg), pp. 256–268, Springer-Verlag, 2009.

[31] "gLite Middleware." http://glite.web.cern.ch/glite/. Last accessed, October 2009.

[32] I. Bird, L. Robertson, and J. Shiers, "Deploying the LHC computing grid - the LCG service challenges," in *IEEE International Symposium on Mass Storage Systems and Technology*, (Washington, DC, USA), pp. 160–165, IEEE Computer Society, 2005.

[33] "LHC Computing Grid Project." http://lcg.web.cern.ch/LCG/. Last accessed, October 2009.

[34] A. Roy, "A Science Driven Production Cyberinfrastructurethe Open Science Grid," *Journal of Grid Computing*, 2010.

[35] "VDT: Virtual Data Toolkit." http://vdt.cs.wisc.edu/. Last accessed, October 2009.

[36] B. Segal, L. Robertson, F. Gagliardi, and F. Carminati, "Grid computing: the European Data Grid Project," in *Nuclear Science Symposium Conference Record, 2000 IEEE*, vol. 1, p. 2/1 vol.1, 2000.

[37] "European Data Grid Project." http://eu-datagrid.web.cern.ch/euLast accessed, October 2009.

[38] A. Skjellum, A. Kanevsky, Y. S. Dandass, J. Watts, S. Paavola, D. Cottel, G. Henley, L. S. Hebert, Z. Cui, and A. Rounbehler, "The Real-Time Message Passing Interface Standard: Research Articles," *Concurrency and Computation : Practice & Experience*, vol. 16, pp. 1–322, December 2004.

[39] " MPI: Message Passing Interface." http://www.mcs.anl.gov/research/projects/mpi/. Last accessed, October 2009.

[40] "EGEE MPI work group." http://egee-intranet.web.cern.ch/egee-intranet/NA1/TCG/wgs/EGEE-II-MPI-WG-TEC-2.pdf. Last accessed, October 2009.

[41] "EGEE Statistics." http://goc.grid.sinica.edu.tw/gstat/. Last accessed, October 2009.

[42] "EGI: European Grid Initiative." http://web.eu-egi.eu/. Last accessed, October 2009.

[43] "The Teragrid Project." http://www.teragrid.org/. Last accessed, October 2009.

[44] "OSG: Open Science Grid - A national, distributed computing grid for data-intensive reasearch." http://www.opensciencegrid.org/. Last accessed, October 2009.

[45] G. Klimeck, M. McLennan, S. P. Brophy, G. B. Adams III, and M. S. Lundstrom, "nanoHUB.org: Advancing Education and Research in Nanotechnology," *Computing in Science and Engineering*, vol. 10, pp. 17–23, September 2008.

[46] "nanoHUB - A resource for nanoscience and technology." http://nanohub.org/. Last accessed, October 2009.

[47] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, and S. Koranda, "GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists," in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, HPDC '02, (Washington, DC, USA), pp. 225–, IEEE Computer Society, 2002.

[48] "LIGO: Laser Interferometer Gravitational-Wave Observatory." https://www.lsc-group.phys.uwm.edu/lscdatagrid/OSG/index.html. Last accessed, October 2009.

[49] S. Benkner, C. Schröder, M. Lucka, and O. Steinhauser, "Grid Services for Parallel Molecular Dynamics with NAMD and CHARMM," in *Proceeding sof the international conference on Computational Science and Its Applications, Part I*, ICCSA '08, (Berlin, Heidelberg), pp. 1036–1051, Springer-Verlag, 2008.

[50] "CHARMM - Chemistry at HARvard Macromolecular Mechanics." http://www.charmm.org/. Last accessed, October 2009.

[51] L. Kleinrock, "A vision for the internet," in *ST Journal of Research*, 2005.

[52] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, (New York, NY, USA), pp. 164–177, ACM, 2003.

[53] "Xen - The Xen hypervisor." http://xen.org/. Last accessed, October 2009.

[54] "EC2 - Amazon Elastic Compute Cloud." http://aws.amazon.com/ec2/. Last accessed, October 2009.

[55] S. Hazelhurst, "Scientific computing using virtual high-performance computing: a case study using the Amazon elastic computing cloud," in *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, SAICSIT '08, (New York, NY, USA), pp. 94–103, ACM, 2008.

[56] "Blue Cloud - IBM's Cloud Computing Solution." http://www.ibm.com/grid/. Last accessed, October 2009.

[57] "IBM Power VM - A virtualisation platform for UNIX, Linux and IBM clients." http://www-03.ibm.com/systems/power/software/virtualization/. Last accessed, October 2009.

[58] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pp. 1 –10, May 2010.

[59] "The Apache Hadoop project - Develops open-source software for reliable, scalable, distributed computing." http://hadoop.apache.org/. Last accessed, October 2009.

[60] "Sun Cloud Computing." http://www.sun.com/solutions/cloudcomputing/index.jsp. Last accessed, October 2009.

[61] "GoGrid Cloud Hosting." http://www.gogrid.com/. Last accessed, October 2009.

[62] "ServePath - Managed Dedicated Server Hosting." http://www.servepath.com/. Last accessed, October 2009.

[63] J. Li, X. Ma, S. Yoginath, G. Kora, and N. F. Samatova, "Transparent runtime parallelization of the R scripting language," *Parallel Distributed Computing*, vol. 71, pp. 157–168, February 2011.

[64] "The R Project for Statistical Computing." http://www.r-project.org/. Last accessed, October 2009.

[65] "MATLAB." http://www.mathworks.com/products/matlab/. Last accessed, October 2009.

[66] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, *Workflows for e-Science: Scientific Workflows for Grids*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[67] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," 2008.

[68] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, pp. 219–237, July 2005.

[69] A. Harrison, I. Taylor, I. Wang, and M. Shields, "WS-RF Workflow in Triana," *International Journal of High Performance Computing Applications*, vol. 22, pp. 268–283, August 2008.

[70] Ludäscher, Bertram and Altintas, Ilkay and Berkley, Chad and Higgins, Dan and Jaeger, Efrat and Jones, Matthew and Lee, Edward A. and Tao, Jing and Zhao, Yang, "Scientific workflow management and the Kepler system: Research Articles," *Concurr. Comput. : Pract. Exper.*, vol. 18, pp. 1039–1065, August 2006.

[71] "The Kepler Project." https://kepler-project.org/. Last accessed, October 2009.

[72] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "VisTrails: visualization meets data management," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, (New York, NY, USA), pp. 745–747, ACM, 2006.

[73] G. von Laszewski and M. Hategan, "Workflow Concepts of the Java CoG Kit," *Journal of Grid Computing*, vol. 3, pp. 239–258, 2005. 10.1007/s10723-005-9013-5.

[74] K. M. Anderson, R. N. Taylor, and E. J. Whitehead, Jr., "Chimera: hypermedia for heterogeneous software development enviroments," *ACM Transactions on Information Systems*, vol. 18, pp. 211–245, July 2000.

[75] E. Deelman, J. Blythe, A. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Kor, "Mapping Abstract Complex Workflows onto Grid Environments," 2003.

[76] J. C. Jacob, D. S. Katz, G. B. Berriman, J. C. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T. A. Prince, and R. Williams, "Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking," *International Journal of Computuational Science and Engineering*, vol. 4, pp. 73–87, July 2009.

[77] S. T. Peltier, A. W. Lin, D. Lee, S. Mock, S. Lamont, T. Molina, M. Wong, L. Dai, M. E. Martone, and M. H. Ellisman, "The Telescience Portal for advanced tomography applications," *Journal of Parallel Distributed Computing*, vol. 63, pp. 539–550, May 2003.

[78] *Earthworks Science Gateway: Widening SCEC Community Access to the TeraGrid*, 2006.

[79] J. P. Morrison, *Condensed Graphs: Unifying Availability-Driven, Coercion-Driven and Control-Driven Computing*. PhD thesis, Technische Universiteit Eindhoven, 1996.

[80] J. Eker, J. Janneck, E. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, "Taming heterogeneity - the Ptolemy approach," *Proceedings of the IEEE*, vol. 91, pp. 127 – 144, Jan. 2003.

[81] "The Ptolemy Project - Heterogenous Modelling and Design." http://ptolemy.eecs.berkeley.edu/. Last accessed, October 2009.

[82] "Taverna Workbench." http://www.mygrid.org.uk/tools/taverna/. Last accessed, October 2009.

[83] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble, "Taverna, reloaded," in *Proceedings of the 22nd international conference on Scientific and statistical database management*, SSDBM'10, (Berlin, Heidelberg), pp. 471–481, Springer-Verlag, 2010.

[84] K. Wolstencroft, P. Alper, D. Hull, C. Wroe, P. W. Lord, R. D. Stevens, and C. A. Goble, "The myGrid ontology: bioinformatics service discovery," *Int. J. Bioinformatics Res. Appl.*, vol. 3, pp. 303–325, September 2007.

[85] "myGrid." http://www.mygrid.org.uk/. Last accessed, October 2009.

[86] R. T. Apteker, J. A. Fisher, V. S. Kisimov, and H. Neishlos, "Video acceptability and frame rate," *IEEE MultiMedia*, vol. 2, no. 3, pp. 32–40, 1995.

[87] R. Haber and D. McNabb, "Visualization idioms: A conceptual model for scientific visualization systems," *Visualization in Scientific Computing*, pp. 74–93, 1990.

[88] "Maverick - A supercomputer built to facilitate large-scale data analysis and remote terascale visualisation." http://www.tacc.utexas.edu/general/news/features/maverick.php. Last accessed, October 2009.

[89] J. D. Impson, "VNC, transparently," *Linux Journal*, vol. 2002, pp. 3–, January 2002.

[90] "VNC: Virtual Network Computing - A protocol to control another computer's screen remotely." http://www.hep.phy.cam.ac.uk/vnc_docs/index.html. Last accessed, October 2009.

[91] "RealVNC - A server and client application for the Virtual Network Computing (VNC) protocol." http://www.realvnc.com/. Last accessed, October 2009.

[92] "TightVNC - A server and client application for the Virtual Network Computing (VNC) protocol." http://www.tightvnc.com/. Last accessed, October 2009.

[93] "UltraVNC - A server and client application for the Virtual Network Computing (VNC) protocol." http://www.uvnc.com/. Last accessed, October 2009.

[94] J. P. Martin, R. J. Vickery, S. Ziegeler, and R. Angelini, "SSH-Enabled ParaView," in *Proceedings of the 2009 DoD High Performance Computing Modernization Program Users Group Conference*, HPCMP-UGC '09, (Washington, DC, USA), pp. 383–387, IEEE Computer Society, 2009.

[95] "ParaView - An open-source, multi-platform data analysis and visualization application." http://www.paraview.org/. Last accessed, October 2009.

[96] "Argonne National Laboratory - A U.S. Department of Energy laboratory." http://www.anl.gov/. Last accessed, October 2009.

[97] J. Cohen, N. Furmento, G. Kong, A. Mayer, S. Newhouse, and J. Darlington, "RealityGrid: An Integrated Approach to Middleware through ICENI," *Philisophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2005.

[98] "RealityGrid." http://www.realitygrid.org/. Last accessed, October 2009.

[99] J. C. Moore, "Visualizing with VTK," *Linux Journal*, vol. 1998, September 1998.

[100] "VTK: Visualisation Toolkit - An open-source software system for 3D computer graphics, image processing and visualisation." http://www.vtk.org/. Last accessed, October 2009.

[101] L. Childers, T. Disz, R. Olson, M. E. Papka, R. Stevens, and T. Udeshi, "Access Grid: Immersive Group-to-Group Collaborative Visualization," 2000.

[102] "Access Grid." http://www.accessgrid.org/. Last accessed, October 2009.

[103] G. Humphreys, M. Houston, Y. Ng, R. Frank, S. Ahern, P. Kirchner, and J. Klosowski, "Chromium: A stream processing framework for interactive graphics on clusters," 2002.

[104] *The RealityGrid PDA and Smartphone clients: Developing effective handheld user interfaces for e-Science*, 2006.

[105] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf, "The cactus framework and toolkit: design and applications," in *Proceedings of the 5th international conference on High performance computing for computational science*, VECPAR'02, (Berlin, Heidelberg), pp. 197–227, Springer-Verlag, 2003.

[106] "Cactus - An open source problem solving environment designed for scientists and engineers." http://www.cactuscode.org/. Last accessed, October 2009.

[107] "HDF5 - A data model, library, and file format for storing and managing data." http://www.hdfgroup.org/HDF5/. Last accessed, March 2009.

[108] "VFD: Virtual File Driver." http://jean-luc.aei.mpg.de/Projects/Gigabit/HDF5-DataGrid.html. Last accessed, October 2009.

[109] D. Stalling, M. Westerhoff, and H. christian Hege, "Amira: A highly interactive system for visual data analysis," in *The Visualization Handbook*, pp. 749–767, Elsevier, 2005.

[110] "Amira - A 3D visualisation suite." http://www.amiravis.com/. Last accessed, October 2009.

[111] "IBM's Open Visualisation Data Explorer." http://www.research.ibm.com/dx/. Last accessed, October 2009.

[112] "LCA Vision - A free scientific visualization tool." http://zeus.ncsa.uiuc.edu/ miksa/LCAVision.html. Last accessed, November 2004.

[113] "The GridLab Project." http://www.gridlab.org/. Last accessed, October 2009.

[114] E. Seidel, G. Allen, A. Merzky, and J. Nabrzyski, "GridLab: a grid application toolkit and testbed," *Future Generation Computer Systems*, vol. 18, pp. 1143–1153, October 2002.

[115] "GAT: Grid Application Toolkit." http://www.gridlab.org/WorkPackages/wp-1/. Last accessed, October 2009.

[116] "Triana - An open source problem solving environment." http://www.trianacode.org/. Last accessed, October 2009.

[117] R. Zheng, H. Jin, Q. Zhang, Y. Li, and J. Chen, "IPGE: Image Processing Grid Environment Using Components and Workflow Techniques," in *GCC*, pp. 671–678, 2004.

[118] H. Jin, "Chinagrid: Making grid computing a reality," in *of Lecture Notes in Computer Science*, pp. 13–24, Springer-Verlag, 2004.

[119] H. Rosmanith and D. Kranzlmüller, "glogin - a multifunctional, interactive tunnel into the grid," in *5th IEEE/ACM International Workshop on Grid Computing* (R. Buyya, ed.), IEEE Computer Society, (Pittsburgh, PA, USA), pp. 266 – 272, 2004.

[120] M. Polak and D. Kranzlmüller, "Interactive videostreaming visualization on grids," *Cluster and Computational Grids for Scientific Computing*, vol. 24, pp. 39–45, January 2008.

[121] D. K. Paul, P. Heinzlreiter, and J. Volkert, "Grid-enabled visualization with gvk," in *Proceedings of the First European Across Grids Conference, Santiago de*, 2003.

[122] M. Kupczyk, R. Lichwala, N. Meyer, B. Palak, M. Plociennik, M. Stroinski, and P. Wolniewicz, "The migrating desktop - the general entry point to the grid," in *6th CARNET Users Conference*, (Zagreb, Croatia), 2004.

[123] L. Hluchý, V. D. Tran, O. Habala, J. Astalos, B. Simo, and D. Froehlich, "Concept of a Problem Solving Environment for Flood Forecasting," in *Proceedings of*

the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, (London, UK), pp. 105–113, Springer-Verlag, 2002.

[124] "Flood Crisis Team Decision Support System Presentation." http://www.eu-crossgrid.org/floods.htm. Last accessed, October 2009.

[125] A. Tirado-Ramos and D. Groen, "On-Line Application Performance Monitoring of Blood Flow Simulation in Computational Grid Architectures," in *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, CBMS '05, (Washington, DC, USA), pp. 511–516, IEEE Computer Society, 2005.

