Development of a grid computing infrastructure to support combinatorial simulation studies of pollutant organic molecules on mineral surfaces

Richard Paul Bruin

St. Catharine's College, Cambridge

A Dissertation Submitted for the Degree of
Doctor of Philosophy

Department of Earth Sciences
University of Cambridge

November 10, 2006

Abstract

This thesis describes the work carried out to develop the *e*Minerals *minigrid* and associated tools that make it easy for scientists to use. The *minigrid* is the first implementation of the *integrated grid* idea, combining compute, data and collaborative resources, hiding unnecessary complications from the user. The computational resources include several clusters and condor pools each with the Globus toolkit installed, allowing remote job submission. The Storage Resource Broker (SRB) is used to provide data management and archiving functionality, allowing access to any results by all collaborators.

I have developed several tools that manage the whole simulation process. The main tool is called *my_condor_submit* (*MCS*), which is a powerful job submission tool that accepts a simple input file from the user and executes the specified simulation. *MCS* provides metascheduling capabilities, choosing where to execute the simulation for the user. All required input and output files are transferred to and from the computational resource by *MCS*, which also collects metadata automatically as part of the submission process.

MCS collects its metadata by making use of *AgentX*, which uses an ontology to retrieve information from a given document based on the logical meaning of the data. Ontologies are difficult for the typical scientist to understand. Therefore *ontologyViz* was developed, which visualises the ontology, allowing the user to easily construct metadata collection statements for use within *MCS*.

Other job management tools are also discussed, which automatically create and manage whole sweeps of simulation jobs. These tools take a sample simulation input file from the user, varying specified parameters and then submitting the simulations using *MCS*. The tools also include an XML to SVG graph-plotting system, allowing the user to quickly and easily see any trends in their results.

Each of these tools have been applied to several scientific studies discussed in this thesis. The main study considers the adsorption of polychlorinated dibenzo-p-dioxins (PCDDs, $C_{12}O_2H_xCl_{8-x}$) onto the $(10\bar{1}4)$ surface of calcite, determining the molecules' preferred location and adsorption energy onto the surface. Additionally, the level of cation ordering in the octahedral layer of a representative clay has been studied, identifying the temperature at which an ordering phase transition occurs within the material.

Acknowledgements

Thanks to everyone who has helped me during the time spent on both creating the work detailed in this thesis and in actually creating the thesis itself. Without all of your help, support and guidance, this work would not be what it is today. Thanks should be directed especially at the various members of the *e*Minerals project for their incessant bug-finding capabilities and the willingness to try out the various, often complex and painful processes I have sent in their direction. Especially to Kat Austen and Toby White for their help with my brief foray into science.

Special thanks should be directed to my supervisor Prof. Martin Dove for showing me his vision of the grid, allowing me to help make this a reality and to guide its future direction.

Thanks also to NERC and the Centre for Ecology and Hydrology (CEH) for funding my studentship, which enabled me to perform the research detailed in this document.

In addition I would like to thank my parents for their support through the long years I have now spent at university and for inspiring me to play with computers and to believe that I could get to here.

Last, but most definitely not least, I would like to thank my fiancée Steph for her love, and for sitting through the many hours and days widowed by the combination of my computer and this thesis.

Declaration

This dissertation is the result of my own original work, and where it draws on the work of others, this is acknowledged at the appropriate points in the text. This dissertation has not been submitted in whole or in part for a degree at this or any other institution. The length of this dissertation does not exceed the page limit.

Richard Bruin

November 2006

For Steph, Mum and Dad. Without you all nothing is worthwhile.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 *e*Science

1.1.1 What is *e*Science?

*e*Science is the term given by John Taylor, Director of the UK government's Office of Science and Technology[1], to refer to the performance of science on a large computational scale, involving distributed computing resources. The term was initially used in reference to the large funding effort made by the UK government to fund this area of research. *e*Science is known by other names within the international community, including *cyberinfrastructure* within the United States of America, where there has also been a significant investment into this research area. The name *e*Science was used in order to draw parallels with the names given to other electronic methods for performing traditional tasks, such as *e*Commerce, *e*Mail, etc.

The following quotes are taken from a talk given by John Taylor, and comprise vague definitions of *e*Science. Although these definitions are not really specific enough for the discussions forming the latter parts of this thesis, it is important that they be mentioned now so that the scene may be set for the reader.

> *e*Science is about global collaboration in key areas of science, and the next generation of infrastructure that will enable it.

[1] http://www.ost.gov.uk

> *e*Science will change the dynamic of the way science is undertaken.
> *John Taylor* [63]

A related definition is given by Ian Foster and Carl Kesselman, the initial creators of the Globus Toolkit, which is discussed later. They define grid computing to be:

> An infrastructure that enables flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions and resources.
> *Ian Foster and Carl Kesselman* [30]

Again, this definition does not quite fit with that used as the basis for the work discussed within this thesis. However, it is close enough for the time being. A more specific definition relating to the grid, as used within this thesis, incorporating compute, data and collaborative sections into a simple to use, integrated system is given later in section 1.2.1.

One of the main aims for *e*Science is the idea of bringing together researchers from both science and computer science backgrounds, allowing them to use each other's experience and skills to further each area of research. Examples of this integration include the *e*Minerals project[2] [25], within which the work described in this thesis was performed, including computer scientists, chemists and physicists, and the myGrid project[3] [60], which involves computer scientists and bioinformaticians.

Additionally, *e*Science research should leverage technologies developed within the computer science community in recent years, improving the quality of the research that can be performed by scientists. These technologies are also used to enable the scientist to perform research that they simply could not hope to consider, much less successfully perform, using traditional techniques and resource access methods.

1.1.2 Why is *e*Science different from traditional science?

The *e*Science research performed since the start of the *e*Science program has had, as its main aim, the enabling of science through the introduction of computer science technologies, such as the the grid, which is defined later in section 1.2.1.

This has meant that the science produced has been more of a byproduct of the changing processes, and that people interested in *e*Science were actually more interested in how

[2] http://www.eminerals.org
[3] http://www.mygrid.org.uk/

these processes themselves were being changed. As such, *e*Science research performed during this time considered the '*e*' much more than the '*science*', although of course, both traditional and computer scientists were heavily involved in the research. However, as *e*Science and its associated tools have approached maturity, the emphasis has refocussed on to the '*science*' that can now be performed, and that would not have been possible without the use of the *e*Science technologies.

The future aim for *e*Science is that it will cease to be a discipline in its own right. Rather, it should become part of mainstream science research. Therefore, in the future, the answer to the question '*Why is* e*Science different from traditional science?*' should be that, although in the past *e*Science concentrated on configuring tools to help the scientist, now these tools are an integrated part of scientific research.

1.1.3 Why is *e*Science different from computer science?

As mentioned above, the main aim of *e*Science, until recently, has been to enable science by introducing technologies that have previously been purely in the domain of computer science research. As such, it can be considered a sort of applied computer science, much like applied physics, which seeks to make use of methods developed within theoretical physics research.

Again, as we move into the future of *e*Science, the use of computer science technologies will become less of a key focus, and more integrated into the whole that is scientific research. This is of course the aim for most computer science research - to become far enough advanced that the user, in this case a scientist, does not even notice that they are using it.

1.1.4 Why is *e*Science important?

The aims of the different *e*Science projects have been to improve scientific research, enabling investigations that could not have been performed without the use of such technologies. This concept is an important one, because the fact that it enables scientific research that can not be done otherwise makes the importance of *e*Science perfectly clear.

A unifying aim for the whole *e*Science research area was that development within the academic community could be greatly accelerated by a large injection of funding. The

developments achieved were then to be fed back into industry and the benefits thereby experienced by the general public. In recent years many systems that started as purely academic areas of interest have crossed over into the public domain. Perhaps the best examples of this are the Internet and World Wide Web. It was hoped that such a large investment in *e*Science research would lead to such an immediate impact on the day-to-day life of the general public.

*e*Science also enables scientists to consider and make use of the increasing number of computational resources to which they have access. Both the number of these resources, and the power they each offer, is increasing at an almost daily rate. However the cognitive ability of the average scientist is not changing at anywhere near the speed required to keep up. This means that for scientists to be able to take advantage of these resources to their fullest extent, any assistance, such as that provided by *e*Science technologies, is very important.

As time passes, and the power made available increases, the more difficult it will become, for scientists to keep a mental image, and even a log of their work within their lab book. This difficulty increases with both the number of computations, and the number of resources that they are able to use at any one time. This means that the scientist will be limited to the set of work they can remember at any one time. To overcome this limitation they must use techniques, like those discussed in this thesis, to manage the submitted tasks, and in fact to allow the number of tasks to be submitted and managed at any one time to be increased by several factors.

The collaboration allowed by *e*Science methods is very important for the furthering of research in the twenty-first century. As the quantity of knowledge held about any particular field grows, a single researcher cannot hope to know everything about a subject area. This is where collaboration becomes important, allowing the knowledge and expertise that can be targeted at a single research area to increase with the number of researchers involved. This increase in knowledge means that the science performed can be improved, and that less time will be spent 're-inventing the wheel', because collaborators need not repeat each other's previous work.

1.2 Grid Computing

1.2.1 What is grid computing?

When people talk about *e*Science, they often discuss '*The Grid*' as being an integral part, if not the all-encompassing idea behind *e*Science. However, there is no '*The Grid*' and in fact, there are many different meanings for the terms '*grid*' and '*grid-computing*' with each having their own advantages and proponents.

The term '*The Grid*' was given to this area of research in order to imply similarities with systems like utility grids, for example, national electrical grids, where resources are available for all to use, and users simply need to 'plug in' and use what they require. This is the definition historically proposed by Ian Foster and Carl Kesselman in one of the first publications on the topic in 1998 [32].

Proponents of '*The Grid*' believe that the computing industry should head in this direction, providing many centrally owned and operated computer systems, available for all to use. All the user needs to do is 'plug in' and use the computing resources they require, of course paying any associated cost. This method for delivery of computing resources is not currently normal, with people still buying their own resources, which they are then often reluctant to share with others. However, there is one notable exception to this trend, called *Sun Grid*[4], where access is given to hundreds of processors for $1 per processor-hour. Whilst grid systems make charges on this scale, it is unlikely that they will become successful. One month of constant usage of one of Sun Grid's processors would easily cost enough to purchase a computer of comparable power that will operate, with minimal risk of hardware failure, for many years.

The shift towards this '*utility computing*' paradigm is a large change, requiring a lot of effort to divert the momentum held by the computing industry's current usage models. As such, *Sun Grid* is still in the early stages of community uptake and there is little evidence of the scale of research that has been performed using the utility. So far, no concrete figures have been published by Sun.

While it is clear that '*The Grid*' includes this type of *utility computing*, the type of resources that should be provided in this manner, and how they should actually be used is not so clear. In addition, it is not clear what resources should be provided to assist with data management and collaboration, both of which are important within '*The Grid*'

[4]http://www.sungrid.net

and *e*Science. This lack of clarity is exemplified by the way that different people talk about '*The Grid*' and what it means to them. Several of these different meanings will now be discussed, with the definition used within this thesis given last, in section 1.2.3. It should be noted that these definitions are related to one another, often blurring where one definition ends and another begins.

1.2.2 Different definitions of '*The Grid*'

Computational grids

The first meaning given to '*The Grid*' will be referred to within this thesis as a 'computational grid'. This type of grid is that most similar to traditional scientific computer usage, in that it is actually concerned with making processing power available for use by scientists. However, computational grids are not simply a new name for distributed, or high performance computing as have been used in the past. Rather, they are a new way of accessing and using such resources. A computational grid is also the definition of a grid system closest to the definition first proposed by Ian Foster and Carl Kesselman in [32], as mentioned above.

One well-known computational grid project is the LCG project[5] [40], which is to be used to process and help with the analysis of huge amounts of data. This data will be produced by the new Large Hadron Collider (LHC)[6] instrument, which is due to be commissioned during 2007 at CERN[7] in Switzerland. The LHC will produce several terabytes of data to be processed per performed experiment.

Computational grid resources are those often considered the most important by scientists new to grid computing, since having access to more computing resources leads to the ability to perform more simulations. However, simply adding more computational resources does not guarantee that more research can be performed. This is due to the very large amounts of data created when running larger numbers of simulations, which can in effect be considered a '*data deluge*'.

Computational resources can be contributed from existing resources, or can be purpose bought dedicated resources. The key point is that the resources should all be accessible

[5] http://lcg.web.cern.ch/LCG/
[6] http://lhc.web.cern.ch/lhc/
[7] http://www.cern.ch

to users in a consistent manner, allowing the user to concentrate on the research they want to perform using the resources, rather than the resources themselves.

Typical resources within computational grids include each of the different types of related computational systems described in section 1.3.2.

Data grids

An additional meaning often given to '*The Grid*', which is heavily biased towards data management will herein be referred to as a 'data grid'. This type of grid has the aim of allowing simple yet powerful data discovery and sharing between collaborators, and tends to be concerned with short to medium term data storage and access.

One of the UK *e*Science testbed projects concerned with this type of grid is the 'Open Grid Services Architecture - Data Access and Integration' (OGSA-DAI) project[8] [4] and [39]. OGSA-DAI is concerned with developing software to assist in the integration of, and access to, disparate data sources and their associated data via '*The Grid*'.

Another data grid project is the 'NERC data grid' (NDG)[9]. The NDG aims to create a data grid through which all data related to several key NERC projects will be accessible. Several large datasets relating to different research areas will be combined, which will allow for enhanced data discovery and delivery for all participating members, even when accessing their own data.

The importance of data grids is typically not as immediately apparent to scientist users when compared with computational grids above. However, once the scientist has received access to the sort of computing resources made available by a computational grid, it soon becomes clear how important associated data grids are. Without the associated data grids the scientist cannot hope to cope with the deluge of data created when using computational grids.

The simple archival of these data files means that the scientist can always come back to them, which is very important and is something that has not been feasible prior to the use of a data grid. Previously the user attempting this archival would have quickly filled their available disk space. The addition of extra space would then result in the user soon forgetting which files were stored on which disk. These problems are tackled by the use of data grids.

[8] http://www.ogsadai.org.uk
[9] http://ndg.badc.rl.ac.uk/

Part of the task of managing the data created when using '*The Grid*' is the enabling of data sharing between collaborators. Should the scientist actually manage to cope with the data deluge without the use of a data grid, then it is highly likely that in a very short timescale the data would become lost within some sort of complex filesystem hierarchy. This, understandably, does not lead to the scientist being able to share their data with collaborators in an efficient manner.

The use of data grids can make this collaboration process much more efficient and easily manageable. It also shows how the boundary between the different types of grid can easily be blurred.

Data grid resources can fall into the following categories:

Data management. Tools within this category handle the created data directly, storing it for the user. All data are presented through one logical system, even though the data may actually be distributed between different locations and storage systems. These systems also take care of controlling access to the data, allowing collaborators to be given access to data without requiring them to be given accounts on each of the different computers on which data may have been stored.

Metadata management. Metadata is commonly referred to as '*data about data*'. It is often used to provide information about the context in which data was created and used. This allows for users and their collaborators to search for data according to its related context, which is a much more powerful approach than simply searching by filename, or filesystem hierarchy.

An in-depth discussion of the usage of metadata within my job submission tools is given in section 3.2.3. In addition, a general description of the metadata tools used within the *e*Minerals project, but developed by other project members, is given in appendix C.

Collaborative grids

The third meaning given to '*The Grid*' refers to collaboration and interaction between project members themselves. This definition is called a 'collaborative grid', which is how it will be referenced for the rest of this thesis. Work on grids of this sort tend to involve enhancing the types of communication between project members, through the use of tools such as scalable, multi-participant video conferencing and associated applications.

One widely used project targeting this sort of grid is the Access Grid project[10], developed at Argonne National Laboratory (ANL)[11]. The Access Grid project provides such a video conferencing framework with additional features such as digital whiteboards, and application sharing called 'The Access Grid Toolkit'.

Collaborations such as the *e*Minerals project, involving grid computing, are often known as *virtual organisations*. The term *virtual organisation* can be defined as '*A group of collaborators, whose aim is to achieve a common goal through the sharing of grid resources*'. This title fits well with the aims of the grid projects, and particularly with the collaboration interests within these projects.

The collaborative tools used within the *e*Science projects include many mainstream tools, which would often not be considered to be grid tools, because they are so common in standard computer usage. However, their combination in the grid computing environment allows them to be considered to be grid tools. These tools include eMail, Instant Messaging (IM), Video conferencing (The Access Grid), and Wikis. One final type of collaborative tool that is not yet as well known outside of projects such as *e*Minerals, is application sharing tools, such as the Multicast Application Sharing Tool (MAST) [38]. These tools allow the user to share any program they are using with one or more collaborators, with each collaborator being able to control the program simultaneously.

My contributions to this area of grid computing are not appropriate for documentation within this thesis, but can be found in [24] and [27], or on the *e*Minerals website[12].

1.2.3 Integrated grids

It can be argued that none of the above meanings quite captures the whole point of '*The Grid*'. If '*The Grid*' is to become useful in the mainstream scientific, academic, and even public communities, then none of these meanings alone covers everything that is required. This implies that some combination of the definitions will be needed. This is a view held within the *e*Minerals project, which will be discussed further in section 1.4.

While this combination need not cover the whole space addressed by each of the meanings of grid individually, it should contain the most useful parts of them. This type of grid and its relationship to the others can be easily visualised by figure 1.1.

[10] http://www.accessgrid.org
[11] http://www.mcs.anl.gov
[12] http://www.eminerals.org/papers/

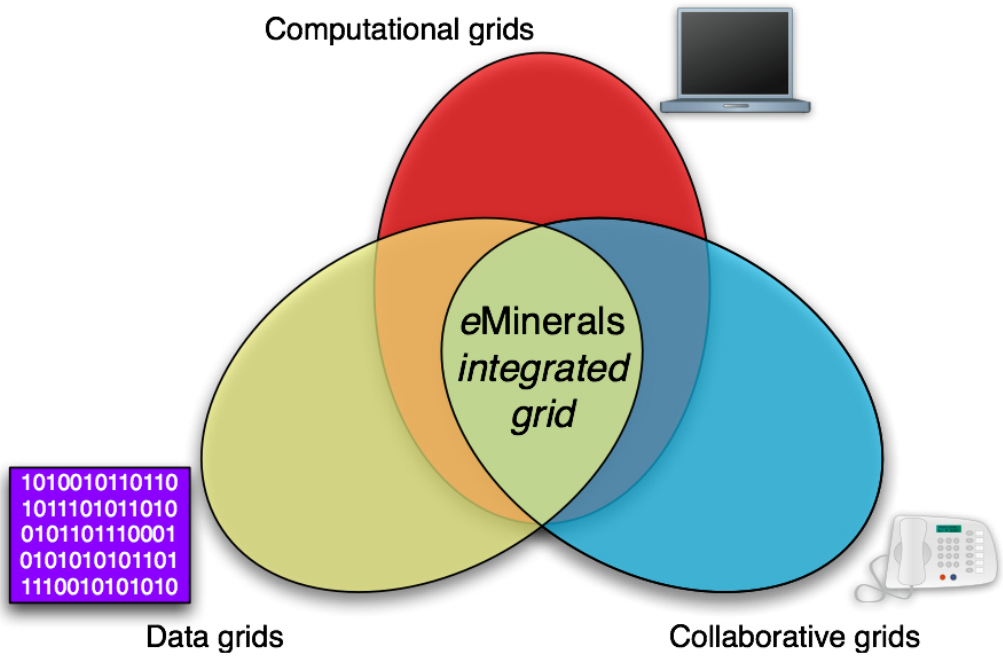

Figure 1.1: Three of the common definitions of '*The Grid*', with the *e*Minerals *integrated grid* concentrating on the central intersection

The *e*Minerals project attempts to consider the middle ground of these different definitions, and covers the intersection of the three definitions, labelled '*e*Minerals integrated grid' within figure 1.1. As such, when the term '*integrated grid*' is used for the rest of this thesis, it will refer to this middle ground between the different definitions, where the useful parts of each of the definitions will be combined.