[126] "Surgery Decision Support Application Presentation." http://www.eu-crossgrid.org/biomedical.htm. Last accessed, October 2009.

[127] H. Rosmanith, J. Volkert, R. Valles, F. Serrano, M. Plociennik, and M. Owsiak, "Interactive fusion simulation and visualisation on the grid," *Parallel and Distributed Computing, International Symposium on*, vol. 0, p. 20, 2007.

[128] "Fusion Simulations, data visualization and future requirements for the interactive grid infrastructure." http://dissemination.interactive-grid.eu/events/past-events/INGRID-2008/presentations/INGRID08c.ppt/download. Last accessed, October 2009.

[129] R. J. Blake, P. V. Coveney, P. Clarke, and S. Pickles, "The teragyroid experiment - supercomputing 2003," *Scientific Programming*, vol. 13, no. 1, pp. 1–17, 2005.

[130] "SGI Onyx." http://www.sgi.com/products/remarketed/onyx3000/tech_info.html. Last accessed, October 2009.

[131] J. Brooke, T. Eickermann, and U. Woessner, "Application Steering in a Collaborative Environment," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, SC '03, (Washington, DC, USA), pp. 61–, IEEE Computer Society, 2003.

[132] "Council for the Central Laboratory of the Research Councils (CCLRC) Report and Accounts." http://www.foi.clrc.ac.uk/Activities/CLRC0405/CCLRCAR0405.pdf, 2004-2005. Last accessed, October 2009.

[133] J. Toledo, "Etherape: A Graphical Network Monitor." http://etherape.sourceforge.net. Last accessed, October 2009.

[134] G. Ratterree, U. Pooch, and W. Marti, "Etherman Version 1.2: A Network Manager's Workbench Application," tech. rep., College Station, TX, USA, 2001.

[135] R. Ball, G. Fink, and C. North, "Home-centric visualization of network traffic for security administration.," in *ACM workshop on Visualization and data mining for computer security*, (New York, NY, USA), pp. 55–64, ACM Press, 2004.

[136] S. Lau, "The Spinning Cube of Potential Doom," *Communications of the ACM*, vol. 47, pp. 25–26, June 2004.

[137] V. Paxson, "Bro: a system for detecting network intruders in real-time," in *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*, (Berkeley, CA, USA), pp. 3–3, USENIX Association, 1998.

[138] "Bro Intrusion Detection System." http://bro-ids.org/. Last accessed, October 2009.

[139] A. Komlodi, P. Rheingans, U. Ayachit, J. Goodall, and A. Joshi, "A user-centered look at glyph-based security visualization," in *IEEE Workshops on Visualization for Computer Security*, (Washington, DC, USA), pp. 21–28, 2005.

[140] E. Le Malecot, M. Kohara, Y. Hori, and K. Sakurai, "Interactively combining 2d and 3d visualization for network traffic monitoring.," in *3rd international Workshop on Visualization For Computer Security*, 2006.

[141] "Real Time Monitor." http://gridportal.hep.ph.ic.ac.uk/rtm/. Last accessed, October 2009.

[142] "ASI: Active Security Infrastructure." https://wiki.fzk.de/i2g/index.php/I2G_Active_Security_Infrastructure. Last accessed, October 2009.

[143] K. Rochford, B. Coghlan, and J. Walsh, "An agent-based approach to grid service monitoring," in *International Symposium on Parallel and Distributed Computing*, 2006.

[144] "Nagios project documentation." http://www.nagios.org/documentation/. Last accessed, October 2009.

[145] "Site Functional Tests." http://goc.grid.sinica.edu.tw/gocwiki/Site_Functional_Tests. Last accessed, October 2009.

[146] "SAM: Service Availabilty Monitoring Tests." https://twiki.cern.ch/twiki/bin/view/LCG/SAMOverview. Last accessed, October 2009.

[147] "Apache ActiveMQ." http://activemq.apache.org/. Last accessed, October 2009.

[148] "Google Earth." http://earth.google.com/. Last accessed, October 2009.

[149] I. Milne and G. Rowe, "OGRE: Three-Dimensional Program Visualization for Novice Programmers," *Education and Information Technologies*, vol. 9, pp. 219–237, September 2004.

[150] "Ogre: Object-Oriented Graphics Rendering Engine." http://www.ogre3d.org/. Last accessed, October 2009.

[151] "Open LDAP." http://www.openldap.org/. Last accessed, October 2009.

[152] "Berkeley Database Information Index." https://twiki.cern.ch/twiki/bin/view/EGEE/BDII. Last accessed, October 2009.

[153] S. Maad, G. Pierantoni, B. Coghlan, and E. Kenny, "Universal accessibility to the grid: A social agent model for the road ahead.," *IADIS International Journal of WWW/Internet*, 2007.

[154] R. Raman, M. Livny, and M. Solomon, "Matchmaking distributed resource management for high throughput computing.," in *Seventh IEEE International Symposium on High Performance Distributed Computing*, (Chicago, IL, USA), 1998.

[155] M. Ezekiel and K. A. Fox, *Methods of correlation and regression analysis, linear and curvilinear.* Wiley New York, 3d ed., 1959.

[156] D. L. Isaacson and R. W. Madsen, *Markov chains, theory and applications.* Wiley, New York, 1976.

[157] C. E. Shannon, "Prediction and entropy of printed English," *The Bell System Technical Journal*, vol. 30, pp. 50–64, 1951.

[158] G. D. Fornay, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[159] X. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for Speech Recognition.* New York, NY, USA: Columbia University Press, 1990.

[160] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," tech. rep., Stanford Digital Library Technologies Project, 1998.

[161] B. A. Berg, *Markov Chain Monte Carlo Simulations And Their Statistical Analysis: With Web-based Fortran Code*. World Scientific Publishing Company, October 2004.

[162] R. L. Winkler, *An introduction to Bayesian inference and decision*. Holt, Rinehart and Winston, 1972.

[163] G. Tsouloupas and M. D. Dikaiakos, "Characterization of computational grid resources using low-level benchmarks," in *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, (Washington, DC, USA), p. 70, IEEE Computer Society, 2006.

[164] F. Georgatos, J. Kouvakis, and J. Kouretis, "Performability aspects of the atlas vo; using lmbench suite," 2008.

[165] M. D. Dikaiakos, "Grid benchmarking: vision, challenges and current status," *Concurrency and Computation: Practice and Experience*, vol. 19, no. 1, pp. 89–105, 2007.

[166] G. Tsouloupas and M. D. Dikaiakos, "Gridbench: A tool for the interactive performance exploration of grid infrastructures," *J. Parallel Distrib. Comput.*, vol. 67, no. 9, pp. 1029–1045, 2007.

[167] H. J. Curnow and B. A. Wichmann, "A synthetic benchmark," *The Computer Journal*, vol. 19, no. 1, pp. 43–49, 1976.

[168] R. P. Weicker, "Dhrystone: a synthetic systems programming benchmark.," *Commmunications of the ACM*, vol. 27, no. 10, pp. 1013–1030, 1984.

[169] A. Aburto, "Flops Version 2.0." ftp.sunet.se/pub/benchmark/aburto/flops/flops.c. Last accessed, September 2009.

[170] M. Frigo and S. Johnson, "The Design and Implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216 –231, 2005.

[171] "FFTW: Fastest Fourier Transform in the West." http://www.fftw.org/. Last accessed, October 2009.

[172] Buss, Samuel R., *3D Computer Graphics: A Mathematical Introduction with OpenGL*. New York, NY, USA: Cambridge University Press, 2003.

[173] "RayTrace software package." http://math.ucsd.edu/ sbuss/MathCG/RayTrace/. Last accessed, October 2009.

[174] "The VirtualGL Project." http://www.virtualgl.org/. Last accessed, October 2009.

[175] Timo Baur and Rebecca Breu and Tibor Kálmán and Tobias Lindinger and Anne Milbert and Gevorg Poghosyan and Mathilde Romberg and Forschungszentrum Karlsruhe Gmbh and Steinbuch Centre For Computing, "Adopting GLUE 2.0 for an Interoperable Grid Monitoring System."

[176] "The GLUE project homepage." http://forge.ogf.org/sf/projects/glue-wg. Last accessed, October 2009.

[177] M. Jouvin, "Subject: RE: Change of name for HEPSPEC06, forwarded 30 Oct 2009 08:42:06 by Maite Barroso Lopez to EGEE-III ROC Manager email list <project-egee-roc-managers@cern.ch>." Email, October 2009.

[178] G. Pierantoni, *Social Grid Agents*. PhD thesis, Trinity College Dublin, 2008.

[179] G. U. Yule, *An introduction to the theory of statistics*. London, 11th ed., 1937.

[180] F. J. Anscombe, "Graphs in statistical analysis," *The American Statistician*, vol. 27, no. 1, pp. 17–21, 1973.

[181] R. A. Fisher, *Statistical methods for research workers*. Oliver & Boyd, Edinburgh :, 11th ed., 1950.

[182] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York: Dover, ninth dover printing, tenth gpo printing ed., 1964.

[183] J. E. Freund, *Modern elementary statistics*. Prentice-Hall, Englewood Cliffs, N.J. :, 3d ed. ed., 1967.

[184] E. S. Pearson and H. O. Hartley, *Biometrika tables for statisticians. v.I / edited by E.S. Pearson and H.O. Hartley*. C.U.P. for the Biometrika Trustees, Cambridge, 1956.

Appendix A

A.1 Previously Published Paper on Visualisation Framework

Reproduced overleaf courtesy of Prof. Erick Elmroth, Umea University, Sweden.

Integrating a Common Visualization Service into a Metagrid.

R. Watson, S. Maad, and B. Coghlan

Trinity College Dublin, Dublin, Ireland,
watsonr@cs.tcd.ie,
WWW home page: http://www.cs.tcd.ie/∼watsonr

Abstract. Existing architectures and tools for visualization on the Grid (Virtual Reality:VR, Augmented Reality:AR) include: CrossGrid [1], glogin [2], GVK [3], Migrating Desktop [4] and Gvid [5]. Whereas these architectures were useful for some applications they cannot be considered as complete. Our paper proposes a complete architecture that provides a more generic solution for applications involving visualization and simulation. The paper presents the architecture and scenarios for interoperability with existing grids.

Key Words: metagrid, visualization, grid

1 Introduction

There are numerous visualization applications that may benefit from the Grid's computational power but are not doing so because of missing supporting grid infrastructure. Here a basic architecture is proposed to serve these applications and enhance their performance. A generic architecture that can be used by all visualization applications is required. The visualization pipeline can be conceived in several ways [6]. In this paper the visualization pipeline is considered as a sequence of four tasks, computation, interaction, rendering and display. The Grid can handle the computation and delegate the data produced from the Grid to a dedicated visualization engine which can then stream the images to a display. For it to be a common facility for arbitrary Grids, the visualization engine should be a part of a metagrid infrastructure.

This paper is divided into six sections including this one. The second section overviews existing grid visualization solutions and their limitations. The third section states the objectives of our architecture while the fourth and fifth sections describe the visualization architecture and it's interoperability with the Grid. The paper concludes with a summary and some comments about further improvements.

2 Literature Overview

In this section a quick overview of the technologies, architectures and tools used so far in grid visualization will be discussed and analysed. The limitations of

those solutions are highlighted and the degree of success of these solutions in various applications is examined.

2.1 Technologies

This work derives from involvement in the EU CrossGrid project [1] which was oriented towards compute and data-intensive applications that involve the interaction of a user in the processing loop. Such applications require a response from the Grid to an action by a human agent in different time scales. Tools developed within the CrossGrid project are glogin, Gvid, GVK, and the Migrating Desktop. These are briefly described here.

glogin: This tool provides a tunnel into the Grid and therefore facilitates interaction with the Grid's resources.

Gvid: Allows rendering to be done on Grid resources, with transmission of resulting video over the Grid.

GVK: With GVK the user is able to control the execution of a grid application by installing a bi-directional interactive link between the scientific application and the visualization tool. The link itself is established using the glogin tool

Migrating Desktop: The Migrating Desktop is a framework for a graphical user interface for application management, grid and job monitoring, data and metadata management. This graphical environment is used as an advanced client for accessing grid resources in CrossGrid.

2.2 Applications

The above CrossGrid visualization tools have been applied in various domains including health and environment. These are briefly overviewed below.

Blood Flow Simulation: A Grid-based prototype system for pretreatment planning in vascular interventional and surgical procedures through real-time interactive simulation of vascular structure and flow. The system consists of a distributed real-time simulation environment, with which a user interacts in Virtual Reality (VR). A 3D model of a patient's arteries, derived using medical imaging techniques, serves as input to the environment for blood flow calculations.

Flood Crisis System: Grid-enabled simulations of three physical systems pertinent to flood crisis management: meteorology, hydrology and hydraulics. The main component of the system is a highly automated early warning system, based on hydro-meteorological (snowmelt) rainfall-runoff simulations.

2.3 Limitations

In each of the examples above there is a problem in that the application developer's task is to write application-specific plugins that can communicate with the appropriate web services. Is there away around this problem? For every application a plugin is required. This plugin can be very specific to the type of

simulation being run within the visualization application. For widely used software, like VTK, it may be possible to have a range of plugins available, but it would be better if no extra software was needed on the application side.

Other interactive visualization applications have used tools such glogin to tunnel into the Grid and run the applications on the Grid through this pseudo terminal. This indeed is running an interactive application on the Grid but there are two problems with this. Firstly, when using common middleware such as Globus [7], LCG2 [8] or EGEE [9] the application has to be installed on the "gatekeeper" that the user uses glogin to connect to. Secondly this application is running on the gatekeeper and not a general compute node.

3 Objectives

In this section the objectives of the architecture will be described. Fundamental issues addressed are:

1. *Providing a complete architecture.*
2. *Covering most of the visualization applications.*
3. *Imposing minimum re-engineering costs.*
4. *Interoperating with all Grids.*
5. *Use of the Grid for the computation.*

What is needed is a complete architecture that will deal with visualization applications that are compute intensive, allowing for interaction and performance increases. Not restricting the architecture to a particular Grid infrastructure is also an important objective, and so we target a metagrid rather than a specific Grid. We intend that the visualization engine should be part of a metagrid infrastructure. This visualization engine is described in the next section.

4 Visualization Architecture

Visualization is generically broken down into four parts,

- Computation
- Interaction
- Rendering
- Display

While the Grid succeeded so far in offering computational power, it hasn't provided a universal solution for interaction, rendering or display. So in our architecture the main focus is on these three aspects. Existing solutions have tackled one or more of these issues to some extent, but not all three together. For interaction, glogin is a very adequate solution that gives the user a direct login connection into compute nodes. Rendering, on the other hand, traditionally depends on the local workstation at which the user is at. In some cases the visualization application is run on the gatekeeper and glogin is used to view

its output. This is not ideal as the graphical capabilities of the gatekeeper are unknown, and more importantly that is not the job of the gatekeeper.

In our architecture we suggest that a dedicated visualization engine be put in place to deal with jobs that require more power in rendering their output. The head node of the visualization engine is added to a gatekeeper queue. By way of example we have constructed a Visualization Engine comprising of the following components:

- 9-node VRengine
- WorkerNode Software
- Scalable Coherent Interconnect (SCI)
- Chromium
- AccessGrid

Chromium [10] is a system for manipulating streams of graphics API commands on clusters of workstations. Chromium's stream filters can be arranged to create sort-first and sort-last parallel graphics architectures that, in many cases, support the same applications while using only commodity graphics accelerators. In addition, these stream filters can be extended programmatically, allowing the user to customize the stream transformations performed by nodes in a cluster.