To clarify, an *integrated grid* can be defined as '*A system in which access to all compute and data resources is presented through a uniform interface, or set of interfaces, hiding the specifics of the resources and their locations from the user. The system must be backed up by system-wide communication, using emerging video conferencing and application sharing techniques. Intelligent sharing and archiving of data through the use of systems such as metadata, must also be incorporated*'.

1.2.4 Grid computing strengths and weaknesses

As should be clear from the preceding discussions, there are many advantages for the average scientist associated with grid computing and its related technologies. Some of

these advantages relate to the quality of the research that can be performed by the scientist, whilst others relate to the ease with which the scientist can perform and manage their research portfolios.

Considering the quality of the research that can be performed, the use of grid systems mean that the scientist will, in the future, be able to access what can effectively be considered limitless computing resources. Currently this accessibility is limited, due to the fact that grid technologies are still effectively under test conditions. It can, however, be expected that once the worth of techniques of this sort have been proven, they will become much more widespread and perhaps, even ubiquitous.

Simply having access to resources on this scale does not on its own lead to the benefits associated with grid systems. This is due mainly to the fact that submitting and keeping track of enough separate computational tasks to take advantage of computing resources on this scale is not possible using traditional methods alone. This is where the combination of techniques and methods associated with *integrated grids* come into their own, enabling the scientist to manage research on this scale.

Although grid techniques have been designed to allow the scientist to manage research on the scale enabled by such techniques, they can also be applied by the scientist running a single calculation at any one time. This application leads to many of the same benefits as experienced when using the large scale systems accessible via grid systems. For example, improved accountability of each individual simulation and its aims, easier management of the simulation submission process, and improved data archival, are all experienced by users at any scale.

A final strength associated with grid systems, which was not experienced during early stages of grid computing usage, but towards which my work contributes, is the highly usable nature of the tools. The average scientist user of these tools should not be required to have any interest in computer science, or the in-depth techniques required to use the new resources that are becoming available. These tools must therefore hide these details from the user, which is the key feature required when making tools usable.

Making the resources simple to access, using uniform access methods across all such resources will be important and a key strength of grid technologies. This is because it means that the scientist no longer needs to care about the resource to which they are submitting, other than whether it is powerful enough to perform the calculations required, and whether the project budget permits its use. All the user need do is perform the same submission procedure whatever and wherever the target resource. The submission procedures used within the *e*Minerals project are discussed in detail in chapter 3.

As grid systems are still an emerging technology there are, of course, some weaknesses associated with the technologies and methods involved. The work documented within this thesis attempts to reduce these weaknesses as far as possible, while emphasising the strengths outlined above.

Grid middleware systems are currently too difficult to install and to use. In addition, the set of features provided does not cover all of the work that the scientist may wish to perform. These weaknesses are simply due to the fact that as computing systems and technologies mature, they generally become simpler, as well as becoming more feature complete. This is a well known feature of computing systems, both in research and mainstream user communities. As such, these weaknesses can be considered transient, and will be overcome as the grid computing field matures.

In addition, there are some weaknesses that cannot be attributed to the immaturity of the field. Firstly, the owners of any resources contributed as a part of a grid system will lose some of their control over the resources. For example, a collaborator could be making use of the resource to carry out a simulation, meaning that the owner cannot use that resource straight away. However, this disadvantage is reduced by the fact that other resources are likely to be available elsewhere within the grid, so the fact that their own resources are not available does not actually matter. Also, if the owner really needs to use their own resource then they can suspend the other user's task, before allowing it to continue once their own task is complete.

Secondly, since the user does not necessarily know where their submitted jobs are executing, it is more difficult for the user to keep track of the status of these jobs. This means that the user may not be able to monitor their jobs' progress, checking for unexpected behaviour, etc. This weakness is not as easy to avoid as the others, although its effects can be reduced through the submission of jobs that only run for a set amount of time, before reporting back to the user. These jobs can then be checked, and if behaving correctly, restarted from where they left off. However, this is not ideal as it involves the user checking each job manually, possibly several times, depending on the length of the jobs involved.

1.3 Related techniques and systems

1.3.1 An overview

The *integrated grid* idea is a completely new concept, with no other techniques and systems covering the whole range of systems that it includes. However, many of the related systems are in fact included as components of an *integrated grid*, each making different contributions.

The following sections will discuss several different types of computational system. Each of these can make valid contributions to an *integrated grid*. However, none of them individually provide the flexibility afforded by their combination. Also, they do not consider the data or collaborative issues, which the *integrated grid* addresses so well.

Related data and collaborative systems are also discussed, which obviously do not consider the computational aspects of the *integrated grid*. They do, however, provide relevant contributions, with many of the data and collaborative tools used within an *integrated grid* also being used in pure data and collaborative research settings.

Each of the related techniques and systems will now be discussed, with their relevant similarities and available contributions highlighted.

1.3.2 Computational resources

Broadly speaking, computational resources can generally be divided into two different categories. These categories are related to, and often contributed as components of an *integrated grid*. The first category, High Performance Computing is targeted at the application of ever more powerful resources to a few large computational challenges. The second category, High Throughput Computing, is aimed at the performance of large numbers of independent computational executions in parallel to one another. These high throughput resources can in turn be divided into several subcategories.

It is important to note that the boundaries between the different categories are often blurred, with concrete definitions very difficult to state. Indeed, the only differences between the high throughput subcategories are due to the manner in which they are configured and used, with the underlying hardware being the same throughout.

Traditional computational resources do not typically consider data issues. Once the data

has been created, they are made available to the user from the machine's submission node. Other than that, the only consideration typically given to data on these machines is that quotas are often enforced, stopping the users from leaving their data on the resource indefinitely. This means that users must be careful to collect the important data from the resource, storing them for later use, and then removing any data that is not deemed important. This is not ideal, and can be considered to promote the scientist's typical data loss problems.

Each of the different computational resources discussed below are important in their own right. Individually, none of them are able to tackle all challenges studied by projects such as the *e*Minerals project, although they can when used in combination. It is for this reason that the *integrated grid*, and specifically its first implementation, the *e*Minerals *minigrid*, combines many different types of computational resource. This combination and their integration will be discussed in chapter 2.

High Performance Computing (HPC)

High Performance Computing (HPC), is the name given to the area of the computing industry dedicated to achieving ever more powerful parallel computers. These machines are often designed to operate best when applied to specific problems, for example, weather prediction or climate modelling. In fact, these machines will perform relatively poorly when compared with standard desktop computers for small applications, such as those performed by the average computer user.

HPC machines are hugely expensive computers, which often fill whole warehouse-sized rooms. The Japanese 'Earth Simulator' HPC machine was named as the world's most powerful supercomputer in 2002, consisting of over 5000 processors. It cost around $350 million (US) to build and was designed to be applied to climate and earthquake modelling. Although this machine combines so many processors, the majority of the cost relates to the purchase of specialist networking interconnects, which are much more expensive than the commodity interconnects used in less powerful machines.

The cost associated with HPC machines does not guarantee longevity, as might be expected. For example, the Earth Simulator was replaced as the most powerful supercomputer in the world by an IBM machine with twice the benchmarked power only two years after its initial installation.

HPC computers are generally best suited to systems that have requirements for very large amounts of memory or processing power. These systems must be split over many

processors with as little inter-processor communication as possible. This division means that the implementation can take advantage of the increase in power and memory afforded by the extra processors, when compared with execution using a single processor.

It is worth noting, that the speed achieved by HPC machines is mainly due to the speed of the interconnects and the number of processors provided over which studies are able to parallelise. In fact, the individual processors used within HPC machines actually tend to be slower than those found in a typical desktop computer. It is purely their number, and the connection technology used, that makes them so powerful in combination.

The division of processing is different to that afforded, and indeed required, by High Throughput Computing, as described below. This is because the calculations to which HPC are applied cannot be divided into tasks that are completely separate from one another and so do require some inter-processor communication. This is where HPC computing and the specialist network interconnects employed come into their own. No other type of computer is able to tackle the problems at which HPC machines are targeted.

The power provided by both commodity and HPC resources have increased at astounding rates. Currently, the most powerful supercomputer in the world is an IBM BlueGene/L system, owned by the US government's Department of Energy (DOE) and installed at the Lawrence Livermore National Laboratory[13]. This machine has achieved a benchmarked performance of approximately 281 TFlops[14] according to the June 2006 edition of the Top 500 Supercomputer Sites list [43].

Scientists have developed the aims of their research in response to this increase in the power of available computational resources, and in light of their previous research accomplishments, meaning that typical simulations now go far beyond the scale that was possible only a few years ago. This means that there is definitely still a need for HPC resources to study these new research topics.

In contrast, some parts of a scientist's research have not changed, and so calculations that required the use of a supercomputer in the past, can now be performed using the scientist's own desktop computer. These calculations now require a fraction of the previous execution time and can be performed for a fraction of the cost.

An example of the movement from HPC to desktop machines is work performed using the DL_POLY [64] simulation code. DL_POLY is used extensively within the compu-

[13] http://www.llnl.gov

[14] 1 Tflops is equivalent to 10^{12} floating point operations per second.

tational chemistry community and is now perfectly capable of being used on standard commodity components. However, it was initially developed to be used only on HPC resources, because the problems to which it is targeted could not be considered using smaller resources. The increasing power available from commodity resources has meant that DL_POLY is now used on smaller resources for smaller systems, and only the very large systems studied require the use of machines as powerful as HPC resources.

High Throughput Computing (HTC)

High Throughput Computing (HTC), is the name given to the area of the computing industry dedicated to obtaining ever more computing power through the use of standard desktop computers and networking interconnects. Whilst HPC computers rely on very fast communications between their processors, HTC is concerned with the idea of running many separate tasks concurrently, and as such does not need the expensive interconnects, etc. that are required for HPC.

HTC resources are generally best suited to studies that can be parallelised with absolutely no inter-processor communications. Such tasks are known as *embarrassingly-* or *trivially-parallel*, because any one section of the study can be performed without interaction with, or any dependency on, the other sections. An example of this sort of problem would be the study of the changing climate in the coming decades. In this study the same initial configuration is specified with many different parameters, which in combination, can possibly affect the climate. Each different combination of these parameters can be considered completely separately from one another. This would allow the scientist to consider in great detail how the combination of different parameters affects the global climate, and so to advise decisions regarding ways to prevent adverse changes from occurring.

The name HTC is used both to express the similarities, and also the important differences when compared with the similarly named HPC. It is worth noting that the sort of systems for which HTC is designed would not achieve an equivalent level of performance when using HPC resources, due to the lower power of individual processors within HPC systems, compared with those in typical HTC systems.

As the power of computers increase, more and more parts of the scientist's research can be performed on less specialised resources, such as those provided by HTC. The use of available HTC resources to perform smaller tasks, means that the scientist can then budget HPC time to the execution of calculations requiring more power than can be

provided by commodity equipment alone.

However, HTC systems, due to their distributed nature, are not ideal for parallel computations, which run using more than one processor, each of which communicating via a network. This is because the communications between any two or more processors in an HTC system will not typically be fast enough, meaning that the benefits gained from using multiple processes are negated by the time spent waiting for communications.

There are two main types of HTC implementation. Firstly, those that make use of resources which have other uses and which volunteer their spare processing power. These resources are often known as distributed, volunteer systems. The second type of implementation makes use of dedicated resources that have been purchased with the intention of their combination and use in this manner. These resources are often known as commodity clusters. Both of these implementation types will now be discussed.

Distributed, volunteer systems

Distributed, volunteer systems are systems, which as the name suggests, make use of volunteered resources that are not all in the same physical location. These resources often have other primary uses, although they can be completely devoted to the distributed system if the owner desires.

There are two main models for implementing these distributed systems. The first model allows any user to submit calculations to be performed using the resources made available through the system. The second model applies all of the contributed resources to one single problem, which is understandably much more constraining and inflexible.

The most well known implementation of this former model is the Condor project[15], developed at the University of Wisconsin [8]. Condor allows the user to provide the required executable and any input files. It will then find a suitable compute resource, based on some requirements as specified by the user, and the status of the available machines registered with its system. Condor then sends the executable to this machine to be run, with any created output files returned to the submission machine upon completion. It should be noted that there are alternatives to the Condor system, but none are as heavily used.

An example Condor pool of this type is the Camgrid system[16]. Camgrid is a university-wide Condor pool, allowing users within Cambridge University to take advantage of spare

[15]http://www.cs.wisc.edu/condor
[16]http://www.escience.cam.ac.uk/projects/camgrid/env2.html

processing power within other departments, when their own resources cannot fulfill their requirements [15].

The Condor system has been heavily used within projects such as the *e*Minerals project. Many job submission tools are based around standard Condor tools, including nearly all of the tools discussed later in chapter 3. Condor has been important in abstracting the user away from the complications involved with systems like the Globus Toolkit[17] that is introduced in [29]. This reduces the amount of so called grid computing that the typical user is required to learn before they can make use of the available resources.

The most well known implementation of the second distributed system model is the Berkeley Open Infrastructure for Network Computing (BOINC) project[18] [2]. The BOINC software was implemented in order to allow for the perhaps best known volunteer computing system, 'SETI@home'[19] [3], which pre-dates *e*Science by several years. SETI@home persuaded members of the public to participate, and to volunteer their own home PCs, to search for extra-terrestrial intelligence, with the chance of being the person who found 'proof' of life on other planets. Each volunteered computer is given a different part of the night sky to be processed, as monitored by radio-telescopes, looking for signs of artificially created transmissions such as those emitted by our own television and radio signals.

The BOINC system is not used within the *e*Minerals project, because it is deemed to be too inflexible. However, it is a popular system within projects that are often in the public spotlight and to which the public are keen to volunteer their spare processing capacity. One such project is the 'Climate Prediction.net' project[20] [59], which is concerned with predicting how the climate will change in the coming years.

One of the main differences between these two models is that the code to be executed must be adapted to work with systems such as BOINC, and must have the area of study fully divided at the outset. Systems such as the Condor system do not require changes to the code and can be targeted at many different studies at once, making them much more general purpose.

Commodity clusters

Commodity clusters are computers comprising two types of machine. The first is a single

[17] http://www.globus.org
[18] http://boinc.berkeley.edu/
[19] http://setiathome.berkeley.edu
[20] http://www.climateprediction.net

'*head*' node, which provides the interface between the whole cluster and the outside world. The second type of node, multiple '*execute*' nodes, are the machines on which actual computation takes place. All access to the cluster is provided through the *head* node, from which the users are able to queue jobs for execution on the *execute* nodes. The cluster as a whole has a partly shared filesystem, meaning that all machines within the cluster are able to access all of the user's files. The whole cluster is then presented as one single machine to the user. Typically, these clusters are configured with scheduling systems such as the Portable Batch System (PBS)[21] software. PBS and other schedulers allow users to submit several simulations to be performed at once. These will then be queued by the scheduler and run as soon as spare processing capacity becomes available.

The 'commodity' part of the name is due to the fact that the components making up the cluster are standard, 'off the shelf' computers. In fact, these commodity clusters have very similar structures to the clusters used to provide HPC resources. The main difference being the technology used to provide the networking interconnects, which typically use standard ethernet, or similar technologies in commodity clusters, and more specialised, very expensive technologies in HPC clusters. This construction from commodity components gives these clusters the advantages that firstly, they are cheap to assemble, and secondly, it is easy to replace broken parts.

However, the low cost of the machines does not come without disadvantages. The primary disadvantage is that standard networking components are not as fast, and have higher latency, than those used within more expensive clusters. This means that commodity clusters can never hope to be as fast, or as powerful, as those combining purpose-built components.

Clusters of this type are traditionally used by logging into the *head* node, copying all required data to the machine, and then submitting the job to the local scheduling system. Once the job has finished executing, the user must then log back into the cluster and copy the produced files to somewhere that they can be analysed, and any interesting results collected.

Jobs submitted to clusters are able to run on single or multiple processors. Those running on single processors, known as 'serial jobs', run with performance similar to when running on the scientist's own desktop machine. Jobs that run on multiple processors, known as 'parallel jobs', are able to take advantage of the extra processing power and memory capacity afforded by the multiple machines being used. It should be noted that running

[21] http://www.openpbs.com

parallel jobs on these clusters is not as effective as when using purpose designed cluster hardware, due to the increased communication times involved.

Commodity clusters are often used for longer running and more computationally intensive simulations than the other HTC resources, due to the fact that the processing power is dedicated to this usage, unlike the distributed resources, where the processing capability is often shared with other tasks. However, the fact that commodity clusters are not as large, or as specialised as purpose designed clusters and HPC machines, means that larger jobs are often better suited to HPC, or specialist cluster resources.

The range of jobs that commodity clusters are suited to is very large, including many of the day-to-day calculations performed by the typical computational scientist. This means that commodity clusters are very popular and many computational science research projects have their own clusters, avoiding the need to use more expensive and less suitable machines.

Commodity clusters are so popular that there are several software systems available, which provide the clustering functionality required, once combined with the necessary hardware. Examples include Redhat Linux's 'Redhat Linux Cluster Suite'[22] and 'Open Source Cluster Application Resources' (OSCAR)[23] produced by the Open Cluster Group.

The computational section of *integrated grids* can obviously be compared fairly directly with some forms of HTC, with little difference for simple jobs. It can be argued that methods such as the BOINC system cannot be used within *integrated grids*, because they are too specialised and constraining in nature. However, resources such as Condor pools and clusters do make good contributions to, and can easily be integrated as a part of a computational grid. This allows for the years of development and research performed creating such systems to be utilised within the new paradigm of '*The Grid*', helping to reduce the negative effects related to the immaturity of the field. In fact, grid systems can themselves be considered to be HTC systems and are very good at dealing with the sorts of systems to which HTC is targeted. HTC systems have been integrated within the *e*Minerals grid systems, as documented later in chapter 2.