We have incorporated a high-speed interconnect, SCI [11], configured in a 3-d torus. At present the only protocol that is supported by Chromium is TCP/IP, so the Dolphin SuperSockets [12] is used to provide a fast and transparent way for TCP/UDP/IP to use SCI as the transport medium. The major benefits are a high bandwidth and much lower socket latency than network technologies like Gigabit Ethernet, Infiniband and Myrinet.

Finally, for the display it is important that streams of rendered images be conveyed from the visualization engine to the (remote) user via widely used protocols that have good supporting display software. We have chosen to adopt the AccessGrid [13] streaming protocols for this purpose. This has the distinct advantage that the visualization output stream can be incorporated into AccessGrid video conferences.

5 Interoperability with Grid

Various job submission will be explored in this section that involve visualization.

Scenario 1: In this scenario the user submits a job to a resource broker. The simulation is then sent to one or more compute nodes by the resource broker.

Scenario 2: Involves the creation of a wrapper around the resource broker which would split the job into two parts. The first computation part would be sent to the compute nodes and the second part would be sent to the visualization engine.

Scenario 3: Involves a modification of the job description with added information about the visualization pipeline.

6 Comments

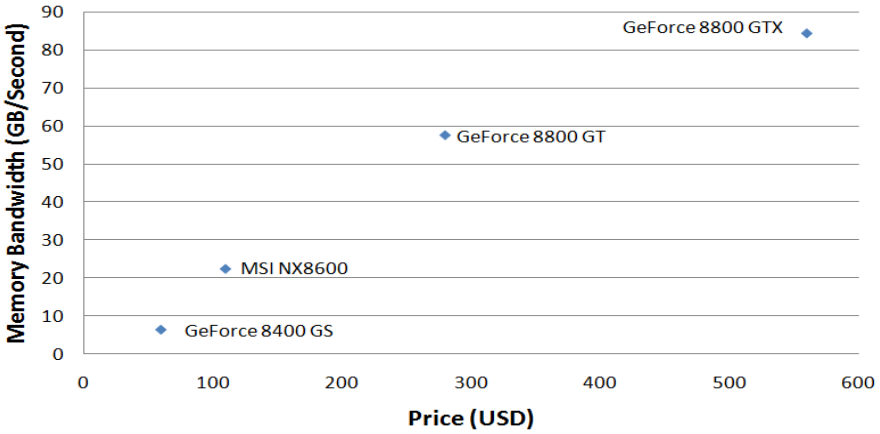
In this paper we have presented the integration of a visualization service into a metagrid and have presented three scenarios of job submission where this visualization engine could be used.

References

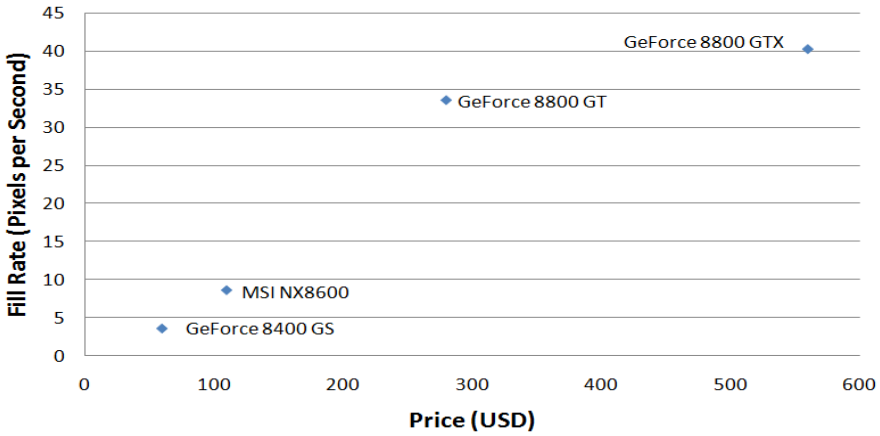
1. The EU CrossGrid Project, http://www.eu-crossgrid.org
2. H. Rosmanith, D. Kranzlmüller, *glogin - A Multifunctional, Interactive Tunnel into the Grid*, In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, PA, USA, November 2004.
3. D. Kranzlmüller, P. Heinzlreiter, H. Rosmanith, J. Volkert, *Grid-Enabled Visualization with GVK*, Lecture Notes in Computer Science. Vol. 2970, 2004, pp.139–146
4. M. Kupczyk, R. Lichwala, N. Meyer, B. Palak, M. Plociennik, and P. Wolniewicz, *Roaming Access and Migrating Desktop*, Proceedings of The 2nd Cracow Grid Workshop, Cracow, Poland, December 2002, pp. 148-154
5. P. Heinzlreiter, *Interactive Result Visualization on the Grid*, Presentation at Grid Computing for ComplexProblems Bratislava, Slovakia, 29. November 2005.
6. R.B. Haber and D.A. McNabb, *Visualization Idioms: A Conceptual Model for Scientific Visualization Systems*, in G.M. Nielson, B. Shriver, and L.J. Rosenblum, (Eds.): Visualization in Scientific Computing, IEEE Computer Society, Los Alamitos, NM, USA, pp. 7493 (1990).
7. The Globus Project, http://www.globus.org/
8. LHC Computing Grid Project, http://lcg.web.cern.ch/LCG/
9. Enabling Grids for E-science in Europe, http://www.eu-egee.org/
10. G. Humphreys, M. Houston, Y. Ng, R. Frank, S. Ahern, P. Kirchner and J.T. Klosowski, *Chromium: A Stream Processing Framework for Interactive Graphics on Clusters*, ACM SIGGRAPH, July 2002.
11. Scalable Coherent Interface (SCI), ANSI/IEEE Standard, 1596-1992, August 1993.
12. F. Seifert and H. Kohmann, *SCI SOCKETS - A Fast Socket Implementation over SCI*, http://www.dolphinics.com/pdf/whitepapers/sci-socket.pdf
13. AccessGrid, http://www.accessgrid.org/

Appendix B

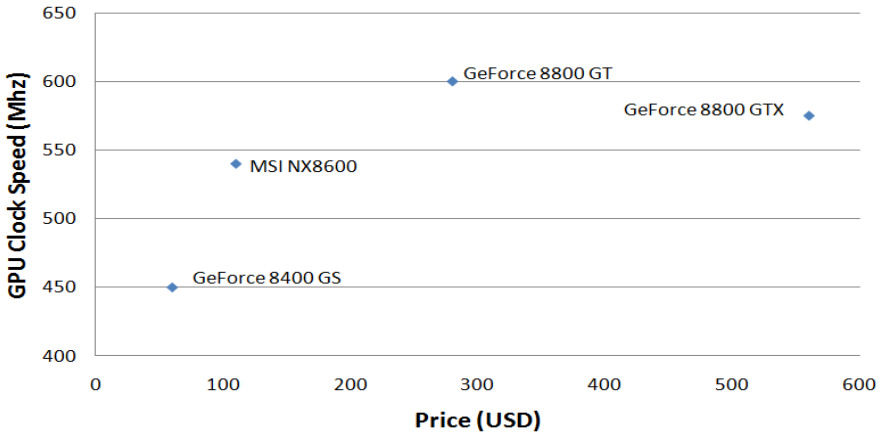
B.1 Metrics of selected graphics cards



(a) Memory Bandwidth of selected GPUs against their cost.



(b) Peak Fillrate of selected GPUs against their cost.



(c) Clock Speed of selected GPUs against their cost.

Figure B.1: Plots of selected graphics cards.

Appendix C

C.1 Example .vge XML File

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<vgworld>
        <sites>
                <site siteName = "gridgate.cs.tcd.ie">
                        <ce cpu_load = "25"
                            uptime = "100ms"
                            disk_size="150gb"
                            disk_util="80gb"
                            mem_size="1024mb"
                            mem_util="500mb"
                            network_stats="Up"
                            process_list="ls"
                        >
                        </ce>
                        <ce cpu_load = "25"
                            uptime = "100ms"
                            disk_size="150gb"
                            disk_util="80gb"
                            mem_size="1024mb"
                            mem_util="500mb"
                            network_stats="Up"
                            process_list="ls"
                        >
                        </ce>

                </site>
                <site siteName = "gridgate.cs.ucc.ie">
                        <ce cpu_load= "50"
                             uptime = "125ms"
                            disk_size="100gb"
                            disk_util="80gb"
```

```
                        mem_size="1024mb"
                        mem_util="726mb"
                        network_stats="Up"
                        process_list="ls"
                    >
                    </ce>
                </site>
        </sites>
</vgworld>
```

Appendix D

The following is a list of extensions that Ogre can, and will use, if they are available.

D.1 GL Core Extensions

GL_ARB_fragment_program
GL_ARB_fragment_shader
GL_ARB_multisample
GL_ARB_multitexture
GL_ARB_occlusion_query
GL_ARB_shader_objects
GL_ARB_shading_language_100
GL_ARB_texture_compression
GL_ARB_texture_cube_map
GL_ARB_texture_env_combine
GL_ARB_texture_env_dot3
GL_ARB_texture_float
GL_ARB_texture_non_power_of_two
GL_ARB_vertex_buffer_object
GL_ARB_vertex_program
GL_ARB_vertex_shader
GL_ATI_fragment_shader
GL_ATI_texture_float
GL_EXT_secondary_color
GL_EXT_stencil_two_side
GL_EXT_stencil_wrap
GL_EXT_texture_compression_s3tc
GL_EXT_texture_cube_map
GL_EXT_texture_env_combine

GL_EXT_texture_env_dot3
GL_EXT_texture_filter_anisotropic
GL_NV_occlusion_query
GL_NV_register_combiners
GL_NV_register_combiners2
GL_NV_texture_compression_vtc
GL_NV_texture_shader
GL_NV_vertex_program
GL_SGIS_generate_mipmap

D.2 Windows GL extensions

WGL_ARB_extensions_string
WGL_ARB_multisample
WGL_ARB_pbuffer
WGL_ARB_pixel_format
WGL_ARB_pixel_format_float
WGL_ARB_render_texture
WGL_ATI_pixel_format_float
WGL_EXT_swap_control
WGL_EXT_extensions_string

Appendix E

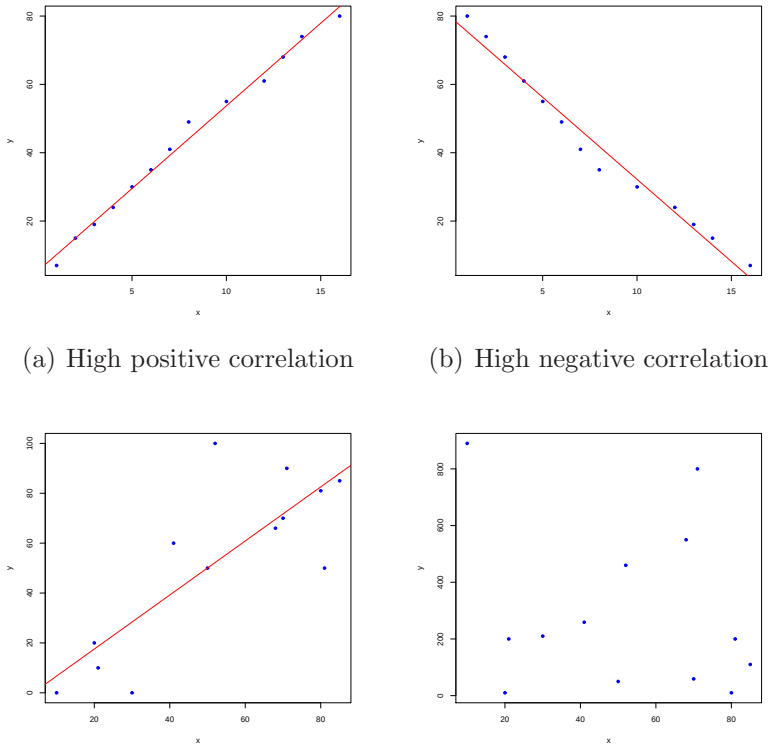
E.1 Correlation

When measurements are made of two or more independent variables, correlation [179] can be used to check if the variables are inter-related, and, if so, to find the measure of the degree of association. Figure E.1 show different possible correlations between variables x and y. The correlation is said to be *positive* if 'large' values of both variables tend to occur together, and is said to be *negative* if 'large' values of one variable tend to occur with 'small' values of the other variable. Correlations are said to be *high* if the observations lie close to a straight line and is said to be *low* if the observations are widely scattered. Variables are said to be uncorrelated if there does not appear to be any relationship between them.

The most important measure of the degree of correlation between two variables is called the *correlation coefficient*. Given n pairs of measurements, (x_i, y_i) of two random variables X and Y, the correlation coefficient is given by:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{[\sum(x_i - \bar{x})^2][\sum(y_i - \bar{y})^2]}}.$$

It can be shown that the value of r must lie between -1 and $+1$. For $r = +1$, all observed points lie on a straight line which has a positive slope; for $r = -1$, all observed points lie on a straight line with a negative slope. The correlation coefficient should only be calculated when the relationship between the two random variables is thought to be linear. If a scatter plot indicates a non-linear relationship then the correlation coefficient will be misleading and should not be calculated. A good example of the problems of only looking at the correlation coefficient and not at the scatter plots is shown in Figure E.2. It shows scatter plots of four data sets which look very different as plots, but these plots all have the same simple statistical properties. They were constructed in 1973 by the statistician F.J. Anscombe to demonstrate both the importance of graphing data before analysing it and the effect of outliers on statistical



(a) High positive correlation (b) High negative correlation

(c) Low positive correlation (d) No correlation

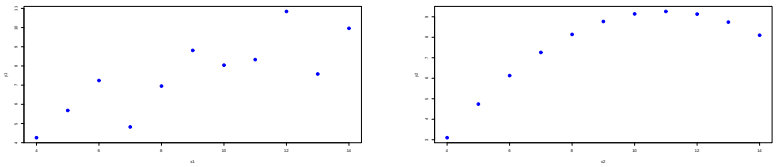
Figure E.1: Types of correlation.

properties [180].

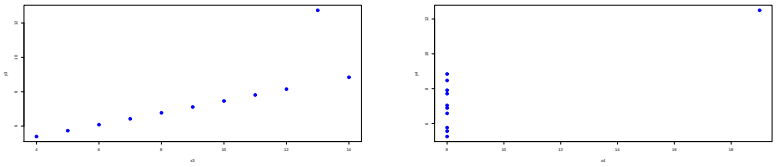
E.2 Significance Test

Performing a significance test [181] is useful to see if the observed correlation coefficient is significantly different from zero. If there is no correlation between the two variables, it is still possible that some outlier value may skew the correlation coefficient as show by Anscombe's quartet. When the true correlation coefficient is zero, it can be shown that the statistic $\frac{r\sqrt{(n-2)}}{\sqrt{(1-r^2)}}$ has a $t-$distribution [182] with $n-2$ degrees of freedom. If a positive or negative correlation is of interest then a two-tailed test [183]is appropriate. The correlation is significantly different from zero at the α level of significance if (see Table E.1):