The development of three commodity clusters and their integration with the rest of the *e*Minerals infrastructure, which represents a prototype implementation of an *integrated grid*, is discussed in detail in chapter 2.

[22] http://www.redhat.com/software/rha/cluster/
[23] http://oscar.openclustergroup.org/

1.3.3 Traditional collaborative methods

Research projects have traditionally been constrained to collaboration between members of the same institution. However, as travel links and national / international communication using telephones and associated technologies have improved, such collaborations were able to happen more effectively in real time and over greater distances.

These collaborations still involved limited numbers of people and / or meetings, due to the costs involved in the travel between different project sites. Also, the lack of face-to-face communication has hindered such collaboration. This face-to-face communication is better than purely aural communication because communicating parties naturally notice visual cues when in conversation, which provides context to the collaborator's dialogue.

In more recent years, with the explosion of the Internet and related technologies, such as eMail and instant messaging, the possibility for realtime, or near-realtime, collaboration between remote sites has improved. However these technologies have not really addressed the need for face-to-face communication, and so are still fundamentally limiting to collaboration.

This has been addressed in recent years by the ability to perform multi-participant video conferences, using nothing more specialised than a standard desktop computer and a webcam. This is still very much an emerging technology, but has already enabled face-to-face communication between collaborators all over the world, enhancing the collaboration performed.

One example of a system tackling this multi-participant video conferencing is the Access Grid system. This has been used within the *e*Minerals project to hold regular meetings and even to present seminars, allowing all project members to attend, wherever in the country they are. This shows the power afforded by these collaborative tools.

1.3.4 Data storage and sharing prior to *e*Science

The traditional manner for use of the computing resources available to the typical computational scientist has been to perform the following steps. Firstly, they must copy any input files and the required executable to the remote computational resource. Secondly, the user must log into the resource and submit the executable to be run. Finally, they must wait for the job to finish, and once it has, copy back the files of interest to their local machine for analysis.

This approach, while successful, has several drawbacks. Firstly, the user may be tempted to leave their files on the remote machine, only retrieving files as and when required. This in turn leads to the user's files being segmented and dispersed, both logically and geographically, which is far from ideal when working within a large collaborative project. This leads to files at some point becoming lost, misplaced or even accidentally deleted.

Another problem that this mode of operation leads to is that even if the scientist does copy all files back to their own desktop machine, then this machine may not necessarily be backed up adequately. This leads to the risk of one hardware failure leading to loss of all of the scientist's important data.

Scientists also frequently need to access their data from sites other than their own office. Traditional storage does not always lend itself to remote access, especially when, as is often the case currently, machines are protected by firewalls or are connected to private networks, inaccessible from the outside world.

Sharing data with collaborators can also be difficult when using this scheme, because the creator of the data must first find the data and then somehow send them to the collaborator. This is not difficult soon after the data has been created, but once even a short amount of time has elapsed, directory structures often become confusing and difficult to navigate. This in turn leads, at worst, to loss of files, or at best, excessive amounts of time being spent looking for the data in question.

Data grids aim to tackle these drawbacks and to allow the handling of created data to scale with the number of simulations that the scientist is able to perform. As such they represent a fundamental change in the techniques that the computational scientist must employ when considering science on the scale enabled by grid computing.

The data grid approach to these problems is to use centrally managed systems, such as those described in section 1.2.2. These systems provide abstractions from the underlying physical filesystems, allowing all users to locate their data exactly as they would if all files were actually in one place, even though their files could be in any number of geographically dispersed locations. Indeed their data may even be in different types of storage systems, such as a combination of traditional filesystems and database systems.

Information about the stored files can also be captured and stored as metadata within grid environments. This makes it much easier for the scientist to find their data at a later date, especially when large amounts of similar data are created at the same time by simulation codes, which automatically create and collate data from large numbers of similar computations.

Several of the UK *e*Science funded projects have as their aim the provision of unified and intelligent data indexing and searching systems, where all data are stored in repositories, both centrally managed and those managed separately by collaborators. The indexing is provided through the use of a very restricted set of metadata, where the possible attributes that can be stored are indexed and have strictly defined meanings. One project that considers the data storage problem in this manner is the NDG project, mentioned previously.

The fact that when using a data grid all data are perceived to be in one location, leads to the trivialisation of the problems associated with physically locating data. File indexing and filesystem structure within that perceived location is, of course, still an issue.

This use of a common location for all files and data created by all collaborators means that the sharing of said data is also trivial. This is because the colleague to whom access is to be given, must simply be told where the file resides, and then be given suitable access permissions.

As the above discussion implies, traditional data storage and sharing cannot really be directly compared to data grids. However, the traditional methods are essential underpinnings to data grid systems, because the files and data must still be physically stored somewhere, even if the user does not know where, or how.

The integration of data storage and sharing systems with the data creation and project collaboration systems, helps lead to the power and importance of *integrated grids*. This is because the archiving and sharing of data created during the scientist's research is very important in furthering research as a whole.

1.4 The *e*Minerals project

The work in this thesis was driven by the *e*Minerals project, or to give the full title: 'the environment from the molecular level' project. *e*Minerals is an *e*Science testbed project funded by the Natural Environment Research Council (NERC)[24]. Its research is concerned with the furthering of grid infrastructure. It also considers the enhancement of environmental research through the application of this infrastructure.

[24]http://www.nerc.ac.uk

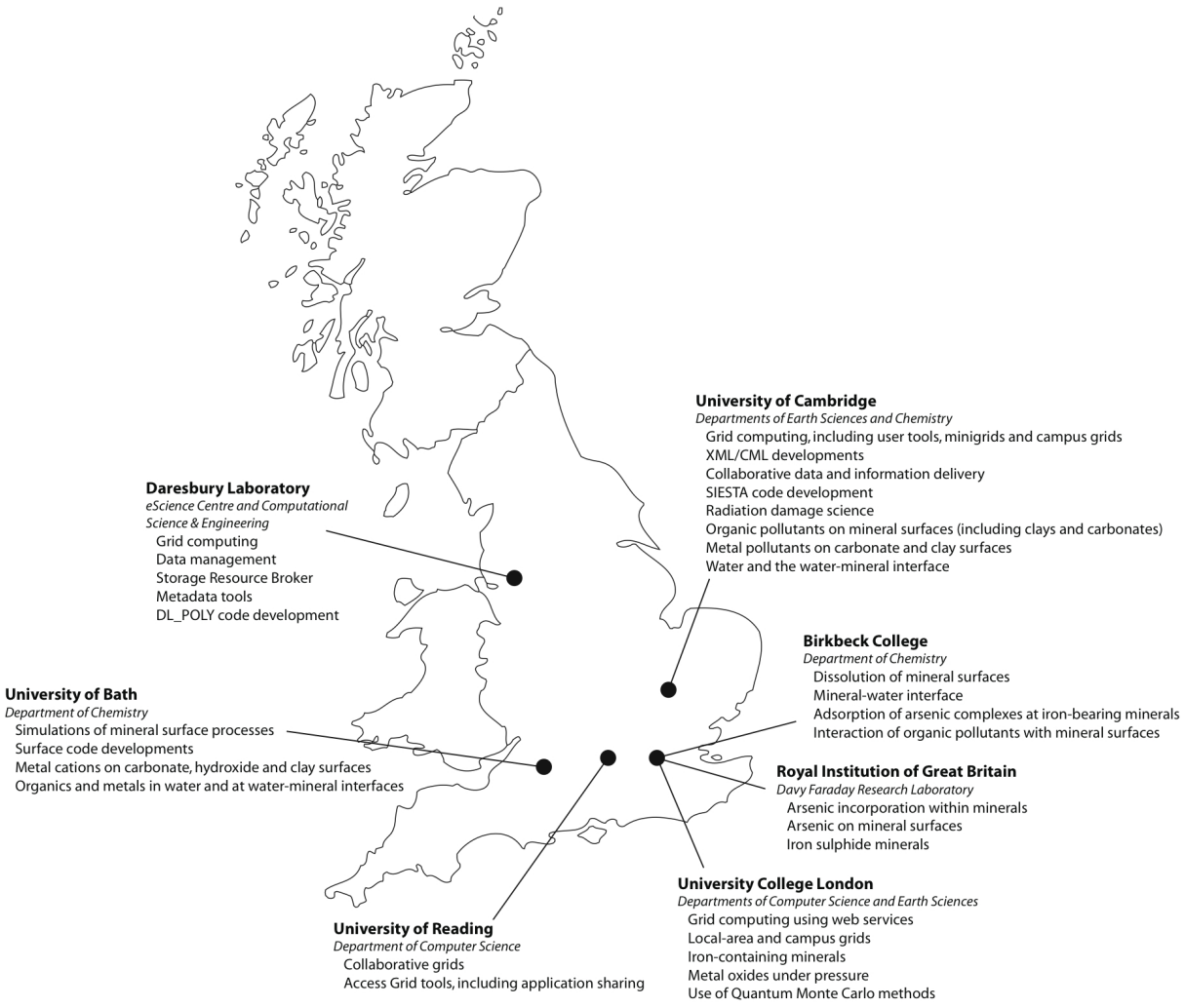

Figure 1.2: A map of the UK, showing the different *e*Minerals project member institutions and their respective areas of expertise.

*e*Minerals is a multi-institution project involving 30 members, ranging from PhD students, through postdoctoral researchers to university lecturers and professors. These members are located in nine departments, at seven different academic and research institutions. The areas of expertise held by each of these project members can be seen in figure 1.2, which lists the different institutions, departments and areas of interest involved with the project.

The research areas of these project members are understandably as diverse as their location within the UK. The project research involves many different relevant environmental topics, in addition to simulation code, and grid system development.

The scientific topics include radioactive waste containment media, and the modelling

of water and surfaces under aqueous conditions. In addition the research includes the modelling of pollutant molecules such as sulphides, halogens, metals and silicates, and finally the modelling of these molecules' adsorption onto mineral surfaces.

Simulation code development has included continued development of the Chemical Markup Language (CML) [45] and [77], and its integration into the project simulation codes. This development and integration is discussed in detail in appendices A and B.

Last, but not least, the grid technology research has included the development of automated workflow processes, involving combinations of the project simulation codes. In addition, simulation job creation and brokering across the grid, including metadata capture and processing, and visualisation and delivery of scientific data to the scientist have been considered. Final grid research topics are virtual organisation operation and inter-institution collaboration, and the usability of grid systems.

Due to the diverse interests of the researchers involved within the project and their distribution throughout the UK, a very important part of the project has been the consideration of collaboration methods, and the development and use of tools to improve and enable this collaboration. Without this focus on collaboration, the *e*Minerals project would not have been able to take advantage of the different specialisations of each of the project members.

The different aspects of the research performed by the *e*Minerals project will now be discussed, with the interests broken down into the scientific (section 1.4.1), grid technology (section 1.4.2), and collaboration areas (section 1.4.3).

1.4.1 Scientific challenges

The *e*Minerals *science cube*

A large part of the *e*Minerals science challenge is concerned with the adsorption of pollutants of varying chemical makeup onto various naturally occurring surfaces. These different systems are considered with varying levels of scientific theory, enabling the scientists to consider different aspects of the problem that could not all be considered by one level of theory alone. This area of interest, with its three axes is shown graphically in figure 1.3. This representation has become known within the *e*Minerals project as the '*e*Minerals *science cube*'.

The pollutant adsorption problem is important, because this affects how these pollutants

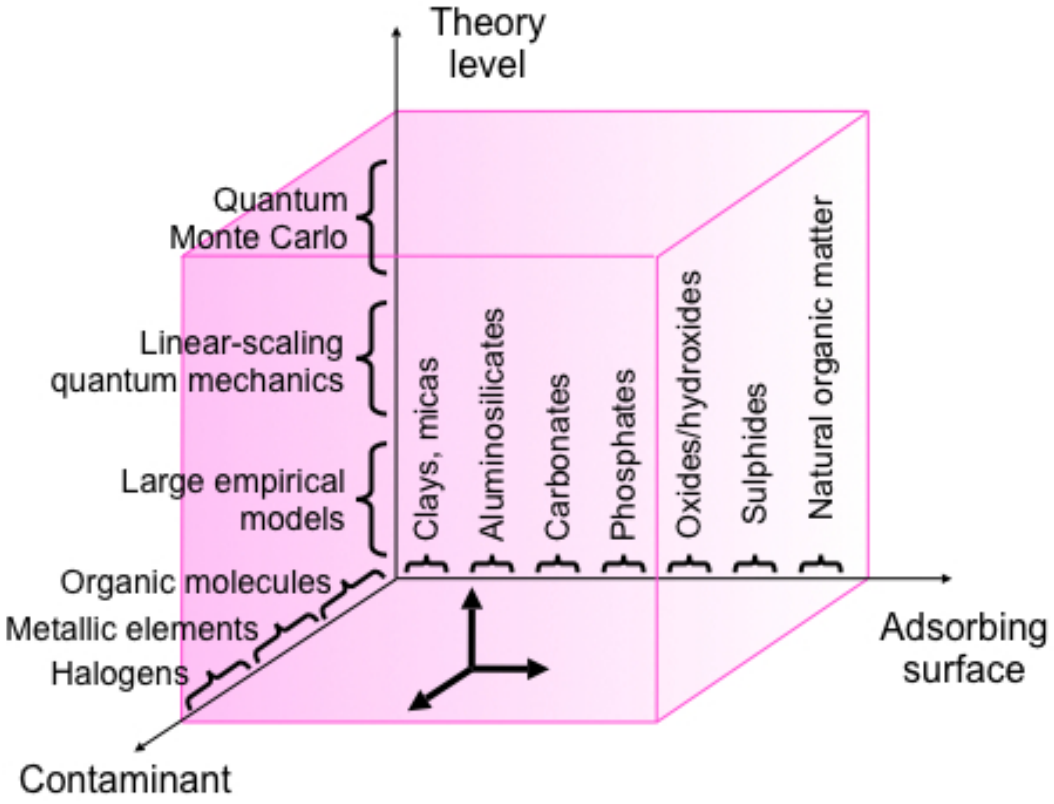

Figure 1.3: The *e*Minerals *science cube*. A 3D representation of the different topics and level of theory considered within the *e*Minerals project.

will be transported within the environment and transferred into the human food chain. These pollutants are often long-lived, meaning that they cannot be left to decay on their own; rather they must be removed from the environment once released. These reasons mean that it is very important that factors making some of these pollutants be adsorbed into soils more than others is understood. This understanding will make it possible to more accurately predict the dangers of accidental chemical spills, or to quantify the resulting outcomes of accidents. One such accident for which such predictions could be useful is the fire that occurred at the fuel processing plant in Buncefield, England in December 2005[25].

It is also important to realise that pollutants do not stay at the point where they initially escape into the environment. This is due to the fact that pollutants are able to be carried by water flow, which occurs through all but the most arid of soils. This can transport pollutants quickly and perhaps undetected into areas where the environment could be more easily adversely affected. This sort of transportation is often the cause of pollutants

[25]http://news.bbc.co.uk/1/hi/uk/4517962.stm

such as pesticides from farms, making their way into nearby streams and rivers, poisoning the wildlife found in those waterways.

The three axes of the *science cube* and their respective importance will now be discussed. This will show that there is a requirement for many different types of computational resource within the *e*Minerals project. In addition, this requirement influenced the development of the *integrated grid* idea, and its first implementation, the *e*Minerals *minigrid*, which is discussed in detail in chapter 2.

Adsorption surface and contaminant species

Wide ranging expertise is held within the *e*Minerals project related to the area of contaminants, especially with regard to their adsorption onto many different important materials, typically those found within soils throughout the world. These different materials and contaminants do not all behave in the same manner due to the different elements involved in their makeup. This means that no single project member can possibly know everything about all such materials found in soils throughout the world.

Over the past few decades many different pollutants have been released into the environment. Amongst these pollutants, three families that are studied heavily within the *e*Minerals project are the polychlorinated biphenyls (PCBs, $C_{12}H_xCl_{12-x}$), polychlorinated dibenzo-p-dioxins (PCDDs, $C_{12}O_2H_xCl_{8-x}$) and polychlorinated dibenzo-furans (PCDFs, $C_{12}OH_xCl_{8-x}$). These chlorinated aromatics are both long-lived and toxic, often feeding back into the human food chain. Other pollutants studied include the heavy metals, such as arsenic, which are often released due to mining and drilling, and are well known for their harmful effects. The understanding of the transport of pollutants such as these is vital for the development of future remediation strategies.

Each of the different materials shown on the adsorption surface axis in figure 1.3 are considered within the *e*Minerals project, and are all found in soils throughout the world. The material considered most in this thesis is calcite ($CaCO_3$), which is a carbonate mineral. Chapter 5 considers the adsorption of the PCDD family of pollutants onto the surface of calcite.

In summary, it should be clear from the above discussion that such a combination of different mineral surfaces and contaminant species could not be covered using more traditional research methods. In addition, future work covering such wide-ranging topics will require collaborations as large and diverse as that demonstrated by the *e*Minerals project.

Theory level

Within the computational chemistry community there are many different methods that can be used to model structures and interactions at the molecular level. Each method has its own strengths and weaknesses when compared directly with the others. For example, some methods give more accurate results than others, whilst the less accurate methods will be faster when implemented, meaning that the loss of accuracy is often found to be acceptable.

The different methods used within the *e*Minerals project, and which will be described below, are:

- Empirical models
- Quantum mechanics
- Quantum Monte Carlo

Empirical models, also known as 'Empirical Potential models' or 'potentials' are, as the first two names imply, methods which are in some way enabled and parameterised by empirical evidence or results. The calculations are actually fitted to results from physical experiments, or from ab-initio calculations[26]. This fitting ensures that the initial structure of the system being modelled is correct with respect to atom positioning, etc. During the actual simulation atoms are treated, in effect, as spheres with mathematical relationships expressing how they attract and repel one another.

Empirical models apply a very simple theory, which means that they can be applied to systems of any size, from the very small right through to the very large. An example empirical model simulation code is DL_POLY, mentioned previously, which has been shown to scale to models comprising several million atoms.

Quantum mechanics, also known as 'Electronic Structure Methods', are a group of similar methods that calculate the system energy with respect to either a wavefunction or the density of electrons surrounding the nuclei of the atoms in the system. A wavefunction is a mathematical formula, which due to wave particle duality, can represent the position and movement of particles in the system.

[26] ab-initio calculations are calculations performed from first principles.

Quantum mechanical methods implement a more complicated theory than empirical models, meaning that they cannot be used to model systems on the same scale. However, it does lead to more accurate results and can still be applied to a large range of model sizes. Smaller models can be simulated using standard commodity resources, while larger systems require the use of HPC resources.