$$\left|\frac{r\sqrt{(n-2)}}{\sqrt{(1-r^2)}}\right| \geq t_{\alpha/2,n-2}.$$



(a) (b)



(c) (d)

Figure E.2: Anscombe's quartet: All four sets are identical when examined statistically, but vary considerably when graphed. Data set from *Graphs in Statistical Analysis* [180].

Sample Size	Critical Value
5	0.88
10	0.63
15	0.51
20	0.44
25	0.39
30	0.36
50	0.28
100	0.20

Table E.1: 95% confidence intervals for the true correlation coefficient given in Pearson and Hartley [184].

E.3 Regression Line Estimation

If the correlation coefficient indicates that the random variables are independent, and the scatter plots are also indicating a possible correlation, then a regression line can be drawn, which makes predicting the value of one of the variables given the value of the other variable possible. It is important to realise that there are two regression lines, one to predict y from x and one to predict x from y. If the variables are linearly related, then the regression line of y on x can be shown as:

$$y = a_0 + a_1 x.$$

A straight line can be represented by the equation $y = a_0 + a_1 x$ and the least squares method estimates a_0 and a_1 such that the line gives a good fit to the data. At

any point x_i the corresponding point on the line is given by $a_0 + a_1 x_i$, so the difference between the observed value of y and the predicted value is given by $e_i = y_i - (a_0 + a_1 x_i)$. The least squares estimates of a_0 and a_1 are obtained by choosing the values which minimise the sum of squares of these deviations. The sum of the squared deviations is given by

$$S = \sum_{i=1}^{n} e_i^2.$$

It can be minimised by calculating $\frac{\partial S}{\partial a_0}$ and $\frac{\partial S}{\partial a_1}$, setting both these partial derivatives equal to zero, and solving the two simultaneous equations to obtain the least squares estimates, $\hat{a_0}$ and $\hat{a_1}$, of a_0 or a_1. These two simultaneous equations in $\hat{a_0}$ and $\hat{a_1}$ are called the normal equations. They can be solved to give

$$\hat{a_0} = \bar{y} - \hat{a_1}\bar{x},$$

$$\hat{a_1} = \frac{\sum x_i(y_i - \bar{y})}{\sum x_i(x_i - \bar{x})} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

After the least squares regression line has been calculated, as in Figure E.3, it is possible to use this model to predict values of the dependant variable. At a particular value, x_i, of the controlled variable, the point estimate of y_i is given by $\hat{a_0} + \hat{a_1} x_i$

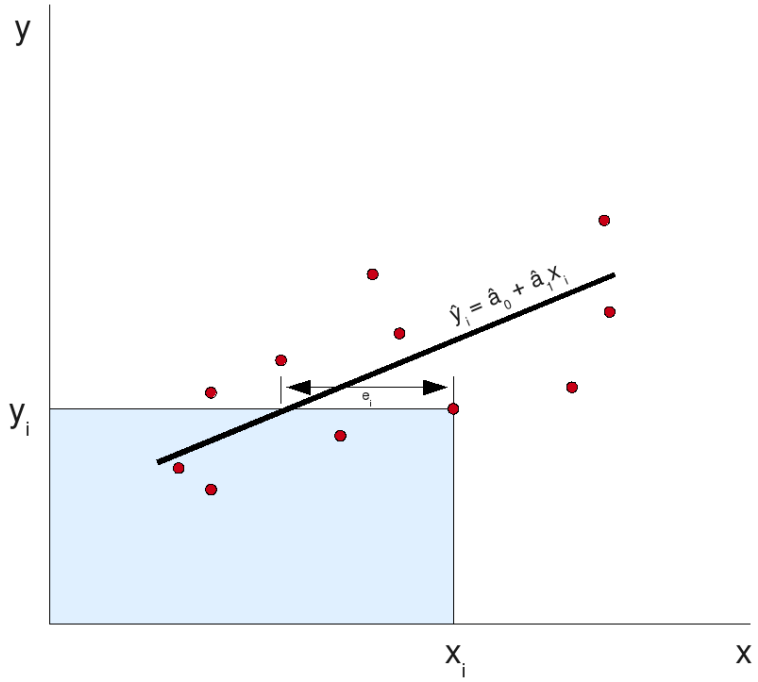


Figure E.3: The regression line of y on x.

E.4 Confidence intervals in linear regression

It is useful to know the confidence intervals for a_0, a_1 and $a_0 + a_1x$, given n pairs of observations $(x_1, y_1), ..., (x_n, y_n)$. It can be shown that the variances of $\hat{a_0}$ and $\hat{a_1}$ are

$$\hat{a_0} = \frac{\sigma^2_{y|x}}{n}\left[1 + \frac{n\bar{x}^2}{\sum(x_i - \bar{x})^2}\right]$$

and

$$\hat{a_1} = \frac{\sigma^s_{y|x}}{\sum(x_i - \bar{x})^2},$$

and that both $\hat{a_0}$ and $\hat{a_1}$ are normally distributed. In order to obtain confidence intervals for a_0, a_1 and $a_0 + a_1x$ an estimate of the residual variance, $\sigma^2_{y|x}$, must be calculated. The sum of the squared deviations of the observed points from the estimated regression line is given by $\sum(y_i - \hat{a}_0 - \hat{a}_1x_i)^2$. It can be shown that an unbiased estimate of $\sigma^2_{y|x}$ can be obtained by dividing the sum of squares by $n - 2$, giving

$$s^2_{y|x} = \frac{\sum(y_i - \hat{a}_0 - \hat{a}_1x_i)^2}{n - 2}.$$

The denominator, $n - 2$, shows that two degrees of freedom have been lost. This is because the two quantities $\hat{a}_0$ and $\hat{a}_1$ were estimated from the data, so there are two linear restrictions on the values of $y_i - \hat{a}_0 - \hat{a}_1x_i$. It can then be shown that the $100(1 - \alpha)$ per cent confidence interval for a_1 is given by

$$\hat{a}_1 \pm t_{\frac{1}{2}\alpha,n-2} \times \frac{s_{y|x}}{\sqrt{[\sum(x_i - \bar{x})^2]}},$$

for a_0 by

$$\hat{a}_0 \pm t_{\frac{1}{2}\alpha,n-2} \times s_{y|x}\sqrt{\left[\frac{1}{n} + \frac{\bar{x}^2}{\sum(x_i - \bar{x})^2}\right]}$$

and for $a_0 + a_1x_0$ by

$$\hat{a}_0 + \hat{a}_1x_0 \pm t_{\frac{1}{2}\alpha,n-2} \times s - y|x\sqrt{\left[\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum(x_i - \bar{x})^2}\right]}.$$

The confidence interval for $a_0 + a_1x$ is shown in Figure E.4, and from the figure it can be observed that the shortest interval is when $x = \bar{x}$.

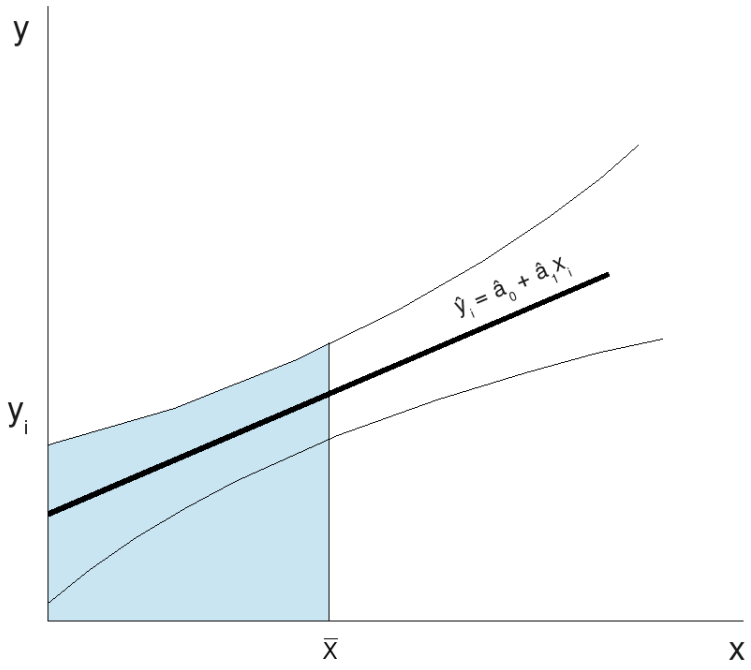


Figure E.4: Confidence intervals for $a_0 + a_1 x$.

Appendix F

F.1 Benchmarking Results

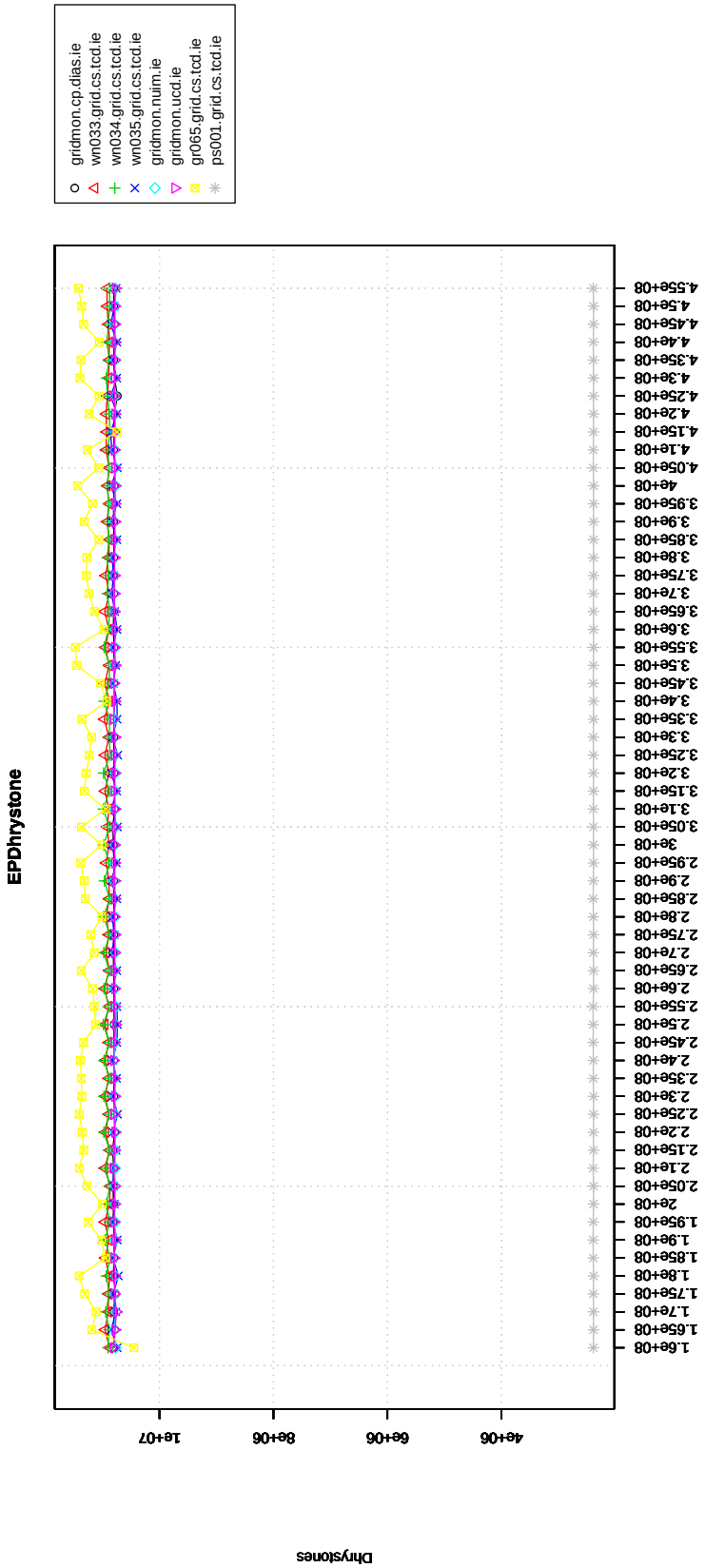


Figure F.1: EPDhrystone micro-benchmark results from Grid-Ireland.

Figure F.2: EPWhetstone micro-benchmark results from Grid-Ireland.

Figure F.3: EPFlops micro-benchmark results from Grid-Ireland.

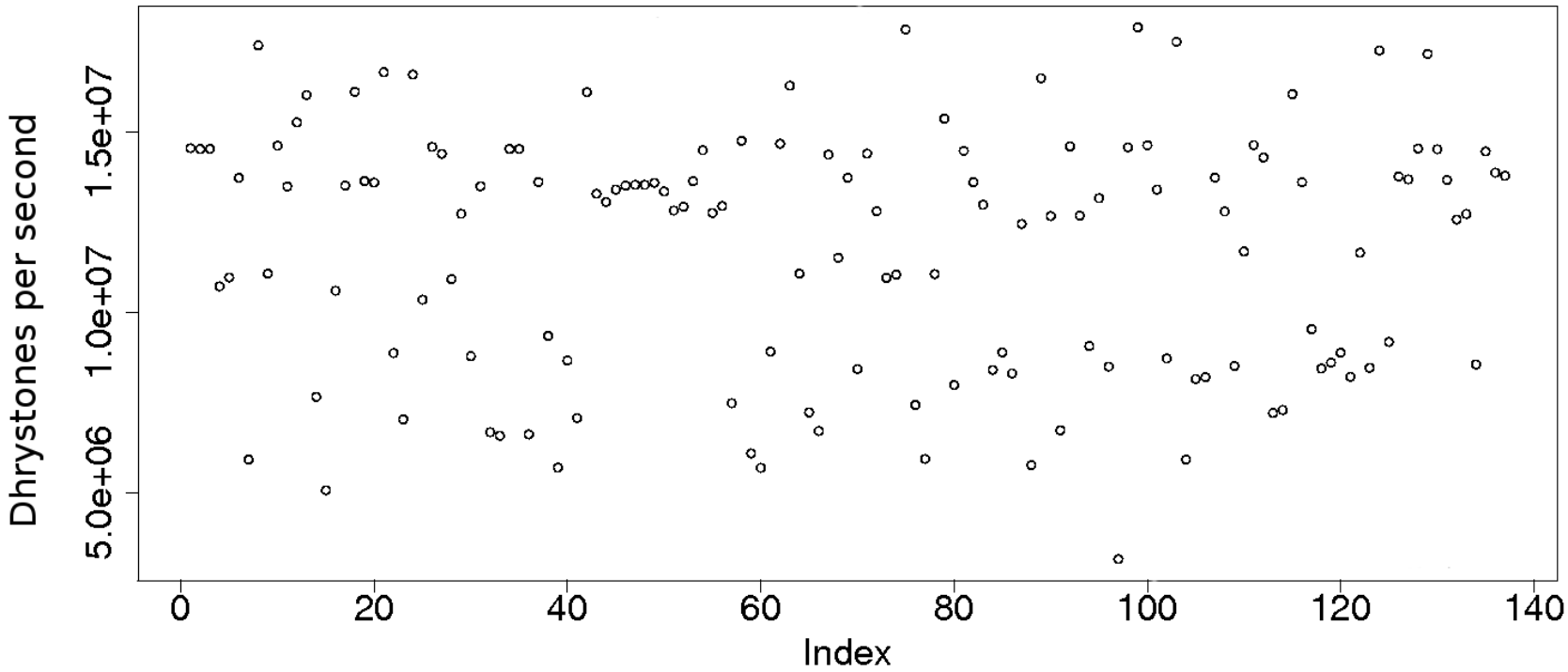


Figure F.4: EPDhrystone micro-benchmark results from EGEE.

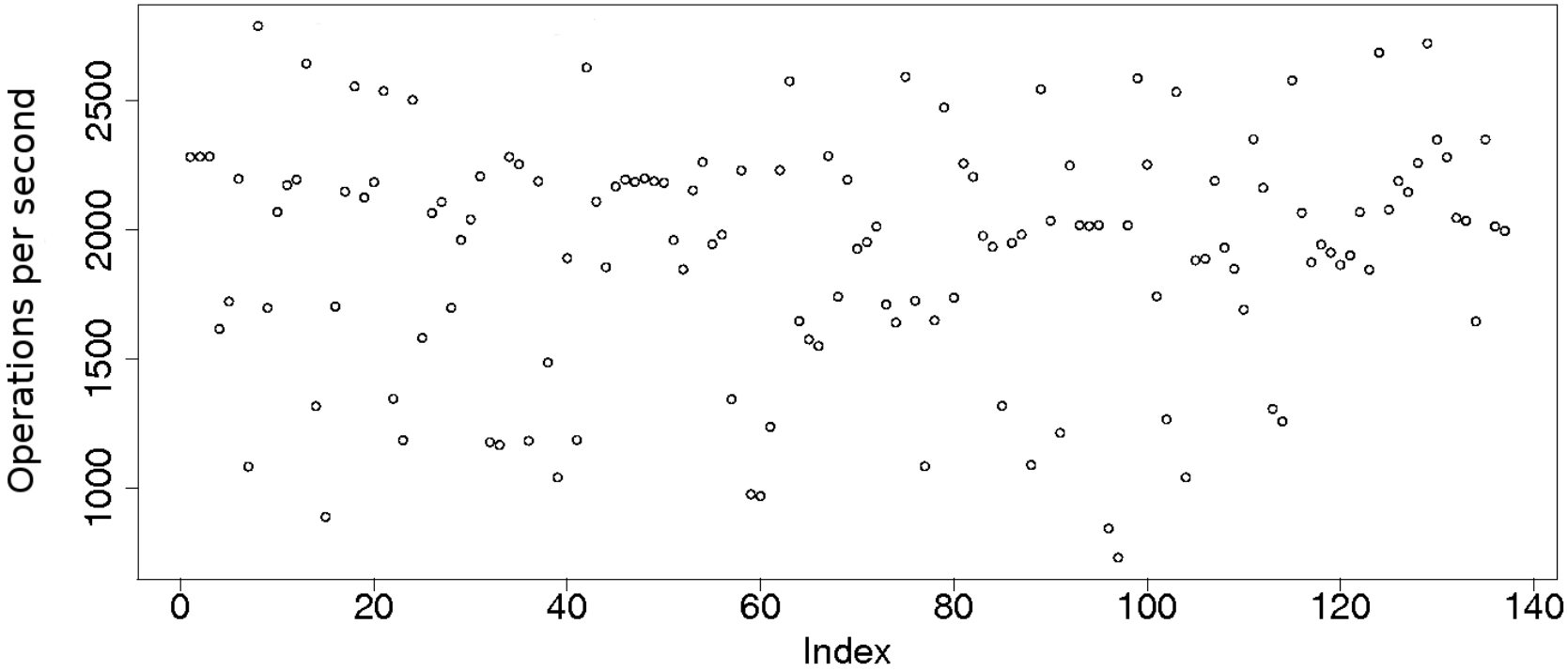


Figure F.5: EPWhetstone micro-benchmark results from EGEE.

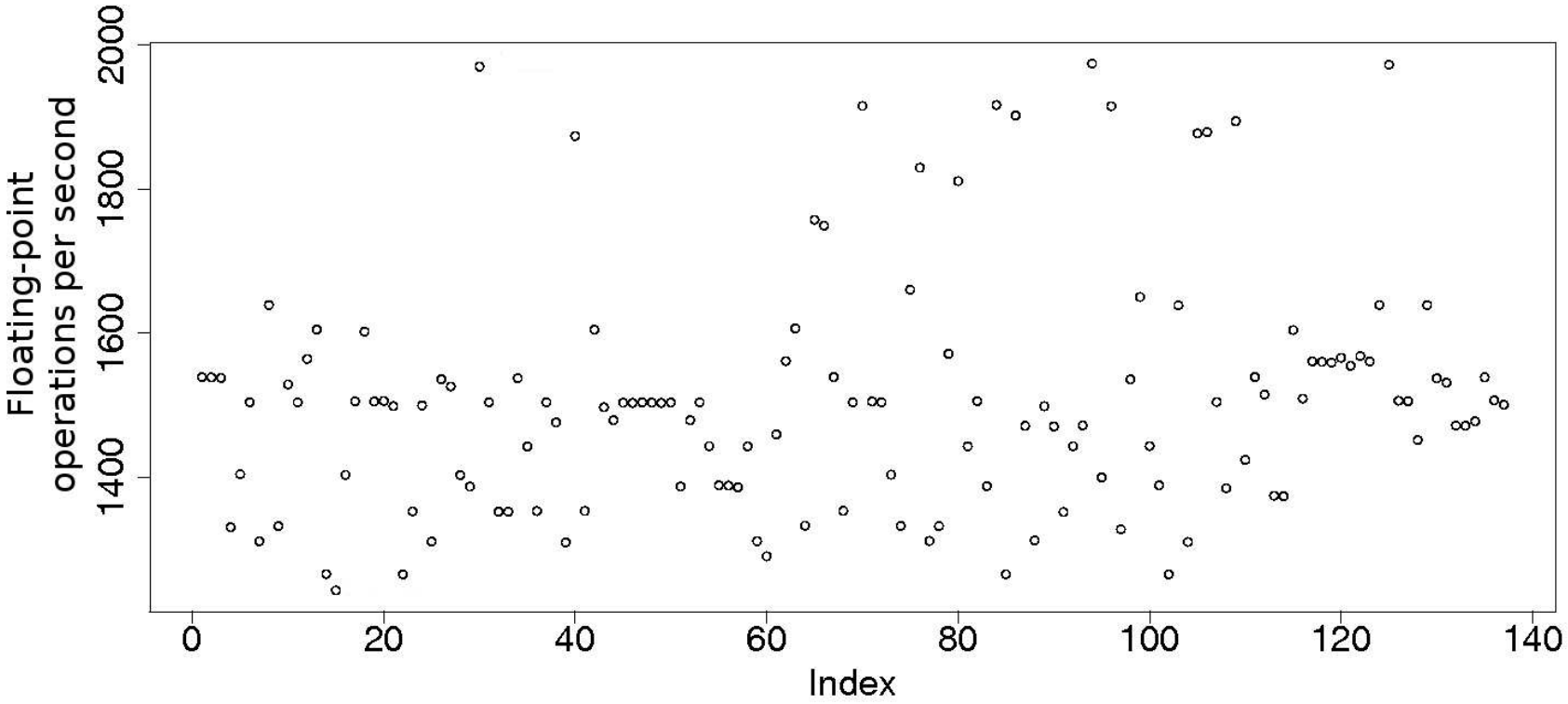


Figure F.6: EPFlops micro-benchmark results from EGEE.

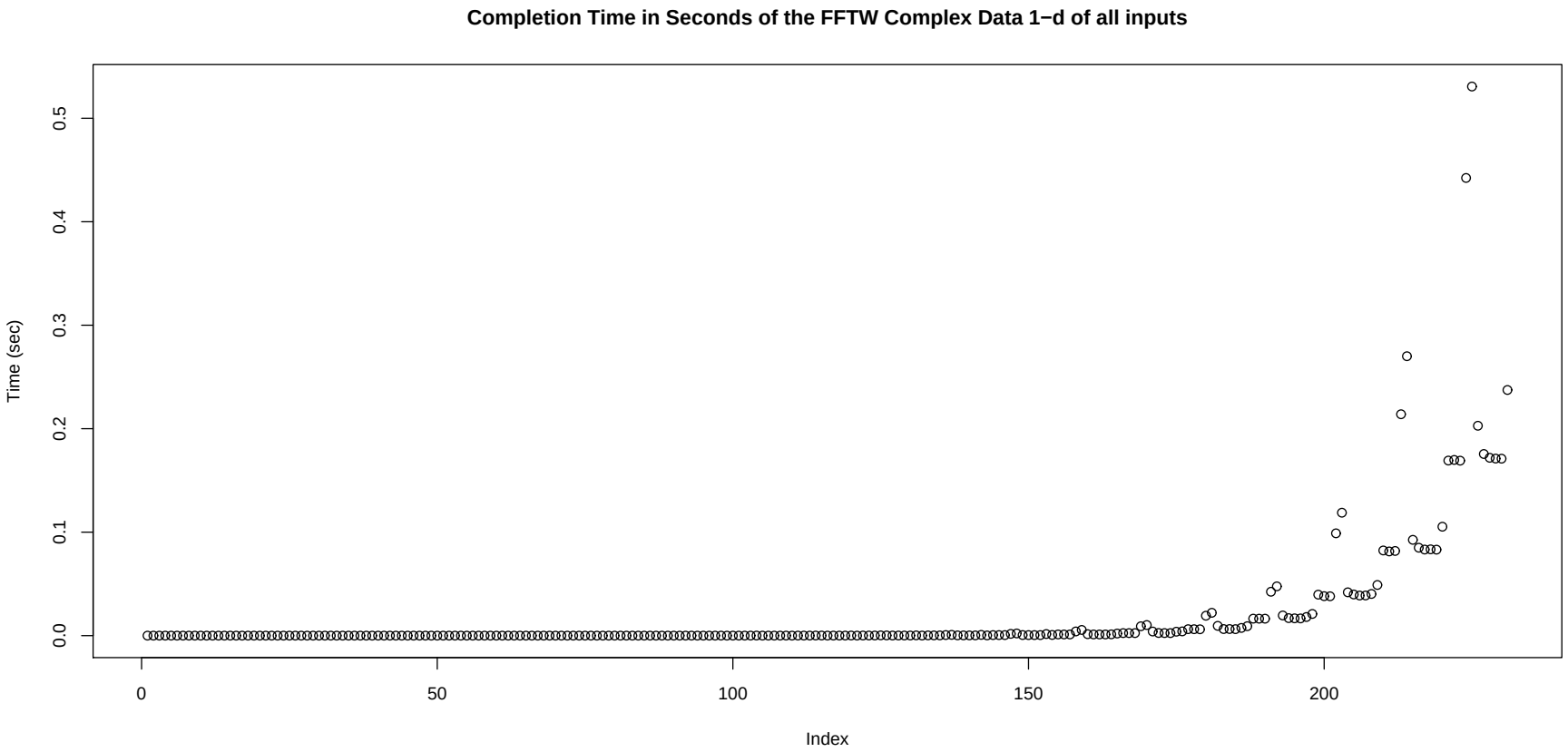


Figure F.7: Completion times of the Complex Data 1-d FFTW runs, for all input sizes.

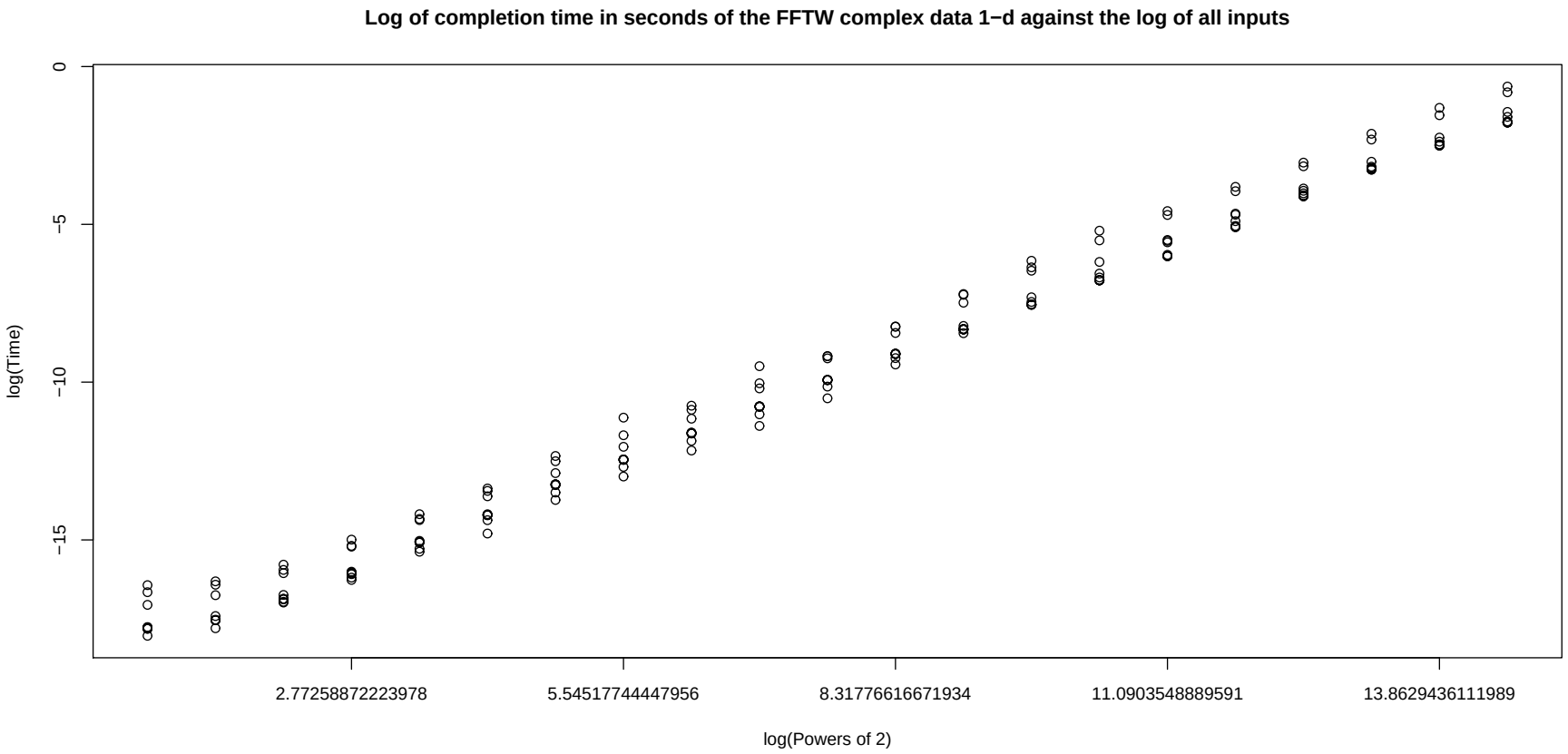


Figure F.8: Log of Completion times of the Complex Data 1-d FFTW runs, plotted against log of all input sizes.

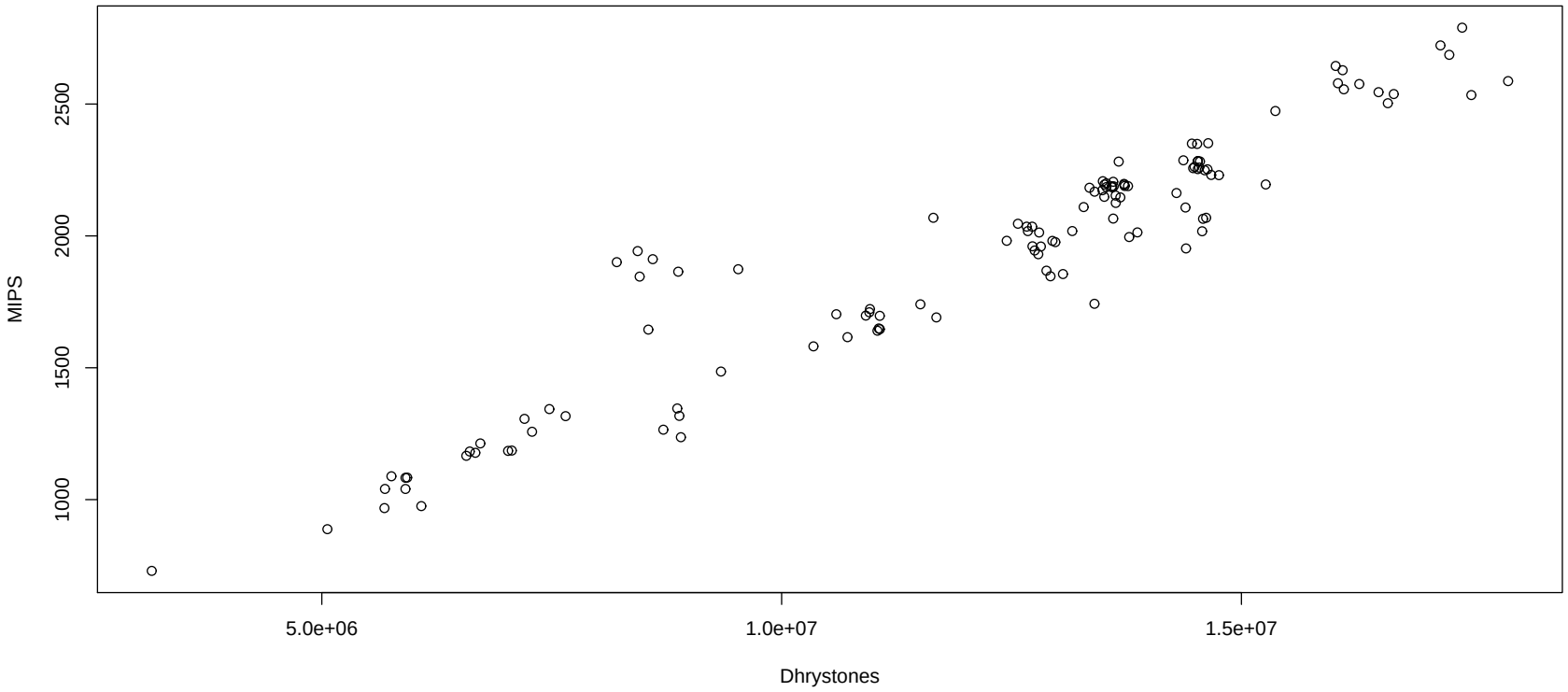


Figure F.9: Scatterplot of the EPDhrystone estimates against the EPWhetstone estimates, showing a linear relationship.

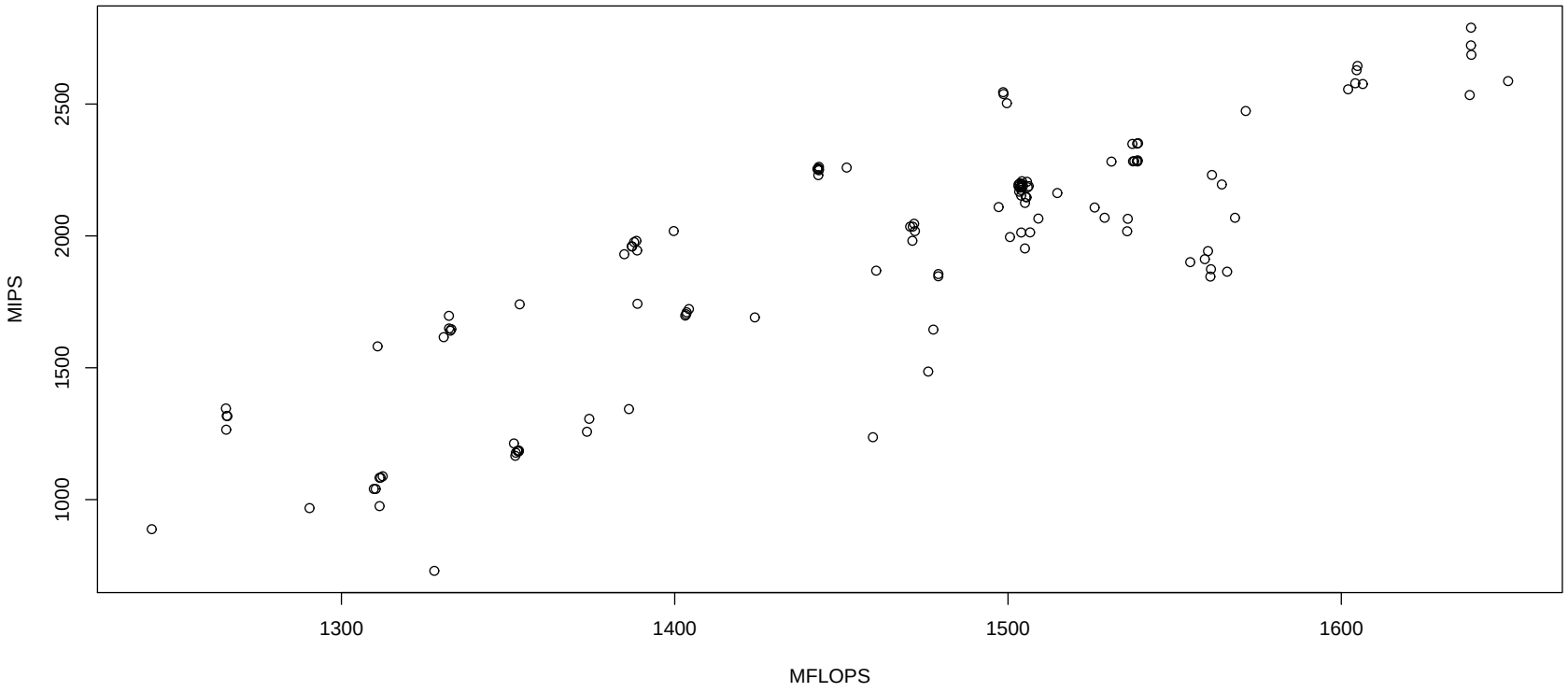


Figure F.10: Scatterplot of the EPWhetstone estimates against the EPFlops estimates, showing a linear relationship.

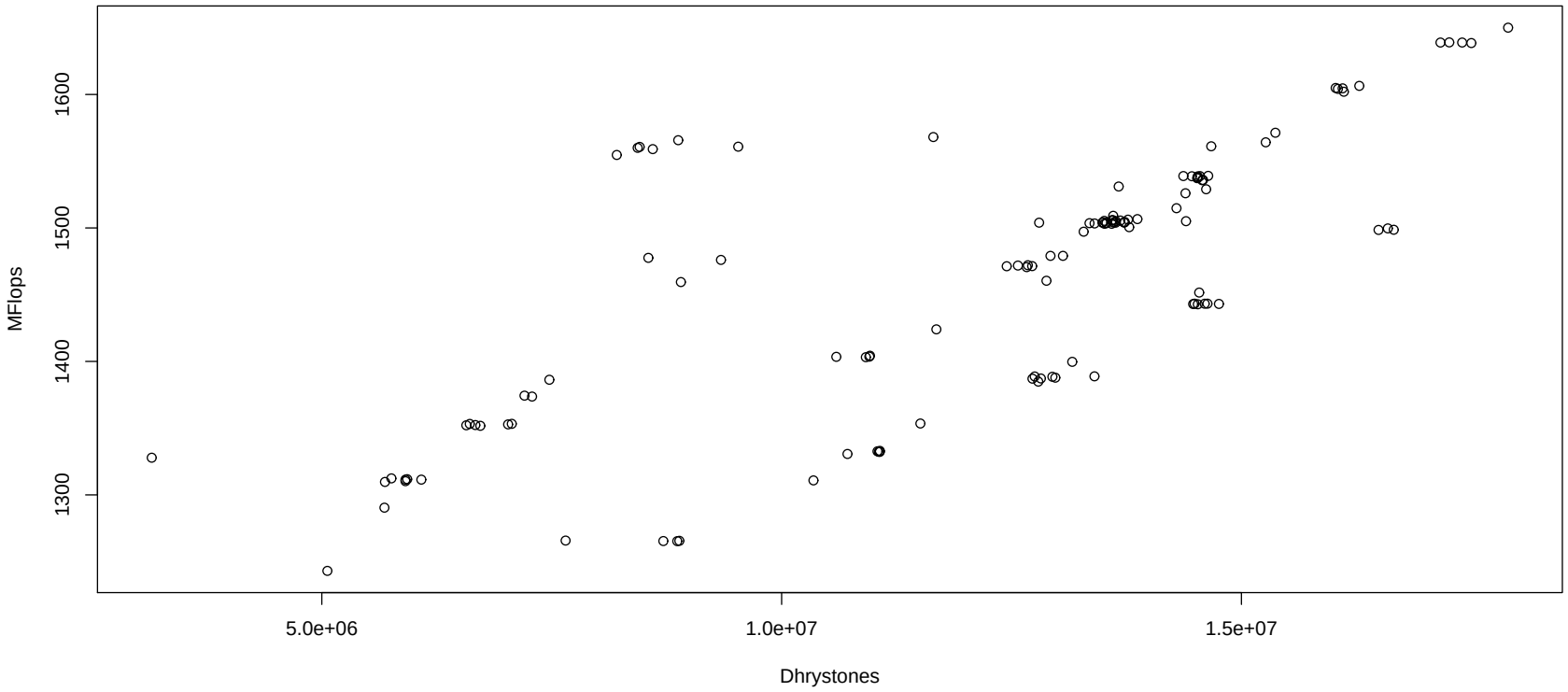


Figure F.11: Scatterplot of the EPDhrystone estimates against the EPFlops estimates, showing a linear relationship.

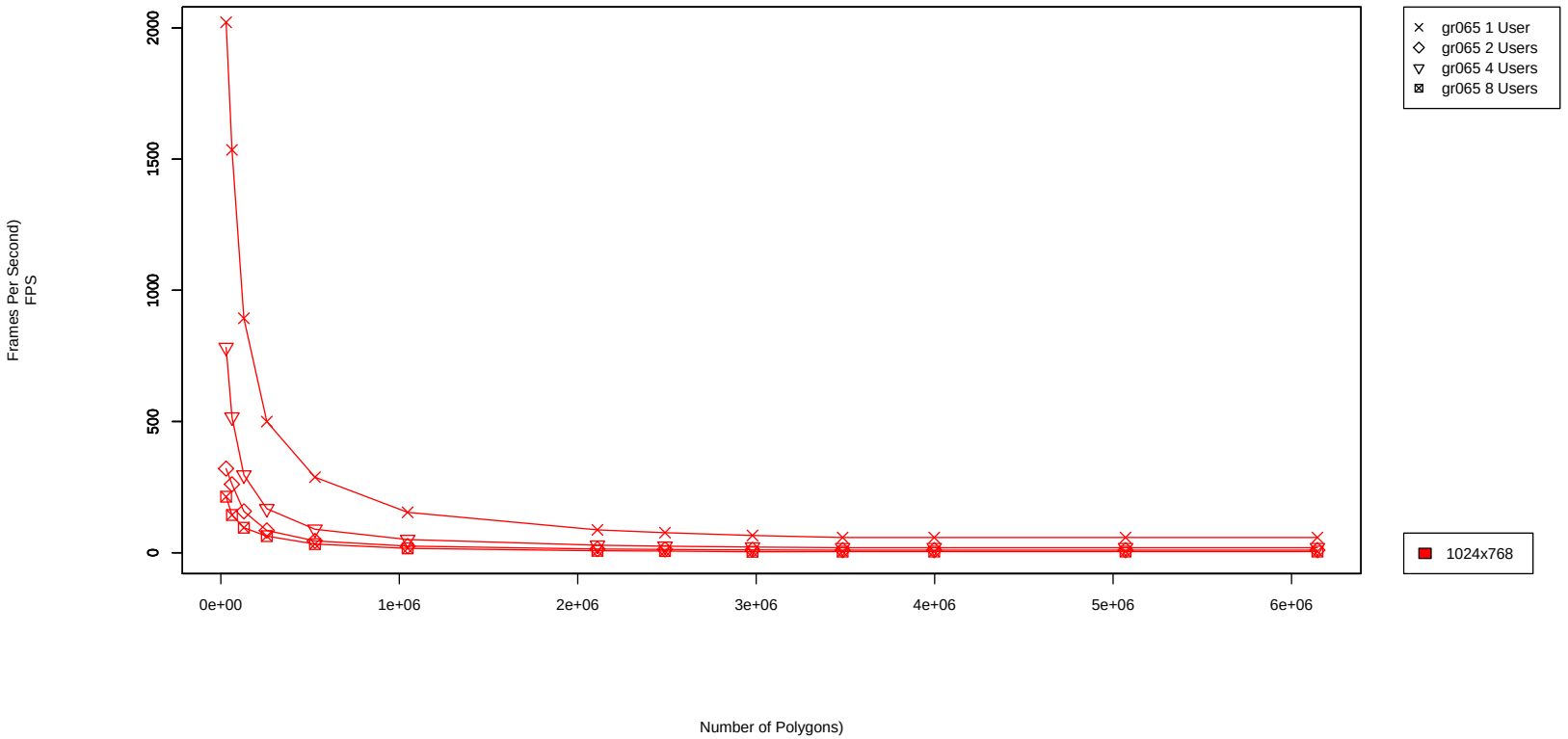


Figure F.12: Frames per second output from glxspheres running on gr065.grid.cs.tcd.ie, plotted against increasing polygon input sizes, for n users where n=1,2,4,8.

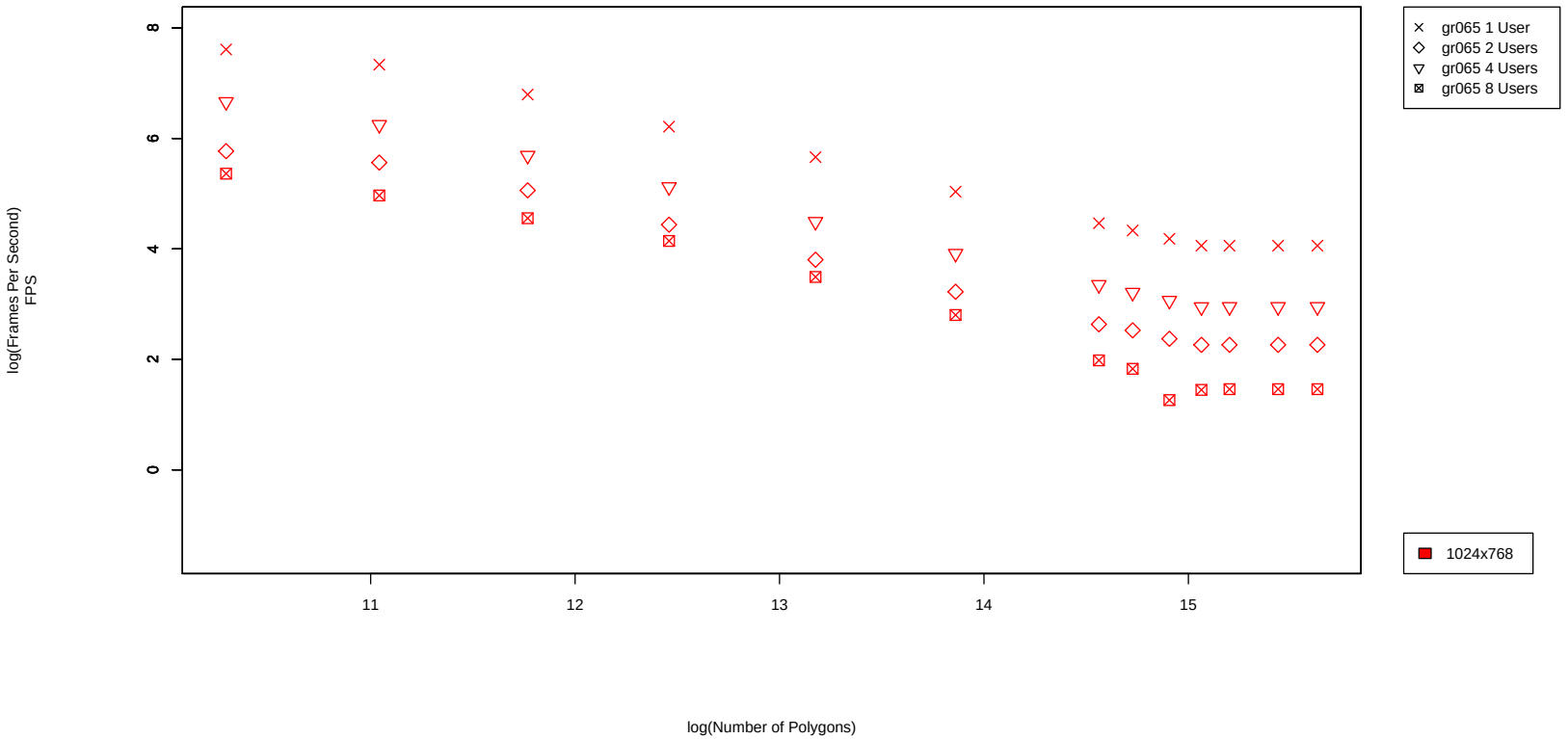


Figure F.13: Log of frames per second output from glxspheres running on gr065.grid.cs.tcd.ie, plotted against the logarithm of all polygon input sizes, for n users where n=1,2,4,8.

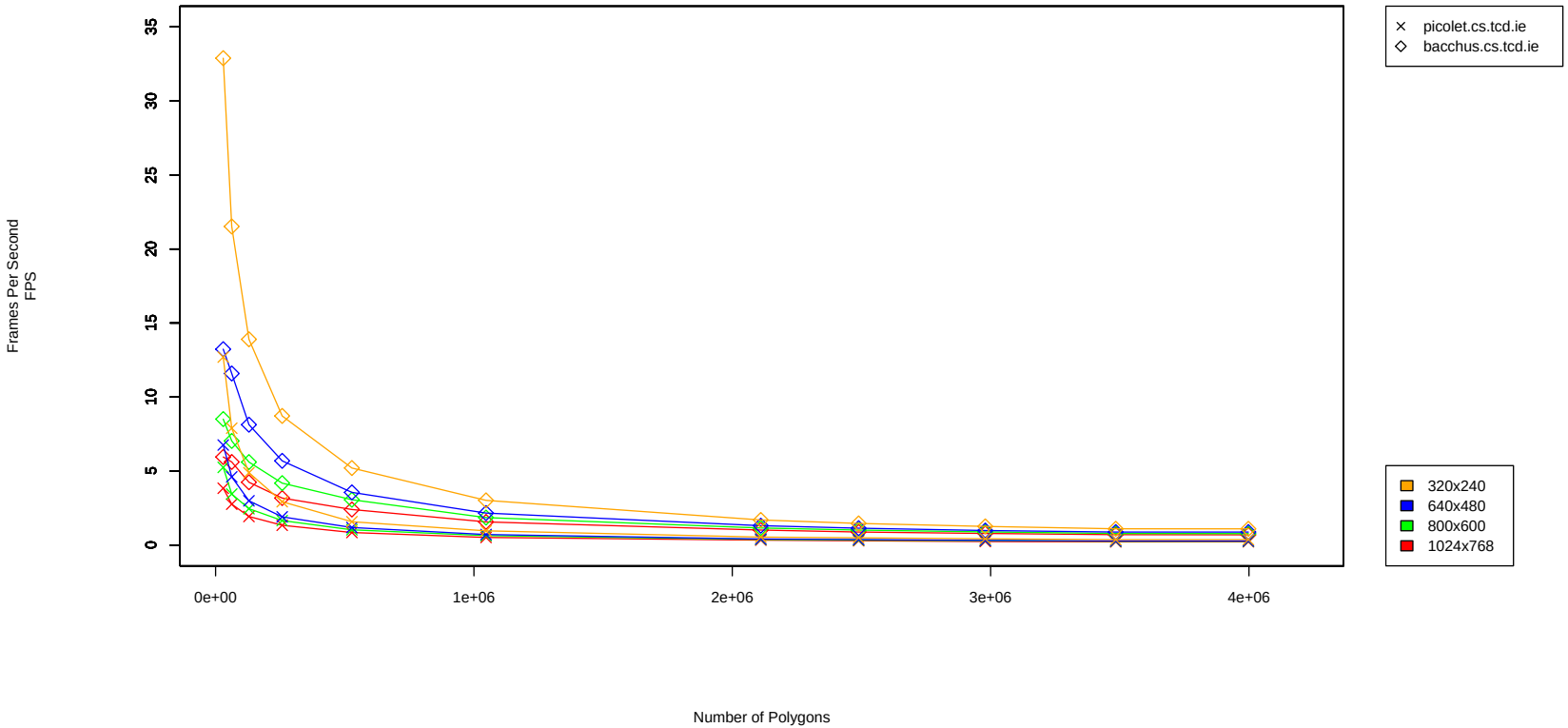


Figure F.14: Frames per second achieved by glxspheres with varying polygon input size and resolution running on *picolet.cs.tcd.ie* and *bacchus.cs.tcd.ie*.

Figure F.15: The rendered output is sent to *picolet.cs.tcd.ie*.

Figure F.16: The rendered output is sent to *bacchus.cs.tcd.ie*.

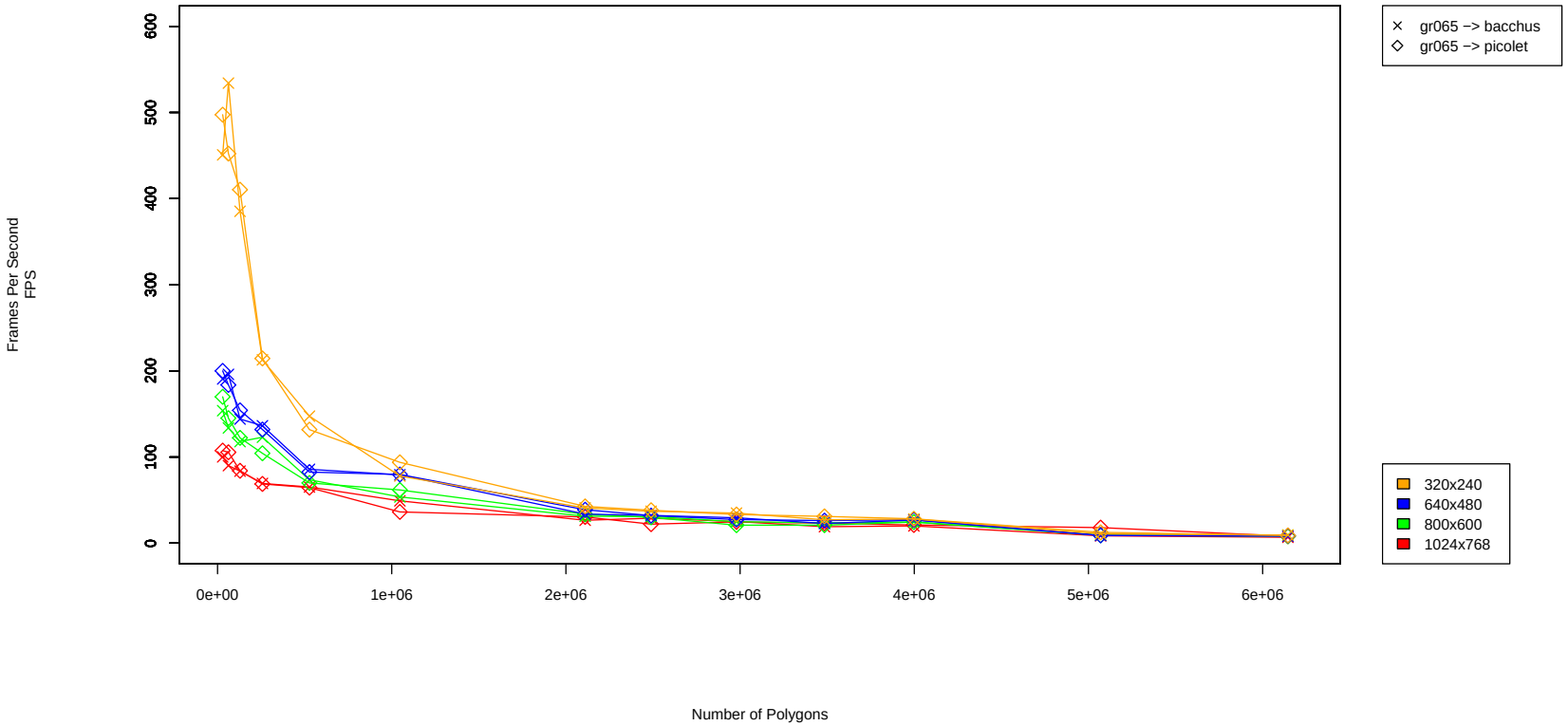


Figure F.17: Frames per second achieved by glxspheres with varying polygon input size and resolution running on *gr065.grid.cs.tcd.ie*, with the frame-spoiling VirtualGL option enabled and the rendered output sent to two desktops, *picolet.cs.tcd.ie* and *bacchus.cs.tcd.ie*

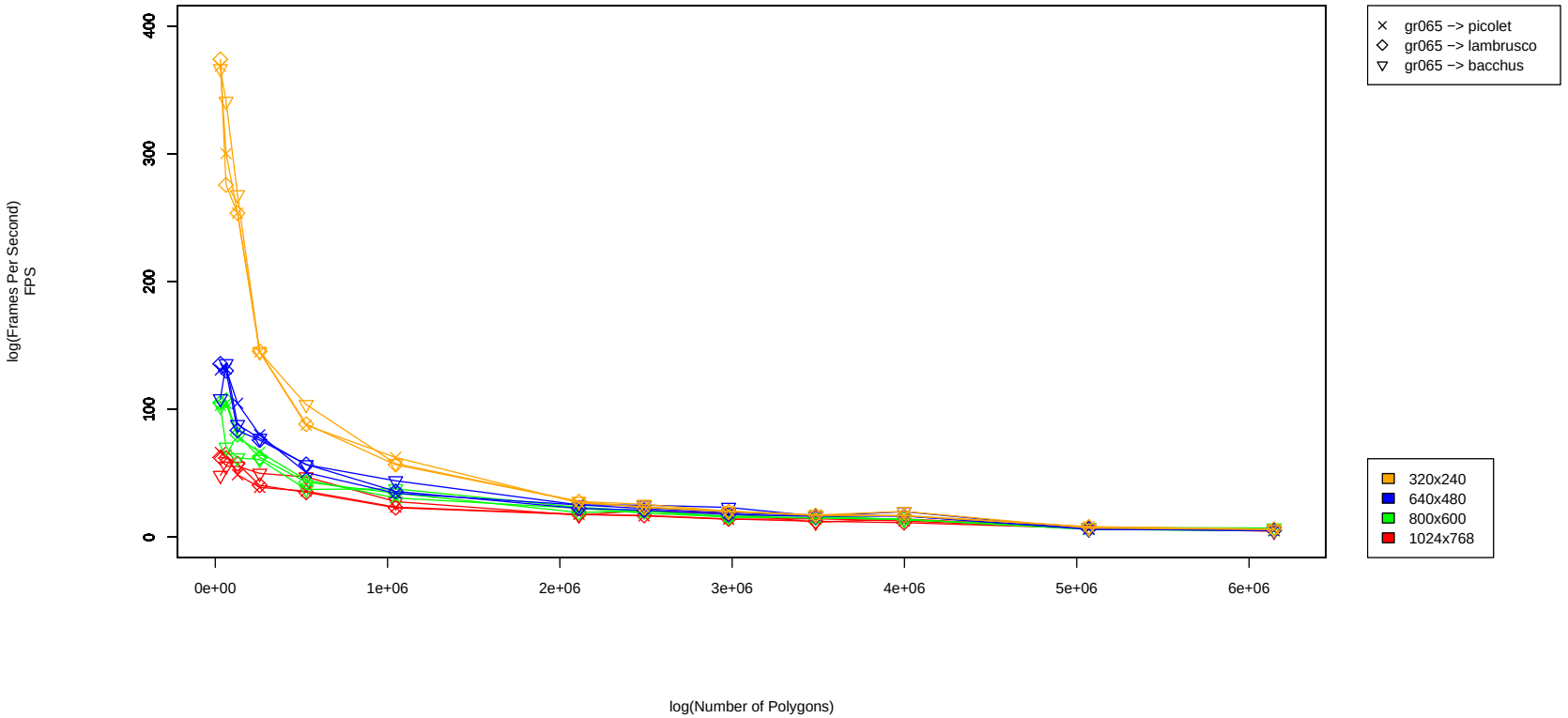


Figure F.18: Frames per second achieved by glxspheres with varying polygon input size and resolution running on *gr065.grid.cs.tcd.ie*, with the frame-spoiling VirtualGL option enabled and the rendered output sent to two desktops, *picolet.cs.tcd.ie*, *bacchus.cs.tcd.ie* and *lambrusco.cs.tcd.ie*.

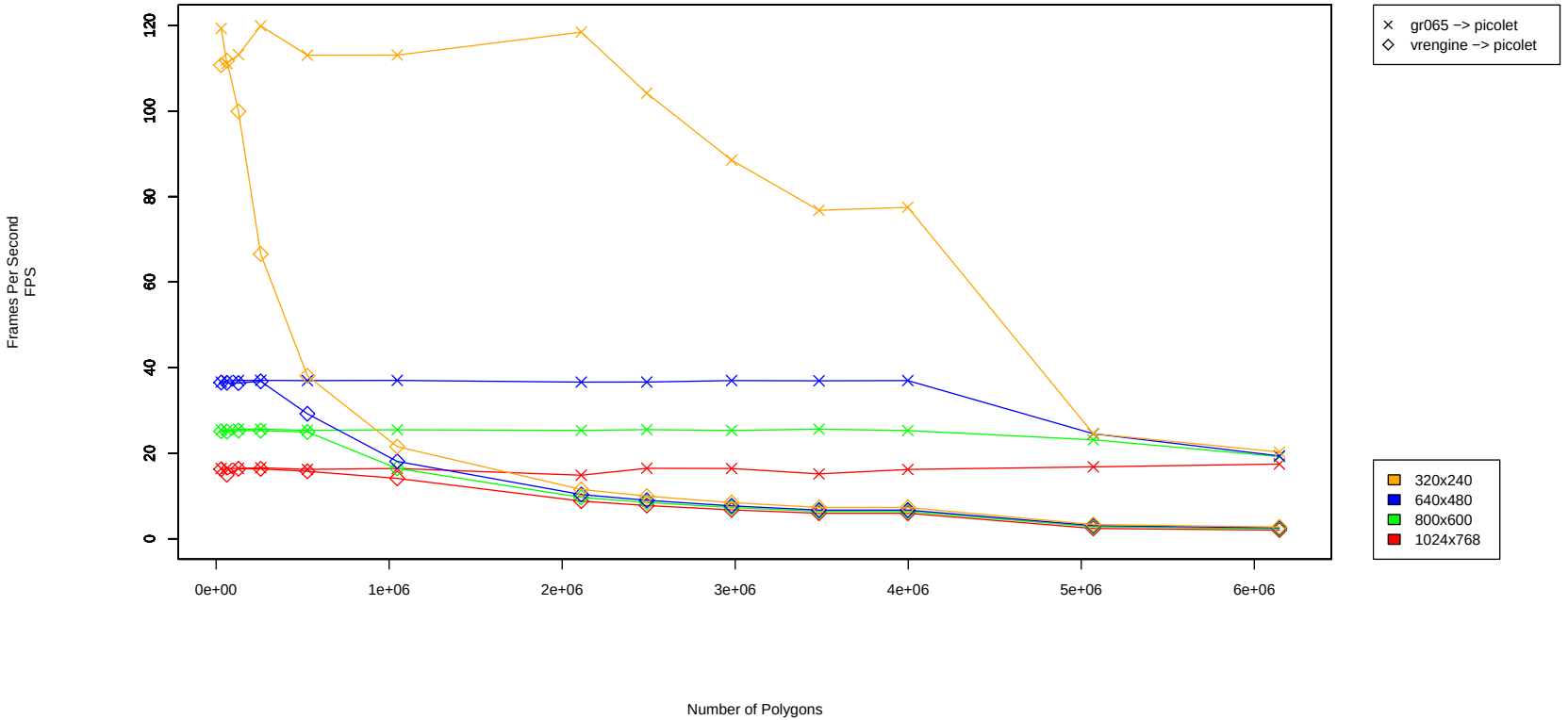


Figure F.19: Frames per second achieved by glxspheres with varying polygon input size and resolution running on *gr065.grid.cs.tcd.ie* and *vrengine.cs.tcd.ie*, with the frame-spoiling VirtualGL option disabled and the rendered output sent to a user's desktop.

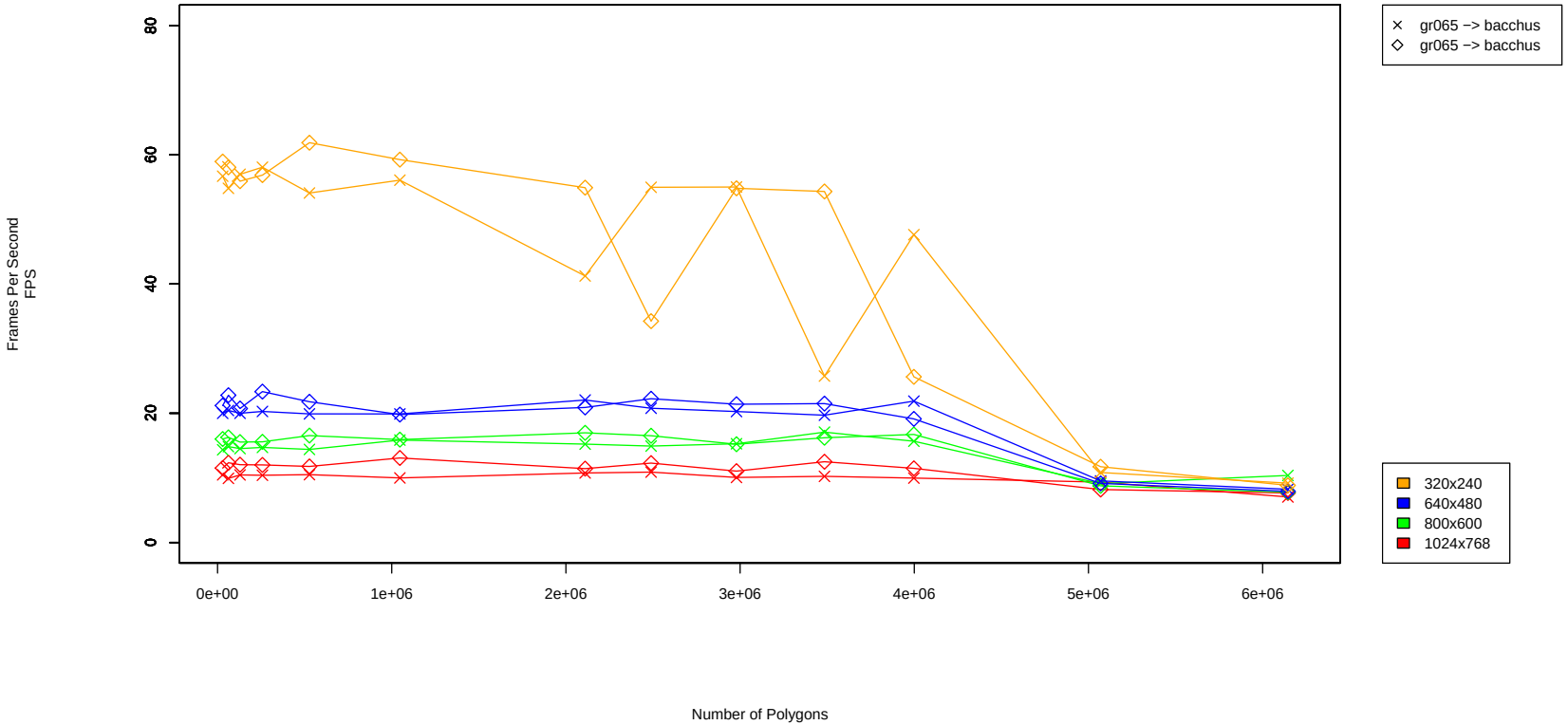


Figure F.20: Frames per second achieved by glxspheres with varying polygon input size and resolution running on *gr065.grid.cs.tcd.ie*, with the frame-spoiling VirtualGL option disabled and the rendered output sent to two user desktops.

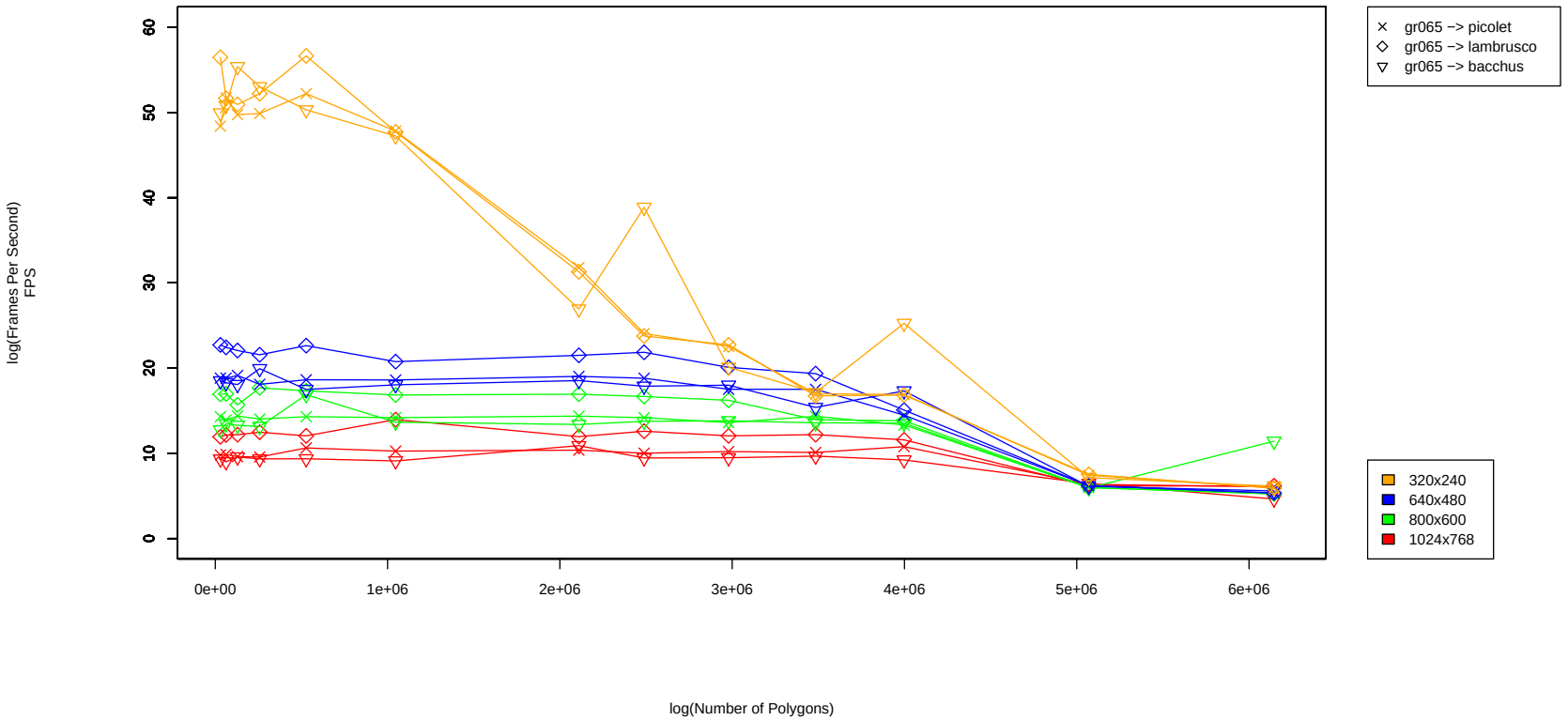


Figure F.21: Frames per second achieved by glxspheres with varying polygon input size and resolution running on *gr065.grid.cs.tcd.ie*, with the frame-spoiling VirtualGL option disabled and the rendered output sent to three user desktops.