Density Functional Theory (DFT), is the name given to one implementation of quantum mechanics. Two DFT implementations are used within the *e*Minerals project: SIESTA[27] [58] and CASTEP[28] [56]. These two codes implement DFT in different ways, CASTEP represents the wave function as a wave, whilst SIESTA uses atomic orbitals instead. These differences mean that SIESTA scales better with the number of atoms being modelled, which results in it being used for larger systems. CASTEP on the other hand, is more accurate, and so is used for smaller systems, where accuracy is more important.

Quantum Monte Carlo (QMC), is the name given to an emerging technique, which seeks to solve the same problem as DFT and other techniques, whilst avoiding most of the approximations that are required. One implementation of the QMC methods is CASINO[29] [46], which is used within the *e*Minerals project to simulate iron bearing minerals.

QMC is very computationally challenging, because of the complicated theory it implements. This means that it can currently only be applied to very small models, and even then it requires the use of HPC resources.

1.4.2 Computational challenges

As mentioned previously, the *e*Minerals project began at a time when there were no widely available grid resources. In fact, it was not even clear at the time how effectively grid computing could be applied to computational chemistry challenges at all.

Building a grid system is much more than just configuring machines in a traditional manner, installing relevant grid tools on them. Rather, the separate resources must be linked and integrated together is a usable manner, allowing the user to make use of all resources as if they were one cohesive unit. In addition, the integration of many different types

[27] http://www.uam.es/siesta/
[28] http://www.tcm.phy.cam.ac.uk/castep/index.html
[29] http://www.tcm.phy.cam.ac.uk/~mdt26/casino2.html

of computational resource sets grid systems apart from large scale traditional resources, such as HPC, or large clusters.

The computational simulation scientists within the *e*Minerals project are definitely more than typically competent computer users, as one would expect when considering the sort of simulations that they perform on a daily basis. The question therefore arose as to how the scientists should be given access to computing resources within a grid infrastructure. The answer to this question came in the implementation of the *e*Minerals *integrated grid*, which was created as part of the research documented within this thesis. This work is discussed in much more detail in chapter 2.

1.4.3 Collaborative challenges

As has already been mentioned, the *e*Minerals project is a fairly large project, with many collaborators working from geographically dispersed locations. Collaboration on this scale and of this geographic dispersal would not have been possible to manage until relatively recently. The advent of cheap, high quality video conferencing applications and associated tools have allowed such wide ranging collaborations to be practicable.

Even with the tools made available to virtual organisations aiding collaboration, keeping a project the size of the *e*Minerals project motivated and working towards a common goal is a difficult process. It is however, one which has been achieved within the *e*Minerals project, where a main driver towards collaboration has been the desire to study large scientific topics. These topics require expertise from multiple people and are too large for a single researcher, or even a whole institution, without a very large injection of funding. This means that scientists must collaborate between institutions in order to achieve their research goals.

The *e*Minerals project has taken great care to ensure that the collaboration works as one large, cohesive unit. Some of the project experiences and outcomes concerning this cooperation can be seen in [17], [23] and [24].

1.5 Applying the grid to scientific research

As has been made clear in previous sections, grid computing is very different from traditional computational chemistry methods. These differences, which have been examined in detail in section 1.3, are very important, meaning that the scientist cannot simply

stop using traditional methods one day and start with grid methods the next. Rather, the change must be gradual, with the scientist needing to adjust their mindset slightly to work with an *integrated grid*. This change, coupled with the scientist's experience once they move to *integrated grids*, can result in an epiphany of sorts, with the scientist wondering why they did not change sooner.

It is important for the scientist to realise quickly that when dealing with *integrated grids*, they will find themselves creating much more data than was previously possible, and in a much shorter time. The managing of this date deluge is not possible using traditional data management techniques, so the scientist must embrace the data grid and the techniques associated with it.

The scientist must move from working with files stored on their local hard disk, or on a collection of different machines' hard disks, to a system where all of their data are stored on a centrally managed system. This centrally managed system requires data to be retrieved from it, before any simulations can be run. Data must also be uploaded at the end of a simulation before any further analysis can commence.

Forcing the scientist to transfer all data files via a data grid gives two advantages. Firstly, users are able to easily share their data with collaborators. Secondly, every file a user ever uses as input for a simulation, or that is created by the simulation itself, is stored and archived for later access. These advantages can be used as a good basis for convincing the scientist that their working methods should change, and that this change will result in better accountability for their work, and reduced work at a later date.

Another area that requires the scientist to change their working practices is the collection and archiving of metadata related to all of their research. This is a completely new concept to most scientists, and as such, scientists often require some convincing to adopt metadata capture methods.

One way to help convince the scientist to use metadata facilities, is to provide as much automatic metadata capture and processing as possible. This is the method taken within my grid submission tools and will be discussed in detail in chapter 3.

The final part of the change from traditional methods to *integrated grid* usage relates to the change in collaboration methods. The change here is not as drastic as that experienced with the move to data grid usage, and is much more obviously beneficial to the scientist on their first being presented with the change.

These collaboration changes are also closely related to the data changes detailed above,

as it is easier to share data between collaborators when the data are all in one place, as is advocated by the data grid approach. Relevant metadata collection at data creation time also assists in this collaboration. This metadata collection means that the collaborator can simply search through the metadata for work relating to their interest, and then if they have been given appropriate access permissions, download and look at the data without even having to bother the originating collaborator.

This can be a common scenario within projects such as the *e*Minerals project, where several project members at different institutions collaborate on performing the simulations, but where only one collaborator is physically able to perform any particular simulation run.

1.6 An outline for this thesis

This introduction has discussed the history, background and current state of grid computing research, and the challenges experienced by the *e*Minerals project when I started my research. This thesis is primarily concerned with the work that I have carried out to tackle these challenges and to ensure that the vision of the *integrated grid* became a reality.

Chapter 2 of this thesis will discuss the requirement for, and development of, the *e*Minerals *minigrid*, the first implementation of an *integrated grid*, defined earlier in this chapter. This development allowed for the testing of grid systems and their interaction with one another. In addition, it allowed the integration of each of the different grid types as discussed above.

This meant that tools performing job submission and management were required in order to take advantage of the available resources. My work in developing and refining these tools, enabling the user to make use of *integrated grid* systems without being forced to revert to traditional usage methods, is discussed in chapter 3. These tools make it possible for the scientist to perform very large numbers of simulations in a short amount of time and with the minimum of effort.

Once the user is able to make use of these tools and the *integrated grid* infrastructure, the challenge faced by the user then develops from creating and submitting jobs on this scale, to being able to deal with the deluge of data created by these jobs. Chapter 4 will discuss my work in simplifying and improving the access scientists have to their data when created on this scale. This includes data visualisation, which aims to assist with

the analysis of created data, reducing the work that the user is required to perform, and allowing them to consider more data.

As a part of implementing the *integrated grid* idea, and developing the tools to use these systems, it has been important that I apply these systems and tools myself, ensuring that they are usable and provide the sort of functionality required by the user. To this end I have applied these tools to the study of several systems. The main study investigated the adsorption of the PCDD family of pollutant molecules onto the surface of the calcite mineral. This is an important environmental issue because calcite comprises a large part of the world's soil makeup. Chapter 5 discusses my experiences of applying my tools to these studies, considering how they can be useful for real scientific research. In addition, other research topics to which the tools have been applied are discussed, showing how the tools are used generally within the *e*Minerals project.

Finally, chapter 6 will collate and discuss my conclusions related to the grid, and in particular *integrated grids* and their usage. My experiences and efforts will be discussed in relation to the current status of research into grid computing and its application to environmental science.

Chapter 2

Creating the *e*Minerals *minigrid* –The first implementation of an *integrated grid.*

This chapter discusses my efforts in constructing a testbed system, conforming to the specification of an *integrated grid*, as outlined previously in section 1.2.1. This testbed system is called the '*e*Minerals *minigrid*'. Issues associated with user access and usability are detailed later in chapter 3, where tools for job submission and management are discussed.

Parts of the text of this chapter have also been adapted and published in the IEEE Journal of Computing in Science and Engineering [12].

2.1 *e*Minerals background

At the beginning of the *e*Minerals project there were no computational resources attached to the project at all. Project member institutions did of course have their own resources, which had been purchased and used for previous research projects. However, no resources were explicitly provided for the project, with the expectation that the resources used would be a combination of these existing resources with those purchased throughout the duration of the project.

Before I started on *e*Minerals, initial work on the configuration of some grid resources was started. This involved the parallel creation of two separate Condor pools, one in the

Department of Earth Sciences, at the University of Cambridge and the other at UCL. This latter pool eventually developed into a campus-wide pool and achieved the status of the largest academic Condor pool within the UK, combining over 1000 processors. These pools were developed independently, but not in isolation, so that experiences from each pool could be fed into the development of the other, leading to better systems than a more close-knit design process would have achieved.

Access to these Condor pools was allowed through the installation of Globus on a suitable submit machine on each pool. Users could then log into these machines using *gsissh*, which is a version of the well known '*ssh*' tool that supports authentication using X.509 certificates[1]. This means that the user need not know, or even have, a password on the submit machine. Once users had logged into the chosen submit machine, they could submit their simulations using the standard Condor tools.

As an alternative to the user logging in and using the Condor tools to submit their jobs, they could also use other Globus tools. The use of these tools allows the user to submit jobs directly to the Condor pools from their own submission machine, without needing to log into the remote pool's submit machine. This method did not become widely used within the *e*Minerals project, because the Globus tools have a very complicated, non-intuitive command line syntax. In fact, the *e*Minerals users found these commands prohibitively hard to learn and use.

A comparison of the difficulty involved when submitting jobs directly, with being allowed to login to the computational resources, led to the users choosing the latter option. This led to research being performed using these initial resources in a manner very similar to research prior to *e*Science and grid computing. This is obviously not ideal for a project concerned with the research of grid technologies. As such, when I started with *e*Minerals, my work concerned the creation of a more integrated system, usable without requiring the user to login to different submit machines depending on where they wished to submit.

At around the time of my starting research within the *e*Minerals project, the idea of an *integrated grid* was suggested, as defined previously in section 1.2.3. To reiterate, an *integrated grid* is a grid system, which allows for uniform access to all computational resources, with associated functionality for the handling of data and metadata, and related

[1]X.509 certificates allow the user's identity to be certified by the certificate issuer. They allow the user to connect to other systems, which trust the issuer of the certificate, without the use of passwords for authentication. The machine from which the user is connecting exchanges encrypted messages with the machine to which the user wishes to connect. These messages allow both machines to prove that the other is 'who' it claims to be. Once this has been proven permission can be given to the machine for login access, job submission, etc.

collaborative tools. This chapter describes my work in creating the first implementation of the *integrated grid* idea, the '*e*Minerals *minigrid*'.

2.2 The *e*Minerals *minigrid*

2.2.1 The *minigrid* as an integrated system

The *e*Minerals *minigrid*, as mentioned previously, is the name that has been given to the core integrated compute, data, and collaborative resources created and used by the *e*Minerals project. Initially the *minigrid* was created as a test system to measure what was achievable with available technologies, and to be an initial implementation of a network of grid resources.

The created network was called a *minigrid* for two reasons. The first reason was the fact that the network of resources is smaller than that generally implied by the term 'grid', whilst still combining each of the key features of a larger system. The second reason was that it was developed in such a way that it allowed for connections with other similar grid systems. These connected *minigrids* could then be considered a 'grid'. It is worth noting that a *minigrid* is not the same as a 'lightweight grid'.

The *minigrid* was designed to be heterogeneous in terms of the resources it combined, using many different computational architectures and software versions. This was in an effort to develop a testbed that could be effectively compared with the future aims of production level grid systems, and that could be applied to the different needs of the very different simulation codes used by the *e*Minerals scientists. These production level grids will be very large, with components distributed nationally, if not internationally. Even though many projects have attempted to enforce homogeneity, it can be argued that any resource on such a scale will not be able to practically ensure complete homogeneity, on either hardware or software levels. As such, for the comparison of a system such as the *e*Minerals *minigrid* with future production grids, the *minigrid* was required to be completely heterogeneous in composition.

In addition to the necessary provision of computational resources for the project scientists, the *e*Minerals project is concerned with performing research in the field of grid computing. The *minigrid* as a whole was implemented as part of this research into grid computing methods, and to allow the further research of grid job submission and management systems.

The integration of the many different computational and data resources into a single grid infrastructure means that they can each be accessed in a consistent manner. The following sections discuss the make-up of the *minigrid*, as well as the configuration, monitoring, and management of the different resources and how they have been integrated.

2.2.2 The *minigrid* computational grid component

The most diverse section of the *e*Minerals *minigrid* is the combination of computational resources, where several different computational architectures, operating systems and utilisation methods have been integrated. The *minigrid* was developed to be flexible, integrating many of the different computational systems described previously in section 1.3.2.

Initially, the *minigrid* simply combined the two Condor pools at Cambridge University and UCL, both of which were made accessible via the Globus software installed on two 'gatekeeper' machines, one at each site. These gatekeeper machines provide access to the compute resources by having both Globus and Condor installed on them. They act as translators, converting from commands sent to the Globus software, into commands which can be understood by the Condor software. This means that the user need not learn each of the different Condor commands, instead they can use the same Globus commands to access each resource.

This simplification, allowing the user to use one single set of commands to access the two resources, was not really an advantage at this stage, because the user could simply use Condor commands on each of the resources. However, it is much more important once resources of different types are combined. The use of Globus also means that the user need not log into each of the gatekeeper machines individually, they can simply run jobs by performing the necessary commands on a correctly configured submit machine. This submit machine was required to have the Globus client software installed on it, and could be the user's own desktop computer, if they use some form of Linux, Unix or Mac OS X. For users that could not install Globus on their desktop, submit machines were provided as part of the *minigrid*, allowing the users to log into them and submit their jobs to each of the resources.

While the combination of these two Condor pools was a good start, it did not represent the full spectrum of computational resources required within a grid environment. This is because, while Condor pools are great for large numbers of small calculations, they are not suitable for larger, longer running simulations due to the shared nature of their machines.

As such, three Linux based commodity clusters were built, configured and integrated with the other machines. These clusters are known within the *e*Minerals project as the '*Lakes*'. The design and configuration of the '*Lakes*' is described in section 2.4.

In addition to the *Lake* clusters, another existing commodity cluster was integrated with the *minigrid*. This cluster, called *Pond*, has identical construction to the *Lake* clusters, but combines more, lower powered processors, and relies on lower bandwidth networking interconnects.

Both the *Lake* and *Pond* clusters were built from commodity components, meaning that they were relatively inexpensive to build and maintain. However, this means that they are still not suitable for some of the larger simulations performed within the *e*Minerals project, which require more powerful resources that can be used in parallel. These larger jobs are often not large enough to allow the efficient use of HPC resources and as such require the provision of purpose designed cluster hardware on a scale much smaller than HPC resources provide.

To address this need for more intermediate level resources, I built and configured an Apple XServe cluster called 'Lagoon'. This cluster consists of purpose designed hardware, combining more powerful processors and faster networking interconnects, to provide higher performance than can be achieved by the *Lake* clusters when applied to the correct type of simulation.

Lagoon was initially configured in a very similar manner to the *Lakes*, with a Sun Grid Engine (SGE) scheduler[2], which is also accessible via the Globus software. However, due to some hardware problems, followed by difficulties with the operating system configuration, the decision was made to remove the SGE system and replace it with Condor. This use of Condor meant that the configuration of the operating system could be much simplified, removing the need for a shared filesystem across the whole machine, which cannot be configured as easily under Mac OS X as it can under Linux.

In addition to each of the above resources, an IBM pSeries cluster was integrated with the *minigrid*, called 'Acet-blue'. This cluster represents a small scale HPC resource and is owned by one of the *e*Minerals project member institutions (Reading University). *Acet-blue* allows members of the *e*Minerals project to perform even larger simulations than those possible using *Lagoon* and the *Lakes*. *Acet-blue* was already configured for use by local users. My contribution was to configure the Globus software, allowing the cluster to be used in the same manner as each of the other *minigrid* compute resources, from a

[2]http://www.sun.com/software/gridware/

suitably configured submit machine.

Access has also been provided to the *Camgrid* system, mentioned earlier, by providing a suitable gatekeeper machine. This resource can be used to perform the longer running jobs that have traditionally been reserved for resources such as the *Lake* clusters. This is because many of the machines within *Camgrid* are very powerful commodity machines, that are dedicated to *Camgrid* usage unlike traditional Condor systems, which share resources with other primary uses, scavenging spare cycles when they are not being used by others.

The different hardware and software systems combined within the *minigrid* have been integrated through the installation of Globus gatekeepers on each system. This allows for submission to any of these different resources through the use of the Globus tools from a single submission machine, which as described above, can be the user's own desktop computer.

Each of the above machines are in themselves not anything new, and alone, are not really grid resources. It is only the way that they are configured and integrated with one another that makes them into grid resources. The process of configuration and management of resources designed to be used as part of a grid can be very different from the equivalent processes for non grid-enabled computational resources. The actual configuration applied to the *Lake* clusters is described in much greater detail in section 2.4.

2.2.3 The *minigrid* data grid component

As has already been explained, the provision of ever increasing numbers of computational resources does not, on its own, lead to improvements in productivity or research quality. Managing the data created by the simulations on the scale allowed by these extra computational resources is equally important, and has been tackled by implementing a data grid as part of the *minigrid*.

The Globus software allows for the transferring of data files between two appropriately configured machines. However, these tools often require the user to know the names and exact location of all files that should be transferred. These details may not be known when for example, a typical simulation can produce seventy or more data files, with often obscure names, decided upon by the particular simulation code. In addition, the location of these files may not be intuitive when using the default scratch directories created by the Globus software's submission process.

Simply transferring the files back to the user's own desktop computer does not lead to the benefits associated with data grids, as described previously. The users will simply have the same difficulties as they had prior to the advent of grid computing, but on a larger scale, due to the number of individual simulations enabled when using the computational resources made available by the grid. These difficulties include the loss of files within complex filesystem hierarchies, and the temptation to leave the data on the remote computational resource, rather than spend time identifying and then copying them back.

Before I started within the *e*Minerals project, expertise was already held in the administration and use of the Storage Resource Broker (SRB)[3] [7]. This experience led to the decision within the *e*Minerals project to use the SRB for all of our data management requirements.

The SRB system consists of a central application server and cataloguing database, known as the 'Meta information CATalog (MCAT) server', and a number of vaults. These vaults are used to store the users' actual data, and are simply large, appropriately secured disks with the necessary software installed. Each of these different vaults are hidden from the user, with all data within the SRB being presented as a single logical filesystem. This means that the vaults can be distributed geographically without the user needing to remember where each of their files is stored, as the SRB presents a single virtual location from which all data can be found.

The presentation of a virtual location to the user is achieved by keeping track of various pieces of information for every file stored. This information includes the name and both the file's physical location and its location within the logical filesystem presented by the SRB. The MCAT server collates all of this information, allowing the user to retrieve their files, even though the user does not actually know where they are physically stored. The user simply needs to request the file by its logical name and location. This is then mapped to the physical filename by the SRB software and the file retrieved, returning it to the user.