Glossary

API	Application Programming Interface is an interface that defines the ways by which an application program may request services from libraries and/or operating systems. 12, 49, 50, 134
AR	Augmented Reality is the merging of live direct or indirect views of a physical real-world environment with virtual computer-generated imagery. 32
BDII	Berkeley Database Information Index - An OpenLDAP based implementation of an Information Index. 47
Blue Cloud	IBM's Cloud computing initiative. 16
CAVE	A Cave Automatic Virtual Environment is an immersive virtual reality environment where projectors are directed to three, four, five or six of the walls of a room-sized cube. v, 2, 5
CE	Compute Element. 43, 44, 49
Chromium	Chromium is a system for interactive rendering on clusters of graphics workstations. Various parallel rendering techniques such as sort-first and sort-last may be implemented with Chromium. 24, 42, 45
Condor	Condor is a high-throughput computing software framework for parallelization of computationally intensive tasks. 11, 131

Condor-G	Condor-G allows Condor jobs to be use resources not under its direct control and is used to talked to Grid and Cloud resources. 11
EAI	Enterprise Application Integration is the use of software and computer systems architectural principles to integrate a set of enterprise computer applications. 9
EC2	Amazon Elastic Compute Cloud - A component of the Amazon web services suite designed to allow scalable deployment of applications by allowing users to manage the creation and termination of server instances on demand. 11, 14, 15
EDG	European DataGrid - Started in January 2001, with the goal of constructing a test infrastructure to provide shared data and computing resources to the European scientific community. 12
EGEE	Enabling Grids for E-sciencE - The largest multi-disciplinary grid infrastructure in the world designed to provide a reliable and scalable computing resource available to the European and global research community. 1, 11, 12, 19, 28, 36, 47, 112, 117
EGI	A European body created with the aims of protecting and maximising European investment in e-infrastructures by encouraging collaboration and interoperability between national Grid infrastructures. 13, 41, 112
fps	Frames Per Second. 20, 68, 69, 76
gLite	A next generation middleware for grid computing developed by partners in the EGEE Project. 11, 12, 43, 47, 48, 51, 112

Globus	A software toolkit that allows sharing of distributed, heterogeneous resources. 9–13, 17, 23, 28, 43, 132, 133
GRAM	Grid Resource Allocation Manager - A component of the Globus toolkit for job submission and management. 11
Grid-Ireland	The Irish National Grid Infrastructure. 1, 43
GridBench	GridBench is a tool for evaluating the performance of Grids and Grid resources through benchmarking. 76
GridICE	A grid monitoring tool designed to allow monitoring of resource utilisation through a Web front end. 36
GVid	GViD allows the secure transport across the Grid of visual data originating from arbitrary OpenGL and X11 applications. 28, 43
GVK	Grid Visualization Kernel proposes a fully grid-enabled approach to scientific visualization. The infrastructure of GVK features a portal for arbitrary simulation servers and visualization clients, while the actual processing of the visualization pipeline is transparently performed on the available grid resources. 28
Hadoop	Is an Apache project which develops open-source software for reliable, scalable, distributed computing. 16
HDF5	A file format and library designed to store and organize large amounts of numerical data.. 25
I4C	An Agent-based wide-area service and resource monitoring solution developed at Trinity College Dublin. 36
IDE	Integrated Development Environment. 17, 19

int.eu.grid	A continuation of the CrossGrid project initiated in 2006 with the objective of providing an advanced Grid-empowered production infrastructure in the European Research Area. 1, 6, 28, 43
IPGE	Image Processing Grid Environment. 27
LCG	LHC Computing Grid - An international computing grid designed to meet the computational and data-handling requirements of the experiments conducted at CERN's Large Hadron Collider. 12, 28, 35