The design of the SRB software allows for the use of any number of vaults, installed on different architectures, and using different filesystems. As such, a relatively small number of vaults can be installed and used initially. Then, once the users start to fill these vaults with their data, additional vaults can be commissioned. This allows the whole system to scale with the requirements of the users. Using the system in this manner also allows the project to take advantage of the fact that hard disk drives are constantly improving

[3] http://www.sdsc.edu/srb/index.php/Main_Page

in speed and capacity, with new vaults being larger than preceding vaults, but at lower cost.

To make use of the SRB, and to take advantage of its ability to scale with usage, I configured each of the different computational resources described above to include an SRB vault, with the exception of the *Acet-blue* resource, where local administrators installed their own vault. Disk space was allocated to each vault and the relevant software was installed, allowing all project members to use the SRB.

The decision to use the computational resources to house the SRB vaults was made in order to ensure that the complexity of the different firewall holes required between *minigrid* resources was kept to a minimum. In addition, the SRB vaults could take advantage of the fact that the gatekeepers of each of the computational resources are relatively underused, since they do not run simulations themselves. The spare processing capacity can therefore be dedicated to operating the SRB vault, although in fact, the SRB software does not require a large amount of processing power.

To access their data, the user must use one of several different client tools. These tools range from command line tools similar to those found on any Linux or Unix based system, right through to web browser based tools, such as *TobysSRB*, as developed within the *e*Minerals project and discussed in [79].

The command line tools, the 'Scommands', are a set of commands combining the functionality of the standard Unix filesystem navigation commands, and the standard FTP commands. The Scommands are used extensively within the *e*Minerals project, especially within the job submission tools described in chapter 3. They have been named such that the user can apply their knowledge of these standard commands, prefixing them with an 'S'. The commands include '`Scd`', which allows the user to change logical directory within the SRB, '`Sls`', which lists the contents of the current directory. In addition '`Sput`' and '`Sget`' are provided, allowing the user to transfer files to and from the SRB.

As has already been mentioned, once the data has been stored within the SRB, or any other data grid system, it can easily be shared with collaborators. As such, the SRB can also be considered to be part of the collaborative section of the *minigrid*, which again shows how the distinction between the different grid types is often blurred.

2.2.4 The *minigrid* collaborative grid component

The third section of the *e*Minerals *minigrid* is the development and use of advanced collaboration systems. The majority of this development and research has been performed by other members of the project. However, it is important to mention here because the systems used comprise a key part of the *minigrid* and of the *integrated grid* idea as a whole.

My work on collaborative tools has included development of visualisation tools and is discussed in detail in chapter 4.

2.3 Why was an implementation of the *integrated grid* required?

As has already been mentioned, when the *e*Minerals project first began, there was no grid on the scale often considered a prerequisite for grid research. In addition to this lack of a unifying grid system, there were not locally, or even nationally organised grids. As such, it was necessary for the *e*Minerals project to set up its own grid, in order to be able to research methods for configuring grids, and to create associated tools, etc.

Since the inception of the *e*Minerals project, national and even international grids have been created, such as the 'National Grid Service' (NGS)[4] and Eurogrid[5] [41]. However, these systems did not become operational until a large proportion of the *e*Minerals project had elapsed. This meant that they could not be relied upon to be the main source of the computational resources required by the *e*Minerals project.

It is important to note that to perform the grid research required within the *e*Minerals project, it is necessary to have full control over the resources being tested. A normal user cannot install the different grid tools and software, which require administrative access to the resources. This sort of access is not generally allowed, unless the user in question owns or controls the resource. Therefore, for the grid research to be performed, a certain amount of test resources, over which the tester has administrative control, are required. This was the reason to construct the *e*Minerals *minigrid*.

[4] http://www.ngs.ac.uk
[5] http://www.eurogrid.org

A part of the remit of the research performed by the *e*Minerals project was the testing of how grid techniques could be applied by intermediate to large scale collaborations. This research had the aim of evaluating whether it would be possible to apply these techniques to even larger scale collaborations. As such, the *minigrid* was designed and developed, from the outset, to be scalable to large numbers of resources, and to be as flexible as possible. This meant that it was important for externally owned and administered computational resources to be integrated with the *minigrid* systems, in what appears to the scientist to be a completely seamless fashion.

The *minigrid* was developed, such that any integration of external resources should be simple and seamless, whatever the actual resource involved. Single, parallel computers, commodity clusters, Condor pools of any size, or even entire grid systems, should all be usable in exactly the same manner. This integration has been achieved within the *e*Minerals project and is discussed in more detail in section 2.7 and chapter 3, where the cases of integration with the NGS and NW-Grid[6] systems will be described.

2.4 Configuring the *Lake* clusters – A case study in grid resource configuration

2.4.1 An introduction to grid cluster management

The following sections form a case study, describing the construction, configuration, management and usage of the three *Lake* clusters introduced earlier. Issues such as their installation and configuration, and how the clusters are monitored and managed within the *e*Minerals *minigrid* will be discussed.

The techniques employed to ensure data stored within the SRB data grid system are secure and safe from hardware and other failures, will also be described. In addition, there will be some discussion of the methods used to access the clusters, although the majority of my work related to computational resource usage will be described in chapter 3.

[6] http://www.nw-grid.ac.uk

2.4.2 Cluster structure and operation

Cluster structure and basic installation

Each of the three *Lake* clusters is assembled from standard commodity components. They consist of sixteen compute nodes, each with a single 2.8 GHz Pentium IV processor, 2 GB of RAM and a 120 GB system hard drive. The nodes are connected together using gigabit Ethernet and are controlled by a separate head node of near-identical specifications. The only difference in the head node's specifications is that it contains a RAID array providing 720 GB of data storage in addition to the main system hard drive. This storage is used to contain an SRB vault, which can be accessed from any of the *minigrid* resources, using the SRB tools.

The *Lakes* are not aimed at parallel simulations; rather they are aimed at long running serial jobs. These jobs cannot be executed using some of the other *minigrid* HTC resources, which are shared with other uses, and so cannot be applied to such long running tasks without risking that the jobs will be stopped before completion. However, the clusters have been used for some small parallel jobs, where a minimum of inter-node communications were required. As such, the main requirement for the networking hardware, is that it is optimised for bulk data transfer, which requires high bandwidth rather than ultra-low latency. Gigabit Ethernet was used to provide these communications, even though it has a slight latency penalty as compared with 100 Mbit Ethernet. This penalty was minimised by careful optimisation of the Linux kernel TCP/IP configuration. An average latency of approximately 100 μs was achieved using 64-byte packets, which is perfectly acceptable for the high-throughput style calculations to which the clusters are applied. Users were then encouraged to use other resources to provide for their high-performance requirements.

Each of the clusters run Mandrake Linux 9.2[7] and Oscar 2.3.1. Mandrake provides the core operating systems, while Oscar[8] provides all of the necessary cluster installation tools. Oscar also provides systems for all basic operations, such as job scheduling and basic monitoring, via PBS and the Maui job scheduler[9].

Most of the code used by the *e*Minerals scientists is written in Fortran. Some of these simulation codes also use Message Passing Interface (MPI) libraries, even when running

[7]Mandrake Linux has been renamed to Mandriva Linux since the installation of the clusters and can now be found at http://www.mandriva.com/en/community/

[8]http://oscar.sourceforge.net

[9]http://www.clusterresources.com/products/maui

on a single processor. It is our users' experience that the performance of different codes is affected by the choice of compiler and MPI implementation. As such, the clusters had several different Fortran compilers and MPI libraries installed (both MPICH and LAM implementations). Additionally, various maths Libraries including ScaLAPAC and BLACS were installed, each using their standard configuration.

The three clusters were installed in Cambridge, UCL and Bath, with routine day-to-day maintenance performed remotely from Cambridge. Any additional assistance is provided by project members local to the cluster, if required.

Middleware for user access within the *minigrid* environment

Operating and managing computations within a grid environment is quite unlike running jobs on a standalone cluster. One example difference is that standard tools are primarily designed for use in standalone clusters, which are accessed through the use of *ssh*, or some other direct connection mechanism. This access model does not directly fit with the grid model employed within the *e*Minerals project, where users are not allowed direct access.

The *e*Minerals *minigrid* makes use of several different versions of the Globus software, namely versions 2.4.3, 3.2.1 and 4.0.x are all installed. However, only the functionality of version 2, which has been propagated into later versions is used. These different versions are used in order to promote heterogeneity within both the *minigrid* and the tools developed to access it.

The *minigrid* does not make use of the web services functionality in later versions of Globus, since at the time of its creation, these systems were still very immature. In fact, since the creation of the *minigrid* the web services functionality within the Globus software has been completely re-implemented, moving from OGSA[10] [31] to WSRF[11] standards [19]. Usage of web services within the *minigrid* would have required that each of the different tools and systems be re-developed in order work with the newer standard. In addition, the web services implementations at the time were not able to manage the state of submitted jobs, leading to further required work without any associated gains over the functionality provided prior to the introduction of web services.

The key feature provided by the use of Globus is that access to the clusters is provided through the use of X.509 certificate driven authentication, without the users needing to

[10] http://www.globus.org/ogsa/
[11] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

```
if ( defined($description->mpi_type()) )
{
    if ( $description->mpi_type eq "lam-intel" )
    {
        $mpihome = "/opt/lam-intel";
        $ENV{LAMHOME}=$mpihome;
        $ENV{LAMRSH}="ssh -x";
    }
    ... other cases ...
}
```

Figure 2.1: First modified section of the Globus PBS jobmanager (pbs.pm), enabling the user to choose between different compiler-MPI combinations.

directly log in. Jobs can be submitted from the users' own desktop computers, or from some designated submit machines, using the Globus client tools. User accounts were created for each user on each of the clusters, enabling some accountability and helping to track which user performs each job. The users do not know and, in fact, are unable to change their account passwords on the cluster machines, since they are not needed when using Globus's certificate authentication systems.

This use of one user account per project member also allows for the use of the Scommands as part of submitted jobs, to transfer data to and from the compute resource. The use of the Scommands in this manner is discussed in detail in chapter 3.

Once jobs have been submitted to a cluster via Globus, they are passed on to the cluster's own internal scheduling system. In the case of the *Lake* clusters this is provided by PBS, which handles internal job submission and Maui, which handles the scheduling of jobs until required resources are available.

It was found that the use of several different MPI libraries leads to difficulties with the implementation of the interface between Globus and the underlying scheduling systems. As a result, it was necessary to extend the standard installation of Globus, to enable the user to specify the MPI installation that should be used at runtime. By default the Globus interface to PBS uses a default MPI implementation, as specified by an environment variable on the cluster being used.

```
if ( $description->jobtype() eq "mpi" )
{
    if ( $mpihome =~ "mpich" )
    {
        $pbs_job_script->print("$mpirun -np "
           . $description->count() . "");
        if($cluster)
        {
            $pbs_job_script->print(" -machinefile "
            . "\$PBS_NODEFILE ");
        }
        $pbs_job_script->print(
            $description->executable()
            . " $args < "
            . $description->stdin() . "\n");
    }
    elsif ( $mpihome =~ "lam" )
    {
        $pbs_job_script->print("$mpihome/bin/lamboot"
             . " \$PBS_NODEFILE\n");
        $pbs_job_script->print("$mpirun C "
            . $description->executable()
            . " $args < " . $description->stdin()
            . "\n");
        $pbs_job_script->print(
            "$mpihome/bin/lamhalt\n");
    }
}
... rest of code ...
```

Figure 2.2: Second modified section of the Globus PBS jobmanager (pbs.pm), enabling the user to choose between different MPI libraries.

Mark Calleja, who also worked on the *e*Minerals project at the time, extended the perl implementation of the interface between the Globus and PBS job specifications to be able to accept an extra value in the Globus job's Resource Specification Language (RSL), specifying which MPI implementation to use. This extra RSL tag is then passed to the PBS scheduler (Maui), and the user specified MPI implementation used. If the user does not specify the extra tag, then the default MPI implementation is used, as with the standard Globus behaviour. The modifications to the Globus RSL requires the addition of code to the Globus PBS jobmanager (pbs.pm). This additional code is shown in figures 2.1 and 2.2

The Globus software has been seen to scale very well for most of the different systems to which it is applied within the *e*Minerals project. However, this scalability is very dependent on the underlying scheduling system. When submitting to PBS resources, such as the *Lake* clusters, it has been possible for one user to submit hundreds of jobs at once from one submission machine to one cluster. This easily exceeds the normal usage requirements for these machines, considering the fact that they only contain sixteen processors each. Submission to SGE resources has been shown to scale to similar levels without problem.

The submission of large numbers of jobs to a Condor system via Globus does not scale as well as these other systems. It has been our experience within the *e*Minerals project, that submitting as few as ten or fifteen jobs to a Condor resource in this manner will cause the gatekeeper machine to struggle. This problem has been addressed within the Globus web services functionality. As such, there are plans to move over to using this functionality. However, this will require the re-writing of many of the *e*Minerals tools, and so has not yet progressed beyond the initial testing stage.

2.4.3 System Maintenance and Management

Although the *Lake* clusters are generally reliable and do not require much outside intervention, perhaps once every two to three months, a compute node will stop responding to connection requests from its head node. In this case, project members at the cluster's location simply reboot the node. If this fails to trigger the node to resume communications with the head node, then a clean network reinstall is performed, using system images stored on the head node. These install images are constantly maintained, with any system changes being mirrored within the image. This means that any required reinstall results in a fully functioning installation, with no further steps necessary to configure the node.

In the majority of cases these steps will reset the misbehaving node into a state in which it is again able to accept jobs from users, and can continue as normal. However, occasionally a node will experience hardware failure, in which case project members with more system administration experience will troubleshoot and fix the node. This troubleshooting is performed remotely as far as possible, with the assistance of local project members as necessary.

The head nodes of the *Lake* clusters were also provided with a redundant system disk, allowing the system to be swapped over to the backup configuration in a matter of min-

utes, even by a project member without any formal computing training. However, this backup disk is not an automated mirror of the primary system disk, since any such mirroring would result in corruption of the main system disk propagating to the backup. The swapping to the backup system disk returns the system to an operational state, allowing suitably trained project members to connect to the system and perform troubleshooting tasks, independent of the project members local to the system in question.

2.4.4 Backup System

The backup system employed on the *e*Minerals clusters was originally written especially for the project by Matt Tucker. I extended the system to be more automatic, ensuring that all users are always backed up securely. Before my changes, a static list was kept of the different users to be backed up. This is manageable for small numbers of users, but it quickly becomes impractical with the number of users expected to be given access to grid resources.

The main difference between this system and other, normal backup systems is that it takes advantage of the fact that each of the cluster nodes has a practically unused hard drive. While the operating system is installed on these drives, no users' files are stored there, because of the shared filesystem used within each cluster.

The actual creation of backup archives containing the cluster users' files is a two stage process within this system. Firstly, each evening, an archive is created per user, containing every file that has been changed by that user during the previous twenty-four hours. Secondly, once a week, every file in each user's homespace is archived and compressed into several files. The number of files created depends on the quantity of data being backed up.

Once the backup archives have been created, they are copied to one of the cluster nodes, taking advantage of their unused space. Each week, the node to which the backup is stored is changed in a round-robin fashion. The files are stored on a node in such a way, that the complete archive and each of the daily incremental archives on a single node can be combined, creating the complete archive stored on the next node. Using this method of storing the backed up data, complete backups can be archived for up to a maximum of sixteen weeks before previous backups start to be overwritten. The number of archived backups is dependent on the size of each week's backup, since once a backup is too large to be stored on a single node's hard drive, the archive is split over multiple nodes, using as many as required to store all of the information.

Unfortunately, there are two possible drawbacks associated with the use of this system. One drawback is that the moving of the archived data, results in large amounts of network communication over the same network as used by the cluster for inter-node job communications. However, this has not been seen to be a problem with these clusters, as they are not aimed at parallel jobs, and so do not perform large amounts of inter-node communication anyway. This means that the jobs and backup system do not actually interfere with each other in this case, although that may not be so for larger, more powerful clusters, which can be used in parallel more efficiently.

The second possible drawback is that there is no protection from physical risks, such as fire, or theft. It is felt that the possibility of these risks is relatively low and that they do not need to be protected against within the *e*Minerals project. In terms of the value of the data stored on the clusters, it can all be recreated relatively cheaply if required, and as such, the damage caused by any such loss is minimal. This of course may not be the case for other projects where data are very valuable, in which case necessary safeguards would need to be applied.

2.5 System Monitoring

Since the *e*Minerals *minigrid* is comprised of a large number of compute resources, it became apparent that a single tool to monitor them all automatically would be required. I developed a monitoring tool, which served as a precursor to the development of the metascheduling capabilities of my grid submission tools, as discussed in chapter 3.

The monitoring tool is comprised of two distinct sections. The first section is the actual monitoring system, which gathers, parses, and then stores usage information pertaining to each of the compute and data resources within a database. The second section performs the visualisation of this information, so that it is quickly and easily understandable by the scientist user. These two distinct sections will now be discussed in detail.

2.5.1 Status querying

Periodically, section one of the monitoring system submits Globus jobs to each of the different compute and data resources available. These jobs run queue querying, and disk usage commands, returning the output to the querying machine. The output from these commands is then parsed and all useful information stored within a database.

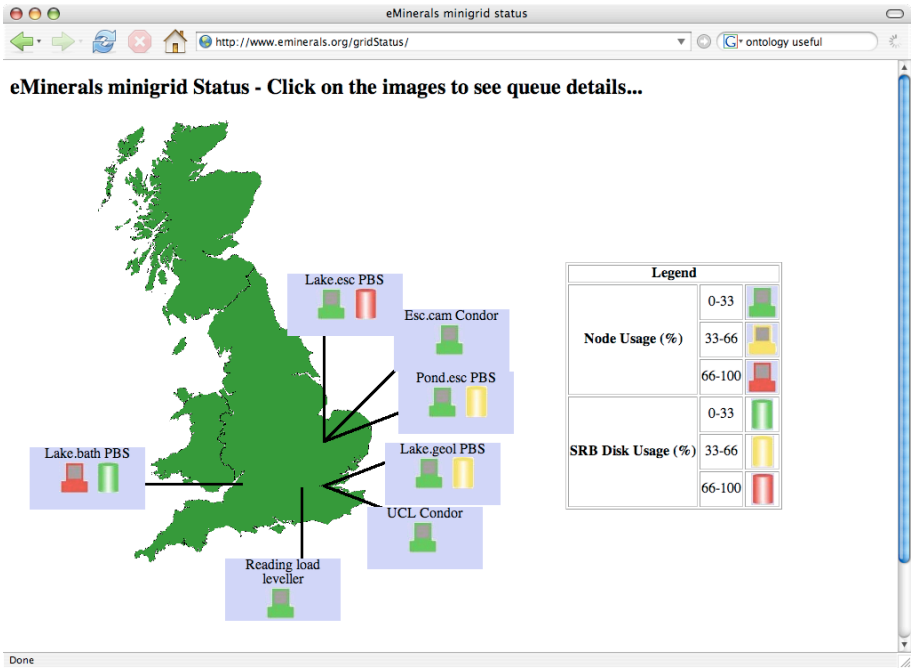

Figure 2.3: The status of the *e*Minerals *minigrid* at the time of writing. Each of the blue boxes represents a compute resource on the *minigrid*. The colour of the computer image gives a simple representation of how heavily the machine is being used. The display and colour of storage cylinders represents the existence of an SRB storage vault and its approximate degree of utilisation.