LDAP	Lightweight Directory Access Protocol - An application protocol for querying and modifying directory services. 9, 10
Legion	An object-based meta-system providing a software infrastructure for the interaction of distributed heterogeneous computing resources. 10, 11
LRMS	Local Resource Management System. 43
LSF	Load Sharing Facility is a commercial computer software job scheduler used to execute batch jobs on networked Unix and Windows systems. 11
MDS	Monitoring and Discovery Service - The information infrastructure provided by the Globus toolkit. 9, 10
Migrating Desktop	The Migrating Desktop Platform provides a framework which allows users to access Grid resources, run interactive applications, monitoring and visualization, and manage data files. . 28
MPI	Message Passing Interface is a specification for an API that allows many computers to communicate with one another. It is used in computer clusters and supercomputers. 11, 12

Nagios	An open source host and service monitoring application. 36
OSG	Built by a consortium of U.S. universities and laboratories, the Open Science Grid is a US national production-quality infrastructure for large-scale science. 13, 14
PBS	The Portable Batch System is software that performs job scheduling and its primary task is to allocate computational tasks, i.e., batch jobs, among the available computing resources. 11
PowerVM	A virtualization platform for UNIX, Linux and IBM clients. 16
PVM	Parallel Virtual Machine is a software system that enables a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational resource. 11
RB	Resource Broker. 49
RealityGrid	RealityGrid uses grid technology to closely couple high throughput experimentation and visualisation. RealityGrid is a collaboration between teams of physical scientists, computer scientists and software engineers in the UK. 23, 31, 135
S3	Silicon and Software Systems. 51
SAM	Service Availability Monitoring - A monitoring framework also developed at CERN. 41
SE	Storage Element. 43
SFT	Site Functional Test - A tool developed at CERN to test the operations of LCG sites from a users perspective. 41

SGI	Silicon Graphics Incorporated was a manufacturer of high-performance computing solutions, including computer hardware and software. Defunct since May 2009. 31
SGS	The Steering Grid Service provides the public interface through which clients can steer an associated application, as well as providing support for constructing distributed applications. 23, 31
Sun Grid	An on-demand computing service offered by Sun Microsystems. 11, 16
TeraGrid	A federation of US supercomputing sites which aims to provide an open scientific discovery infrastructure. 13, 21, 22
TeraGyroid	The TeraGyroid project is part of the RealityGrid Porject and uses grid technologies, high-performance computing, visualisation and computational steering capabiltities to further the research into soft condensed matter simulation. 31
Torque	The Torque Resource Manager is distributed resource manager providing control over batch jobs and distributed compute nodes. It is based on the original PBS project. 11
UI	User Interface. 43
UNICORE	UNiform Interface to COmputer REsources - A technology intended to provides seamless, secure, and intuitive access to distributed computational resources. 11, 12

VNC	Virtual Network Computing is a graphical desktop sharing system that is used to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network. 22
VR	Virtual Reality is a technology which allows users to interact with a computer-simulated environment, whether that environment is a simulation of the real world or an imaginary world. 30, 32
VTK	The Visualization Toolkit is an open-source, freely available software system for 3D computer graphics, image processing and visualization. 24, 28, 31
WebCom-G	A Science Foundation Ireland funded project that used condensed graphs to submit grid jobs. 6, 17
WMS	Workload Management Software. 112
WN	Worker Node. 44, 49
Xen	The Xen hypervisor is an open source standard for virtualisation of x86, x86_64, IA64, ARM, and other CPU architectures. 14–16
XML	Extensible Markup Language. 49–51