This process is performed every 30 minutes, giving all users very recent, if not necessarily completely current information. It was deemed more important that the general operation of the clusters not be effected by the querying of their resources, than that the data should be completely up to the minute. The simulations typically performed within the *e*Minerals project can take several hours, if not days. Therefore the caching of usage data does not tend to affect the validity of the data.

2.5.2 Status visualisation

The visualisation of this collected data makes it simple for any *e*Minerals project member to quickly check the status of a resource on the *minigrid*. This means that users can target their jobs at unused resources, reducing the time spent waiting unnecessarily.

Every time a project member accesses the status tool's webpage[12], the database is queried

[12]http://www.eminerals.org/gridStatus

and an image of Great Britain presented, with boxes superimposed around the country. Each box represents a different compute or storage resource within the *minigrid*. Clicking on any of the compute resource boxes, results in the display of the actual queue status output, as provided by the queueing system. This can help the user to see the current status of their own jobs. The different boxes displayed are coloured, dependent on resource usage. Lightly used resources are shown in green, graduating through to red resources, which represent heavy usage. An example of the visualised output from the grid status tool can be seen in figure 2.3.

2.6 Innovations within the *minigrid*

Many of the features of the *e*Minerals *minigrid* have not been seen in other grid systems before, and as such, can be considered innovative in nature. These features, why they are innovative, and how they can be applied to other grid systems will now be described.

Firstly and perhaps most importantly, the *e*Minerals *minigrid* is the first implementation of an *integrated grid*, as described in section 1.2.1. As such, it is the first grid system to integrate compute, data and collaborative resources in this manner. Indeed, many of the tools used were developed specifically for the *e*Minerals project, with the aim that they would be useful to other projects should they be successful within *e*Minerals. Other tools used were developed by other groups at *e*Minerals project member institutions, finding their first production level usage within the *e*Minerals project.

Another innovation is that the *minigrid* is one of the first grids to only allow access through the use of grid computing middleware. Use of *gsissh* is not allowed, since this was deemed too close to traditional usage methods. This restriction to the use of grid middleware is achieved through the use of appropriate security methods, such as the use of X.509 certificates and tools, including the job submission tools discussed in chapter 3

Many grid systems have two main aims. First, to provide the advantages associated with grid computing to the general scientific user community. The second aim is to enlist enough users for the investment in the resources to be deemed to be well spent. These two aims do not necessarily complement one another, and in fact, the latter aim can lead to the dilution of the former. This can be seen by the way that administrators of grid systems often allow their resources to be used in methods that are identical to non-grid systems.

A final innovation incorporated within the *minigrid*, is that it allows access to each of

the heterogeneous resources in a consistent manner. This has been achieved by ensuring that each of the different sets of resources is given a consistent interface, provided by the use of the Globus Toolkit; specifically the functionality from version 2.4.3 of the toolkit, as explained in section 2.4.2.

However, simply using Globus to access the available resources does not allow for completely consistent access throughout the *minigrid*. Rather, this consistency is achieved through the use of submission tools, which hide the specific details of each resource from the user. This allows the user to concentrate on their research, rather than on how to submit to a particular resource. These tools, which will be discussed in chapter 3, allow the user to leverage whole grids of resources, whereas historically using even single clusters could cause problems.

2.7 Expanding beyond the *minigrid*

The *e*Minerals *minigrid* was designed with the aim that it should, as far as possible, function in the same manner as future, larger grid systems. This resulted in the *minigrid* being built to behave generically, allowing the integration and use of different components and systems.

In addition to this, each of the different resources within the *minigrid* have been configured to present a consistent interface, whatever their mode of operation behind this interface. My job submission tools have also been developed to be generic, relying on as little of this consistent interface as possible. This results in tools that will work with other resources, even when they present only a subset of this consistent interface to the user.

The ability for the submission tools to work with any system providing this interface means that the tools can be used on any such system, without modification and in a manner completely transparent and seamless to the user. The migration from *minigrid* resources, to externally managed and operated grid resources has been tested by the use of NGS and NW-Grid resources by *e*Minerals project members.

There is one requirement imposed on any external resources in addition to the need to present a suitable Globus installation. This requirement is that they must install, or at least allow the installation of client tools for the data and metadata management systems used within the project. To this end, packages of various types have been created, allowing these tools to be quickly and easily installed on any required resource. At the time of

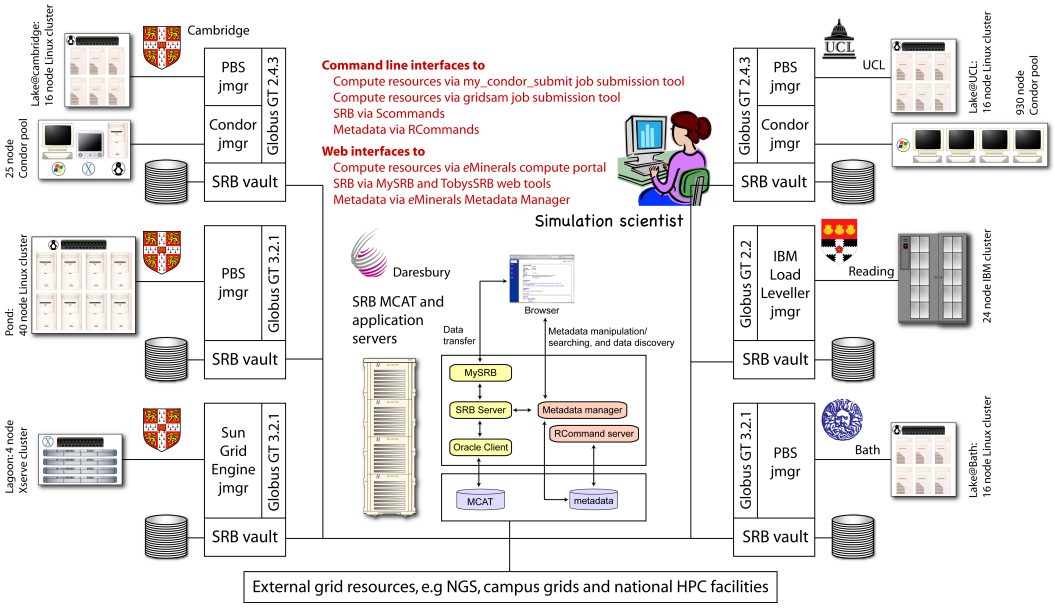

Figure 2.4: Representation of the *e*Minerals *minigrid*. The compute infrastructure consists of the several PC/Linux clusters discussed in detail in this chapter, a small high-memory Apple Xserve cluster (Lagoon), an IBM pSeries parallel computer, and Condor pools in Cambridge and UCL. Most of these resources contain SRB vaults, and the data and metadata components of the *minigrid* are managed by the MCAT server and metadata database in Daresbury.

writing, tools are typically packaged for deb (Debian[13]), rpm (Redhat[14]), and dmg (Mac OS X[15]) based systems, in addition to source code distributions.

Once these packages are installed on the available resource, *e*Minerals project members are then able to take advantage of the new resource, as if it were a part of the *minigrid* all along.

2.8 The current status of the *e*Minerals *minigrid*

The *e*Minerals *minigrid* now combines several different computing resources, each operating similarly to the above described *Lake* clusters, with variations depending on architecture, operating system, etc. The *minigrid* includes four different commodity clusters, three condor pools and one purpose designed cluster machine. In addition to these com-

[13]http://www.debian.org
[14]http://www.redhat.com
[15]http://www.apple.com/macosx/

putational resources, there are six different SRB data vaults and a central application server, which manages the SRB MCAT server and the project metadata systems.

The current status of the *minigrid* can be seen in figure 2.4, which shows each of the different computational resources and the associated SRB data vaults. The central section of the image represents the application server and the different processes it performs. The *Lake* clusters that have been described in previous sections are labelled 'Lake@Cambridge', 'Lake@UCL' and 'Lake@Bath'.

2.9 Other *integrated grid* implementations

When this work was first started there were no other *integrated grid* implementations. However, as time has passed and grid computing research has developed other systems have been implemented. Two such implementations are the Biomedical Informatics Research Network (BIRN)[16] [36] and Enabling Grids for E-sciencE (EGEE)[17] [34] projects, which will now be discussed.

The BIRN project is a virtual community that is distributed geographically across the United States and shares resources in order to advance the diagnosis and treatment of disease. It hosts a collaborative environment, which allows access to all resources in a uniform manner for all of the project's researchers. This in turn allows for multi-institution collaboration much like that provided by the *e*Minerals project.

The research performed within the BIRN project includes the integration of wide area networking, distributed computing and the federation of multiple data sources. As part of this research software is developed that will perform this integration. The BIRN software and resources are used to perform medical research involving experiments on mice, humans and primates. It allows questions to be asked and knowledge shared between all project members, improving their own individual research.

The main difference between the BIRN and *e*Minerals systems is that the BIRN software is necessarily tightly integrated to the systems and processes used by the BIRN scientists. This is because the scientists use very specialised equipment such as Magnetic Resonance Imaging (MRI) machines and the associated data files. The use of such specialist equipment means that the whole system must also be specialised. Therefore they

[16] http://www.nbirn.net/
[17] http://www.eu-egee.org/

cannot be applied to general purpose research, unlike the *e*Minerals systems, which have been designed from the outset for application to any type of computational research.

The EGEE project brings together scientists from over 90 research institutions, in 32 countries around the world. It provides a seamless grid infrastructure that allows access to data and computational resources 24 hours a day. At the time of writing over 40,000 processors and 5 petabytes of storage are made available for use by the project scientists, making EGEE the biggest grid system in the world.

EGEE was initially created with the aim of processing the data that will be created by the LHC experiment. However, the infrastructure has been extended and is now applied to many different types of computational research, ranging from astronomy and life sciences through to civil protection and finance.

As with the *e*Minerals systems, the EGEE infrastructure is designed to be general purpose in nature, allowing all sorts of computational simulations and studies to be performed. However, unlike the *e*Minerals systems, an application must be registered with and deployed across the EGEE infrastructure before it can be used. It can then be accessed by using the gLite software that that will be discussed later in section 3.2.8.

2.10 Conclusions

The main aim of this chapter has been to describe the *e*Minerals *minigrid* and the principle components from which it is made. Also, a case study of the construction and integration of commodity cluster computers has been detailed, showing how such traditional resources can be integrated into a grid.

The most novel feature of this work has been the integration of the compute and data grids through the use of the SRB, for data management, and metadata management tools, which will be described in detail in chapter 3. The data and metadata management tools, and distributed data grid components, have been integrated into the resources at both the hardware and software layers. This is an integration that has not yet been performed in any other grid system, although it is expected to be deployed in other systems in the future.

The other key message in this chapter, which has only been commented on in passing, is that this grid infrastructure has not proved to be difficult to maintain. Running the whole *minigrid*, which is distributed between five sites, has not required dedicated staff

effort, and it has been possible to use untrained team members to help perform occasional troubleshooting tasks when necessary.

The following chapter will now describe how the *e*Minerals project scientists access and utilise the *minigrid* resources, using the job submission tools that I have developed.

Chapter 3

Tools for job submission and management

This chapter discusses my work performed in developing job submission tools to be used by the *e*Minerals project scientists when accessing the *e*Minerals *minigrid*. They have been developed to be easy to use, but still powerful enough to allow the scientist to take advantage of the extra resources afforded by an *integrated grid*. These tools are designed to take care of the whole computational simulation process, from creating relevant input files, through transferring files to the appropriate resource, and executing the simulation. Finally, they transfer the calculated results back to the user.

Parts of the text of this chapter have been adapted from my publication and talk given at the UK *e*Science All Hands meeting[1] in September 2006 [13].

3.1 Grid job submission tool considerations

For grid computing infrastructures to be properly exploited, it is essential that the tools providing access have usability designed into them from the outset. Experience within the *e*Minerals project has shown that it is unrealistic to ask most scientists to work with the raw Globus job submission commands. This is because there are too many commands, which are required to be used in combination with one another. Each of these commands has a complex plethora of options, many of which the user is expected to remember in order to perform even the simplest of operations. Experience has shown that typical users

[1] http://www.allhands.org.uk/

are likely to end up compromising by merely using tools such as *gsissh* to log into grid resources. Once logged in they are able to submit jobs using more familiar and simple batch queue commands. However, it has been found that asking the user to work with Condor job submission scripts is quite feasible [27], allowing them to take advantage of grid systems without being forced to compromise.

As has already been mentioned *e*Minerals project members are not allowed to log into the project compute resources, meaning that they must do everything involved with the job submission and management process remotely. This decision was made since it is likely to be a requirement once the *integrated grid* idea becomes mainstream. This chapter describes the work undertaken to develop tools to handle grid job submission in a way that hides the underlying middleware from the users, and makes grid computing genuinely usable for the project scientists.

Another requirement for grid job submission tools is that they must enable the different sections of an *integrated grid* to behave as one cohesive unit in the eyes of the users. As such, not only must the tools take care of job submission, but they must also consider data issues. This integration must be as automatic as possible, which is important for two reasons. Firstly, any extra work required to perform, for example metadata capture, will discourage potential users from moving to grid systems. Secondly, the more consistent the processes performed by each of the scientists, the easier it will be for collaborators to interoperate and, for example, to search each other's collected metadata.

The main requirement of a job submission tool is to shelter users from the complexities involved in using computational resources of many different types. In doing so the tool must allow the user to automatically perform three stages as part of a single job submission. These stages are:

(i) Copy input data and executable to the remote compute resource;

(ii) The executable must be run, using the input data to create all output data;

(iii) Retrieve the output data to a location accessible to the user.

This simple workflow is exactly the process that computational scientists have been performing manually for many years. However, it does not take advantage of the power of the grid, other than in making use of resources which need not be located at the same location as the user. To take advantage of the data and collaborative sections of *integrated grids*, job submission tools must copy the data to and from appropriate data grid

resources, which must be accessible by all collaborators with appropriate access privileges. This aspect of *integrated grids* is handled within the *e*Minerals project by the use of the SRB system detailed previously.

The steps described so far still do not allow the user to gain all of the advantages provided by the grid. These advantages are primarily associated with improved accountability and ability to locate any created data. To gain these advantages, metadata must be incorporated into the job submission workflow. This allows all aspects of the job to be tracked and archived, so that the user and their collaborators can find this data at a later date. The tools now used within the *e*Minerals project to handle metadata capture and storage are discussed in relation to automated metadata capture in section 3.2.3.

As the grid resources available to users become larger, and the number of different people using the resources increases, the choice of where to send a particular simulation to be run becomes much more complicated. Automated submission tools, such as those described here, should therefore consider the load on each of the different available resources and submit the user's job to a suitable resource. Jobs should only be submitted to a resource that does not immediately have enough spare processing capacity for the job when all available resources are equally busy. This decision process is known as 'metascheduling' and is discussed in detail in relation to the workings of *my_condor_submit* in section 3.2.3.

In addition to all of the features outlined above job submission tools should be flexible, allowing the scientist to perform the research they want, in the manner they want. This is very important, because forcing the scientist to change their mental model regarding their work to too large a degree will lead to the scientist ignoring the advantages provided by the submission tools. This will result in the scientist simply returning to the traditional methods of logging into resources and running jobs manually.

The main thrust of this chapter is to describe *my_condor_submit*, which is the standard job submission tool now used within *e*Minerals. Efforts to wrap *my_condor_submit* to provide other usage mechanisms will also be discussed. These discussions include tools allowing the user to create, submit, and analyse many jobs at once. In addition, efforts that have been made to bring access to the *minigrid* to the user's desktop will be described, including a web browser based tool, the '*e*Minerals Compute portal', and a command line, web services based tool, '*RMCS*'.

3.2 *my_condor_submit* (*MCS*)

3.2.1 An introduction to *MCS*

The main decision made within the *e*Minerals project affecting *minigrid* usability has been to base the tools' input specifications around that used by the Condor project, which has been found to be very simple and easy to work with. *my_condor_submit* (*MCS*)[2] was designed explicitly to take advantage of the simple Condor job specification, extending it to provide the user with a simple way to specify each of the possible parameters related to job submission, within the *e*Minerals project. Previous versions of *MCS*, which were still known by the full name, *my_condor_submit*, have been discussed in detail elsewhere in [16] and [17].

MCS was initially developed by Mark Calleja as a simple stopgap tool to wrap the three stage workflow introduced above while other, more robust tools were developed. This initial tool simply downloaded data from the SRB, executing the user's code and then returning output data at the end of the job. However, *MCS* became unexpectedly popular, requiring that it be further developed and hardened. The initial quick development meant that this extension became unnecessarily difficult, and as such, a full rewrite was performed, allowing for extra features to be added in a much simpler manner.

The current version of *MCS* is provided on the CD included with this thesis and can be found in the *my_condor_submit* directory.

3.2.2 *MCS* design

MCS uses Condor-G[3] [33], a Condor wrapping of the Globus job submission tools, to perform the actual job submission. The process of submitting a simulation is handled as a simple three stage workflow, where first data are downloaded on to the compute resource. Secondly, the actual simulation is executed. Then, finally data are uploaded back from the compute resource to the SRB. This workflow is managed using Condor DAGman[4], which is a simple workflow management system that allows the three steps to be performed in order, waiting for steps to complete before executing subsequent steps.

[2] http://www.eminerals.org/tools/mcs.html
[3] http://www.cs.wisc.edu/condor/condorg/
[4] http://www.cs.wisc.edu/condor/dagman/

This use of DAGman means that *MCS* does not need to directly manage the workflow itself, which can be a very complicated process.

MCS is used with a simple user-supplied input file, which has extensions over a standard Condor submission script. This job specification format allows for simple job management by the user, while allowing for the power exhibited within *MCS* to be used both manually by the user, and automatically from within other tools. *MCS* is designed to be used from the command line, which has been found to be the preferred usage method within the *e*Minerals project. However, it can be wrapped for use through other interfaces by systems such as the java GUI provided with the parameter sweep tools discussed in section 3.4.

The primary purpose of *MCS* is to submit jobs to a grid infrastructure with data management and archiving handled by the SRB. The use of the SRB in this context serves two purposes. First, it is a convenient way to enable transfer of data between the user and a grid infrastructure. This bypasses some of the problems associated with retrieval of multiple files, whose names may not be known beforehand, using Condor and gridftp methods. Second, as has already been mentioned, the use of the SRB enables users to archive all files associated with a study in a way that facilitates good data management and enables collaborative access. This therefore combines a computational grid with the rest of the *integrated grid*.

The main reason for the success of *MCS* was that it matched users' requirements and way of working. This success meant that there was demand for new features to be added in order to enable the user to perform more and more powerful job submissions. As has already been mentioned, one of the first developments involved the complete rewriting of the existing code, modularising it and making it much easier to implement any future developments and upgrades.

Additional functionality allowed for the incorporation of more advanced job submission features, including all of the features described as necessary components of job submission tools above. The developments also included the integration of campus grids and generalisation to other grid infrastructures, including the National Grid Service (NGS) and the NW-Grid systems, which will now be described.

The NGS provides a set of core compute and data clusters in addition to several other contributed resources, which are available for use by UK based scientists. The core resources combine four relatively large clusters operated at four locations around the country. These resources are all accessed via Globus and use PBS to schedule submitted simulations, effectively working as four separate, grid-enabled resources. The compute clusters are aimed directly at the performance of computational simulations, while the

data clusters are aimed at providing the power required for data management, manipulation and mining.

The NW-Grid resources are also a set of four clusters. However, all four clusters are directly aimed at the computational requirements of the users. These clusters are much larger than the NGS resources and are connected by much faster networking interconnects than anything used within the *e*Minerals *minigrid*. This means that they can be used efficiently for large parallel simulations. The NW-Grid system will require that users pay for the computational resources they use, much like the utility computing model discussed in section 1.2.1. However, the *e*Minerals project was given access as part of the machine's configuration and initial testing and as such most of the simulations discussed in chapter 5 were performed using these machines.

The current version of *MCS*, as of September 2006, is version 1.2.0. It has the following main features:

- Full access to the SRB, allowing data transfers using both wildcards and recursion;
- Support for job submission to different grid infrastructures including the *e*Minerals *minigrid*, campus grids, the NGS and the NW-Grid, using the same *MCS* input file;
- Metascheduling with load balancing possible across all resources;
- Integration of metadata and XML tools to automatically generate, collect and store metadata from generated output files, submission and execution environments, and job submission parameters;
- Simple support for both serial and parallel jobs, with knowledge and appropriate handling of the number of processors per physical machine in resources with multiprocessor architectures;
- Support for the use of multiple queues on the same computational resource;
- Support for use with delegated certificate proxies, allowing use from other tools, where the user need not log into the submission machine;
- Use of multiple, redundant databases to remove the risk of a single point of failure from the system;

3.2.3 *MCS* implementation

The *MCS* architecture

MCS combines two separate components, both of which are used for every job submission. The first component is a large Perl program, which is installed on the user's submission machine. The second component is a central database, which is accessed by the first component for each submitted job. These two components will now be discussed.

The user only sees the first component, which once installed on their submission machine can simply be invoked from the command line. This section takes care of each of the functions detailed in the following sections, configuring and submitting the job for the user. Some of these functions are enabled by retrieving information from the central database.

The database component of *MCS* is operated in Cambridge and accessed by each of the different *MCS* installations. This use of a central database system means that every job submitted using *MCS* can take advantage of information about where previous jobs were submitted. This in turn means that all submitted jobs will be distributed evenly across the available resources. However, this use of a central database can introduce a single point of failure, because no jobs can be submitted without access to this database. To overcome this the database is duplicated on a separate computer, which means that one machine failure will not prevent job submissions. When *MCS* detects that it cannot connect to the first database it will attempt to use the second without any user intervention, and indeed, without the user's knowledge.

Various pieces of information are held in the database for each of the different execution machines to which *MCS* can submit. These are primarily pieces of information that are different for each machine. This includes the installation directories for the data and metadata tools, and the command to be used to check the status of the resource's queue. In addition, information regarding the architecture of the machine is held, enabling *MCS* to retrieve the correct executable file from the SRB as part of its metascheduling functionality.

The key piece of information that is stored in the database, which requires the use of a central database is the 'rank' of each of the machines. The machines are ordered by this rank with the machine that was submitted to most recently having the lowest rank, and that which was submitted to first the highest. Each machine's rank is changed automatically by *MCS* every time a job is submitted. The rank is changed through the

use of a database user with very few privileges, meaning that they can change the rank, but not remove or insert any other information, thereby keeping the database secure. The use of rankings in this manner allows *MCS* to load balance across each of the possible execution machines *MCS* prefers to submit to a machine with a higher rank, as is detailed in the discussion of metascheduling below.

Another advantage given by using a central database system is that any change to the details held about an execution machine must only be performed in the central database, and its backup. Using one database per *MCS* installation would require the changes to be made for every installation, which is unlikely to occur without error. This also means that the changes can be made by project members who have expert knowledge of the workings of *MCS*, rather than expecting the users to make them. This helps to keep *MCS* simple to use.

MCS is not restricted to the number of databases it can use. As has already been mentioned, two redundant databases are used within the *e*Minerals project. This number has been found to be ample to avoid the single point of failure problem, but more may be required for larger projects. In order to simplify their operation, the two databases are not mirrored, meaning that the machine rankings held in the two databases are not synchronised. This does not cause a problem with the load balancing operation performed by *MCS* when the use of the backup database is required, because metascheduling decisions are based on a combination of these rankings and the machines' respective levels of usage.

MCS **prerequisites**

Two software systems are required to be installed on the submission machine in order for *MCS* to be able to function. These two pieces of software are the Globus and Condor systems. These requirements will now be discussed.

Condor is required in order for *MCS* to make use of its workflow management system, Condor DAGman, which was mentioned previously. In addition, the jobs are actually submitted using Condor-G, which wraps Globus, allowing for simpler job submission commands. This method also allows for the monitoring of the submitted jobs using the standard Condor monitoring tools, helping the user to trace their jobs once submitted.

The primary requirement for Globus is to provide the required authentication functionality. This allows the submission machine and computational resource to trust one another and removes the requirement for the user to connect directly to the computational resource being used.

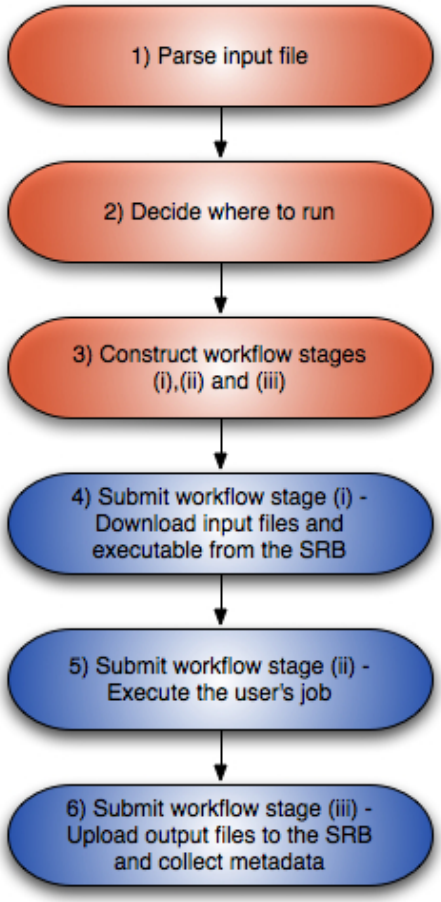

Figure 3.1: Graphical representation of a simplified version of the *MCS* algorithm. Steps in red are those performed internally to *MCS*, whilst those in blue are initially created by *MCS* and then submitted to and handled by Condor DAGman.

Globus is also needed because it allows *MCS* to query the current status of each available computational resource. This is required in order for *MCS* to make decisions as part of its metascheduling functionality, which is described shortly.

However, the use of these prerequisite systems does not come without disadvantages. Firstly, they can be complicated to install correctly and even more difficult to debug a malfunctioning installation. To reduce these difficulties the *MCS* user manual gives installation instructions, telling the user how to set up a standard installation of these systems.

The second disadvantage is directly related to the use of the Globus system. The subset of the Globus functionality used by *MCS* cannot be installed when using the Windows family of operating systems. This means that *MCS* itself cannot be installed on, or used

directly from computers using these operating systems.

Efforts to overcome each of these problems and to reduce the number of systems that the user must install on their computer are discussed in section 3.3.

The *MCS* algorithm

A simplified version of the algorithm implemented by *MCS*, which ignores features such as repetition and error handling, is illustrated in figure 3.1. All six steps of the algorithm are generated and performed by *MCS* itself. However, once *MCS* has configured the required data transfers and metadata capture, control is passed over to Condor's DAGman tool to execute and manage the final three steps.

MCS works by first deciding where to run the user's job, depending on resource and executable availability. The specifics of the *MCS* algorithm relating to this metascheduling are described below. Once a suitable execute resource has been chosen, *MCS* creates a simple three stage workflow relating to stages 4,5 and 6 in figure 3.1. These stages are managed by Condor DAGman, which ensures that steps 5 and 6 only start once the preceding stages have completed. The three stages of actual job execution are:

(i) Setup the job on the remote resource. (Step 4 in figure 3.1);

(ii) Execute the user's job. (Step 5 in figure 3.1);

(iii) Upload output data to the SRB and collect metadata. (Step 6 in figure 3.1);

This three stage process allows for flexible control over the data transfers, which occur in steps (i) and (iii) (stages 4 and 6 within figure 3.1). It also allows for flexible control of the executable, allowing the user to run, in the most simple case, a single executable, or in the more complicated case, scripts that perform their own workflow running several executables, as part of one job submission.

Metascheduling

One problem experienced with the initial stopgap versions of *MCS* was that the user had to specify which compute resource any simulation should be submitted to. This resulted in many jobs being submitted to a few busy resources within the *e*Minerals *minigrid*, whilst other resources were left idle. Since users are not allowed to log in to resources,

it was not possible for them to check job queues; in any case, such an approach is not a scalable solution to resource monitoring and is definitely not practical on the scale afforded by grid systems.

An intermediate, partial solution to this problem was the creation of the status monitoring tool, which was discussed in detail in section 2.5. However, this solution still required user involvement to look at the page and decide where to run. Moreover, there is no way that this model can scale to the number of jobs required to make use of the resources made available by grid methods. It is not possible for a user to judge ahead of time where to send each of a large number of jobs in order to distribute them across available resources.

To solve this problem and to allow better usage of project resources, querying of resource utilisation must be built into the submission mechanism, using up-to-date rather than cached data. This type of metascheduling, based on a round-robin algorithm, has been built into *MCS*.

The metascheduling feature of *MCS* is performed as part of stage 2 of figure 3.1, prior to the creation and submission of the actual job workflow. It must be performed at this stage in order for decisions made when metascheduling to be reflected within the jobs submitted as part of the job workflow. For example, the directory containing the data and metadata management tools varies on each of the execution machines. This information must be included within the created jobs and so must be known before the jobs are created by *MCS*.

To use this metascheduling facility, the user must simply specify the type of machine they wish to submit to, using the keywords '*performance*' to submit to cluster machines, or '*throughput*' to submit to Condor pools. *MCS* then retrieves a list of machines from the central database, querying the machines in turn, checking for available processing capacity.

The querying of resource status is performed by sending a Globus fork job[5] to the resource. This job executes the relevant local queue command, details of which are obtained from the central database. For example, querying the status of a PBS cluster will result in the use of the '*pbsnodes*' command with suitable parsing of its output. Querying one of the available Condor pools, on the other hand, will use a command written specifically for this purpose, wrapping the standard '*condor_status -totals*' command. The need for a

[5]Fork jobs are jobs that operate within a new process spawned from the calling process. In terms of the Globus system, fork jobs execute on the machine running the Globus server processes.

purpose-written command is due to the requirement to poll flocked[6] Condor pools within a campus grid, which the standard commands do not perform. Should all resources be found to be busy, *MCS* will inform the user, and queue the jobs using information in the database to ensure an even balance across the different resources.

As part of the metascheduling, *MCS* takes account of the fact that not all binary executables will run on all resources. Information regarding the choice of executable to use on each machine is extracted from the database. The user must store executables for the various platforms within the SRB. *MCS* will automatically set up the appropriate SRB file download, ensuring that the downloaded binary can be executed on the chosen resource, removing this burden from the user.

The metascheduling capabilities now built into *MCS* have much improved the usage patterns within the *e*Minerals *minigrid*. They have also allowed the *e*Minerals scientists to make much more efficient use of external resources than has previously been possible.

As has already been mentioned, the *minigrid* was used unevenly prior to the development of the metascheduling functionality within *MCS*. Often, resources would be completely idle with no jobs to perform, while others were very busy with many more jobs queued than could be executed. Since the introduction of the metascheduling capabilities into *MCS*, the resources of the *minigrid* are used more uniformly.

The metascheduling functionality ensures that anybody using *MCS* will have their jobs immediately scheduled on any available processors, before any jobs are sent to a resource on which they would have to wait to be scheduled. Once all available resources are busy, any submitted jobs will be distributed throughout the available resources, thereby attempting to minimise the time spent waiting for free processors.

There are, however, some limitations inherent in the metascheduling algorithm implemented by *MCS*. These limitations are:

- Job runtime is ignored;
- The length of any existing queue on compute resources is ignored;
- All resources are considered equal in power.

[6]The Condor project uses the term 'flock' to mean the process of connecting between two or more separate Condor pools, allowing for the delegation of jobs from one pool to another. Two pools that are said to be flocked are configured to accept jobs in this manner.

Job runtime ignored - Unlike some other metascheduling systems, *MCS* does not keep details of previously scheduled jobs. Systems that do keep track of these details are able to track the time spent queueing, and the runtime of any job on each resource. Once this information is stored, it is possible for the metascheduling algorithm to make use of this data, sending jobs to resources where it is likely to have to queue for as little time as possible.

This information is not available to *MCS* though, which means that it cannot take advantage of the predictions that can be made. As such, *MCS* can only decide where to run based on current processor availability. In the event that no processors are immediately available to accept a job, *MCS* is only able to select a resource based on a round-robin technique. This means that each queue will receive a similar number of jobs. However, it is possible that all of the jobs running on a particular resource are short running, in which case this resource would be able to schedule the queued jobs much quicker than another resource running longer jobs, and so would be the preferred choice of execute resource should the algorithm be able to take into account this information.

This shortcoming has not presented a serious problem within the *e*Minerals *minigrid*, because most of the jobs performed within the project tend to last for a similar length of time. This has meant that, assuming each available resource is given similar numbers of jobs, it is unlikely for one resource to suddenly become available while others remain very busy. However, this is not likely to be the case in mainstream usage, where different users will want to run jobs of very differing lengths. As such, for *MCS* to be used in the more mainstream community, it would need to include the ability to capture and use this information.

In addition, the vast majority of jobs submitted within the *e*Minerals project are run as suites of jobs using the tools discussed in section 3.4. Jobs submitted on this scale will automatically be distributed throughout the available resources. This means that only some of the user's jobs will be queued behind long running jobs, reducing the seriousness of this limitation. This allows the user to consider and analyse the results of the jobs that finish sooner while waiting for the jobs that are still queued.

Existing queue length ignored - The *MCS* metascheduling algorithm only considers the number of processors currently available and unused; it does not consider the number of jobs already waiting to be scheduled to these processors. As a result *MCS* cannot prioritise between a resource on which all processors are used, but no jobs are waiting, and another resource where all processors are used and dozens of jobs are already waiting

to run. This can result in jobs being added to a queue that already has large numbers of jobs waiting, when other resources have queues with substantially fewer jobs waiting to be executed. This in turn can result in the user having to wait for longer than necessary for their job to finish executing.

Within the *e*Minerals *minigrid* this has not typically been a problem, due to the fact that most jobs are submitted using *MCS*. This results in the jobs being distributed evenly across all available resources, since each metascheduled submission is made using the same central database. However, this distribution will not occur when submitting to external resources, where users are able to submit using a system other than *MCS*, and which cannot perform a similar load balancing function. This is because *MCS* will not be able to take into account the decisions made by any other submission systems as part of its own decision process.

This problem is especially evident when users are able to submit jobs manually to a set of available resources. This is because it is unlikely that users using this method will distribute their jobs throughout available resources. Instead they are likely to submit all of their jobs to one resource, potentially over-filling this resource to a level where it adversely affects other users.

It is therefore important that all future submission systems are able to perform some level of metascheduling and load balancing, ensuring that all users wait for the minimum amount of time possible before their jobs are executed. However, this is not currently the case and is unlikely for the foreseeable future, while users are able to submit manually without the proper use of grid middleware.

For *MCS* to be more efficient in its use of resources external to the *minigrid*, and for improved mainstream usage, it will need to take into account the lengths of existing queues on available resources. However, this limitation has not yet been a problem within the *e*Minerals project, even when using external resources, because all of these resources have not typically been heavily utilised at any one time. This has meant that *MCS* has been able to submit jobs to directly available resources, rather than submitting to a resource on which a queue already exists. This will not be the case in the future though, when these resources become more heavily used and it is less likely that jobs will be able to be directly submitted to available resources without any associated queueing.

All resources are considered equal in power - *MCS* does not consider the difference in power provided by the different resources to which jobs can be submitted. Instead it assumes that each resource can provide the same power, which greatly simplifies the

metascheduling algorithm. This means that if the user wishes to run a job that needs very powerful resources then they must manually restrict the list of machines to which they want *MCS* to submit in the *MCS* input file, to include just those that can provide enough power.

This is not really a hardship, because the user generally knows ahead of time whether their jobs will require the more powerful resources. In which case they can easily direct *MCS* to use these machines and to not submit to lower powered resources.

In addition, if *MCS* were to be able to take into account the power of each of the resources then the user would need some method for specifying the level of power required by their jobs. Any such method is likely to be less natural for the user to specify than the current method of listing the machines that they are happy to use. In which case, specifying the level of power required is likely to be more difficult to use than the current system, meaning that it is less likely to be used.

Data staging

The data staging sections of *MCS* relate to stages (i) and (iii) of the *MCS* workflow, where data are copied to and from the compute resource. These are shown as steps 4 and 6 in figure 3.1. This staging allows for the running of jobs and for the retrieval and archiving of any generated output data.

Within the *e*Minerals project, the decision was made to transfer all data to and from the execution resource using the SRB. For this to happen, data must be uploaded to the SRB from the scientist's work machine before submitting a simulation. This decision means that the project submission tools must take care of downloading submission input files from the SRB to the chosen computational resource, and uploading created output files back to the SRB after the job's execution has completed. This use of the SRB for all data means that the scientist is able to archive the data files associated with every simulation performed, without having to worry about the amount of space required to do so.

Step 4 of the *MCS* algorithm, shown in figure 3.1, takes care of the downloading of data from the SRB to the chosen compute resource. This is handled by a Perl program called '`pre.pl`', which is automatically generated by *MCS*. `pre.pl` connects to the SRB on behalf of the user, downloading their required files using the Scommands. It incorporates error detection and handling, retrying commands until they succeed in the event of recoverable errors, such as brief network interruptions, etc.

Stage 6 of the *MCS* algorithm is handled by a similarly generated Perl program called '`post.pl`'. This program is more complicated than `pre.pl`, because it must also handle metadata capture and storage. These additional processes are described below as part of the discussion of metadata collection within *MCS*.

The actual computer on which the `pre.pl` and `post.pl` programs are executed depends on the computational resource used. When submitting to a cluster resource the programs are executed on the cluster's gatekeeper node, which is the machine that allows access to the cluster's scheduling system via the Globus software. The user's files are then made available to the execution node, where the user's job is actually executed, via the cluster's shared filesystem.

The process is slightly different when using Condor pools, which typically do not have such a shared filesystem. When using these systems the `pre.pl` and `post.pl` programs are also executed on the resource's gatekeeper node. However, the files are then transferred to and from the execution node by the Condor software. In this scenario, the user must specify the names of the files to be transferred to the Condor software directly as a part of their *MCS* input file.

Data transfers are specified within the *MCS* input file through the use of a number of different statements within the input file. Several of these statements can have default behaviour, and so need not be specified for most job submissions. This means that less-advanced users need not concern themselves with these options. The different statements and their usage will now be discussed.

The first of these statements, '`Sforce`', instructs *MCS* as to whether files being transferred to or from the SRB should overwrite existing files. `Sforce` can be used as follows:

```
Sforce = [true|false]
```

The default value for this statement is `true`, which ensures that files are always uploaded as specified by the user and that the complete output from a simulation will be uploaded. Should the user set the value to `false`, then it is possible that output files from the simulation will be mixed with files from an older simulation, making it unclear what files relate to which simulation. Those files that already exist will not be overwritten, but the new versions will be left on the computational resource where they were created, allowing for manual retrieval if required. This behaviour could be desirable when, for example, some of the older simulation's files are important and must not be overwritten, hence the provision of this option.

The second statement, '`Sdirect`', instructs *MCS* as to whether to transfer files directly between the SRB vault and the compute resource. The specification for this statement is as follows:

```
Sdirect = [true|false]
```

The SRB's default behaviour is to transfer all files via the central MCAT server, which is equivalent to '`Sdirect = false`'. This means that even if the SRB vault is on the same machine as that from which the files are being transferred, they will travel all the way to the MCAT server and back again, creating large amounts of load on the server. By default, *MCS* sets `Sdirect` to `true`, which forces the SRB to transfer files directly to and from the vault, reducing load on the MCAT server.

This direct transfer requires that firewalls in between the compute resource and the associated vaults all allow relevant traffic, which may not always be possible. In the case where firewalls block this traffic, the user is able to set the statement to `false`, allowing traffic to travel via the MCAT server, reducing the number of firewall holes required.

The final statement for which a default value is set is '`Srecurse`', which instructs *MCS* as to whether to make all SRB uploads and downloads recursive. It can be specified as follows:

```
Srecurse = [true|false]
```

The default value is `false`, which means that all SRB downloads and uploads are non-recursive, and will ignore any directories found. The user may override this behaviour by specifying a value of `true`, which will make *MCS* recursively download and upload any directories it finds within the SRB and local filesystems. An example where this behaviour is necessary is when the simulation code being used requires a specific directory structure for its input and configuration files on the execution machine.

The other data management statements that can be specified within the *MCS* input file cannot have default values set, due to their usage being dependent on the user's requirements. These statements and their usage will now be discussed.

The first of these statements, '`Sdir`', allows the user to specify the path within the SRB to a directory that they wish to upload to, or download from as part of the their submitted job. This statement is used as follows:

```
Sdir = <Path to the SRB directory to transfer files to / from>
```

The user must specify either the full path within the SRB to the directory to be used for file transfers, or the path from their home directory within the SRB. This ensures that the user can specify multiple `Sdir` statements, telling *MCS* to transfer files to or from each of the different listed paths.

The second and third statements, '`Sget`' and '`Sput`', allow the user to specify files to download from, or to upload to the SRB at the start or end of the job respectively. These statements are used as follows:

```
Sget = <List of files to download from the SRB to the compute resource,
                              including wildcards>
Sput = <List of files to upload from the compute resource to the SRB,
                              including wildcards>
```

These statements work identically, other than the direction of their transfer, hence their combined discussion. The user can enter a list of files as the argument to each of these statements and can use wildcards within any of the specified filenames. For example, '`Sget = *.psf`', would download all files whose names end in '`.psf`' within the current SRB working directory. The directory to transfer to or from is specified by the preceding `Sdir` statement. Each `Sdir` statement can be followed by a maximum of one `Sget` and one `Sput` statement.

Metadata capture

Simply being able to store and archive all of the data created is a large step towards handling the deluge of data created. However, it does not help the scientist to deal with the sheer number of data files created, or with any further data interaction once the data have been created. For example, a scientist can easily produce several thousand data files within a single day when working with systems on the scale enabled by grid technologies. Simply archiving these files does not assist with analysing trends in this data.

When creating such a large number of files every day, the scientist will quickly lose track of which files relate to which piece of research. To tackle this problem, metadata capture and storage has been built into the data grid section of the *minigrid*. This integration of metadata tools, with the other *minigrid* resources is addressed by the installation of suitable metadata collection software on each of the available computational resources.

```
<?xml version="1.0" encoding="UTF-8" ?>
<cml xmlns="http://www.xml-cml.org/schema"
 xmlns:siesta="http://www.uam.es/siesta/namespace">
 <metadataList>
  <metadata name="siesta:Program" content="Siesta"/>
  <metadata name="siesta:Version" content="siesta-2.0-release"/>
  ...
 </metadataList>
 <parameterList title="Input Parameters">
  <parameter name="SystemName" dictRef="siesta:sname" value=""/>
  <parameter name="SystemLabel" dictRef="siesta:slabel" value="0CDD"/>
  ...
 </parameterList>
 <propertyList>
  <property title="Mesh" dictRef="siesta:ntm">
   <array size="3" dataType="xsd:integer">192 150 360</array>
  </property>
  ...
 </propertyList>
 <module serial="1" role="step">
  ...
 </module>
 ...
 <module title="Finalization">
  ...
 </module>
</cml>
```

Figure 3.2: An example CML file, showing lists of 'metadata', 'module', 'parameter' and 'property' elements. This file does not contain enough information to be useful computationally, but it is a useful demonstration of the language syntax and is taken from a valid SIESTA output file. The '...' lines have been used to replace document contents that are not relevant to the discussion here.

The *e*Minerals project has been systematically changing our simulation codes to write their output and configuration files in the eXtensible Markup Language (XML)[7]. Specifically CML[8], an XML language, has been used. A Fortran library, 'FoX', has been developed to facilitate the writing of CML files [77]. Use is also made of a library called *AgentX* [18], which facilitates general reading of XML files into programs written in Fortran and other languages through the use of logical mappings instead of concrete document structure. *AgentX* is described in detail in appendix A.

The CML files output by the simulation codes used within the *e*Minerals project broadly have their data separated into four components; '`metadata`', '`module`', '`parameter`' and '`property`'. An example CML file, showing each of these components, can be seen in figure 3.2.

[7]XML is a general purpose markup language designed to create special purpose languages describing data of all types. Example XML languages include GML, which describes geographical data, and MathML, used for describing mathematical data.

[8]CML is a language which has been designed to represent chemical data and is used extensively within the *e*Minerals project. This work is discussed in chapter 4. Appendix B presents an in depth discussion of CML.

The `metadata` component consists of general information about the job, such as program name, version number, etc. The `parameter` components are mainly reflections of the input parameters that control, and subsequently identify, the particular simulation. Examples range from the interesting parameters such as temperature and pressure to the more mundane, but nevertheless important, control parameters such as various cut-offs. The `property` components of the output are the data values calculated by the simulation, and are often output both during and upon completion of the run.

The `module` component is different from the other components described here, because it relates directly to the execution of a simulation rather than to the system being simulated. It is used to divide output files into simulation steps, much like traditional output file formats do. Other components are then placed within `modules` to group together step-by-step details of the system considered by the simulation.

`module` components are used heavily within simulation code output in order to divide the simulation into its different stages. SIESTA and DL_POLY are two codes that make extensive use of `module` components, but in different ways. SIESTA uses a `module` for each step performed as part of a relaxation simulation. DL_POLY on the other hand uses them for each timestep performed in a simulation. By convention, the last `module` written within a CML file is used to group together the final calculated properties from the simulation run.

In order to attach meaning to the value and usage of these generic components within a CML document, they can each be given dictionary references, which are labelled with the term '`dictRef`'. Each code using CML will have its own dictionary, allowing the user to look up the meaning of a labelled value within the code's file, retrieving information such as how the value was derived, etc. Even when the dictionary is not provided the dictionary references can be used to identify the simulation code from which the data comes, helping the user to trace the meaning of the data. An example SIESTA dictionary reference is '*siesta:StartTime*', which refers to the time at which the associated SIESTA simulation began executing.

The simulation code output files combine lists of each of these components. These lists of data are vast, and include step-by-step data, as well as final values or averages. Typically, for metadata collection, it is necessary to retrieve some data from each of the `metadata`, `parameter` and `property` components. It is worth remarking that with regard to the property metadata, the distinction between data and metadata is blurred, which is illustrated by a simple example. In a study of the binding energy of a family of molecules, such as the PCDD family discussed in chapter 5, the calculated values of the final energy

for each molecule are stored as metadata. This allows the use of this energy within our metadata search tools, where an example would be to search through a study of all PCDD molecules, for the molecule with the lowest binding energy.

MCS can capture up to four different types of metadata per submitted job, each of which are stored within the *e*Minerals metadata database. These types of metadata and how they are requested within an *MCS* input file will now be discussed.

Environment metadata is information collected about the job's submission and execution environments. This includes data such as the submission and execution machines, the submission and completion time, and the username on each machine. The user can specify that they wish to collect environment metadata by adding '`GetEnvMetadata = true`' to their *MCS* input file.

Default metadata is information collected from the simulation code's CML output file. Each of the metadata components in the first metadata list is collected and stored, since these must be important as metadata for them to be written by the code author in the first place. In addition, each of the parameter components in the first parameter list is collected, since these are the job's specified input parameters. Default metadata is collected when the user specifies '`AgentXDefault = <filename>`' in their *MCS* input file, where '`<filename>`' is the name of the CML file from which to collect the metadata.

User specified metadata strings are sentences that the user wishes to store to help provide context for the submitted job. An example of this type of metadata is the specification of whether the job relates to a test, or a full production run. This example usage would be specified within an *MCS* input file by adding '`MetadataString = jobType,"production"`', where '`jobType`' is the label to be applied to the metadata and '`production`' is the actual string of metadata to be stored.

User specified CML data. The user can specify that they wish to collect specific pieces of data as metadata from their created CML files. This type of metadata is the most complicated to collect. However, it is arguably the most important type and so will now be discussed in detail.

Within *MCS*, metadata is extracted from the XML data documents using the *AgentX* library. Once the data has been located using *AgentX*, *MCS* associates the value with

a user-specified term, such as '*FinalEnergy*', that provides the context of the data. The data and related term are then stored in the project metadata database. The call to *AgentX* in the *MCS* input file follows an XPath-like syntax[9].

A simple example follows, which demonstrates the extraction of a metadata item stored at the start of a SIESTA simulation run, relating to the start time of the simulation. This piece of information can be identified within the output file by the use of its dictionary reference, which is 'siesta:StartTime'. This value is actually collected automatically by *MCS*, as part of its default metadata, but it serves here as a useful simple example of an AgentX metadata capture line.

```
AgentX = siestaStartTime, chlorobenzene.xml:/MetadataList/
          Metadata[dictRef = 'siesta:StartTime']
```

The following is a more complex example, which extracts the final energy from the same SIESTA simulation output. The final energy value is worth collecting as a metadata item, because it is a useful quantity for data organisation, and for metadata search tools. However, it is not as simple to select as the above example and so requires a more complicated entry in the *MCS* input file:

```
AgentX = finalEnergy, chlorobenzene.xml:/Module[last]/
          PropertyList[title = 'Final Energy']/
          Property[dictRef = 'siesta:Etot']
```

This directive specifies that *MCS* is to extract the metadata as a value from the file called '*chlorobenzene.xml*'. This value is then to be associated with the term '*finalEnergy*'. The *AgentX* call looks for this value within the last `module` container, where as described above, `module` elements are used within CML documents to collate together properties related to a single simulation step. By convention, the last `module` holds properties representing the final state of the system. Next, *AgentX* looks within a `propertyList` with title '*Final Energy*', and finally retrieves the value of a `property` element, defined by the dictionary reference '*siesta:Etot*'.

All metadata within the *e*Minerals project is stored and managed using the *RCommands* [69]. They are a set of command line tools that have been developed by another

[9]The syntax is not quite the same as XPath, but is based on it. The XPath specification can be found at: http://www.w3.org/TR/xpath

Metadata element	Example 1	Example 2
Study	A molecular dynamics simulation of silica under pressure	An *ab-initio* study of dioxin molecules on a clay surface
Dataset	A set of simulations identified by the size of the system being modelled and the interatomic potential model used	A set of simulations identified by the number of chlorine atoms within the dioxin molecule and the specific location of the dioxin above the surface
Data object	A directory within the SRB containing the input and output files created by a single simulation	A directory within the SRB containing the input and output files created by a single simulation

Table 3.1: Two examples of how each of the metadata model levels have been used when organising data within the *e*Minerals project.

project member specifically for the storage and indexing of metadata. Metadata can be stored at three different hierarchical levels, called '*study*', '*dataset*', and '*data object*'. These levels represent a subset of the CCLRC metadata model [61], and allow for flexible grouping of metadata, matching the user's own mental model of their research. Examples of the use of the different metadata elements are given in table 3.1. The *RCommands* are discussed in detail in appendix C.

MCS is able to create and store metadata at both the *dataset* and *data object* levels. The *study* level was thought to be too abstract to be considered within *MCS*, and as such, must be controlled by the user manually using the *RCommands*.

MCS is able to collect and archive metadata in such a way that each *dataset* may contain one or several *data objects*, depending on the type of simulation being executed, as specified by the user. An example use, where each *dataset* would contain a single *data object*, would be when a calculation being performed is standalone in nature. Alternatively, when running a whole sweep of calculations, each with slightly differing properties, it makes sense for each simulation to create its own *data object* within a single *dataset*.

3.2.4 Obtaining and learning to use *MCS*

MCS is provided for any *e*Minerals user to download from the project SRB system. This delivery method was chosen over methods such as downloading from the *e*Minerals website, because it helps to ensure that the prospective user has accounts on the required

grid systems, including the computational resources and software systems such as the SRB, *RCommands*, etc.

Much effort has been made to make *MCS* easy to install and to use. To this end, installation packages are provided in formats suitable for various different Linux distributions, allowing for system-wide installation by someone with suitable administrative privileges. In addition, *MCS* is made available as a source code archive, which can be installed by any user, or on any system for which an installer is not provided. Each of these packages can be found in the my_condor_submit directory of the enclosed CD.

The *MCS* distribution comes with an extensive user manual, which includes details of how to install *MCS* and each of its prerequisite software packages. It also describes how *MCS* works, and how the user can make use of each of the different input file statements. This manual allows any advanced user to quickly and easily discover how to perform their required task, while less advanced users can use the provided examples to create their own jobs.

In addition to the user manual, a second manual is provided, which lists and explains each of the error messages that *MCS* can return to the user. Many of the common grid tools provide little documentation or help regarding the error messages they return, which are often vague and leave the user confused. This manual attempts to prevent that problem for *MCS*. *MCS* itself outputs meaningful error messages, rather than simply returning an error number. The manual combines these meaningful messages with suggestions as to how the user may prevent the error from reoccurring.

3.2.5 *MCS* testing

As part of the development of *MCS*, extensive test cases were created, ensuring that *MCS* works as expected. Tests were written to consider every possible input file statement and to consider both valid and invalid values for these statements. In addition, both valid and invalid combinations of these different statements were tested in order to prevent the user from entering options that cause *MCS* to behave unexpectedly.

A final manual provided with the *MCS* distribution details each of these extensive tests. Each test case is listed with a description of its purpose. The available options within the *MCS* input file are then listed and cross referenced against the test files that apply to each option.

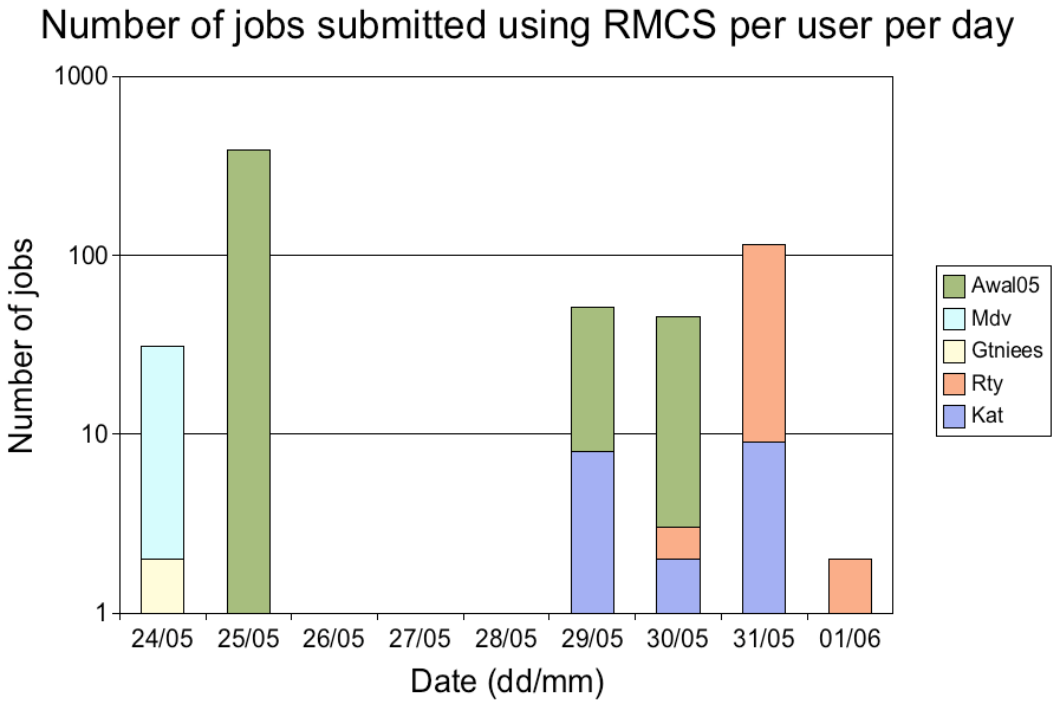

Figure 3.3: A graph showing the usage of *RMCS* (and therefore *MCS*) per user per day for a period of 9 days. Note that the period where no jobs were submitted relates to a bank-holiday weekend during which no use of *RMCS* was made. This data can also be seen in table 3.2.

The combination of this document and the actual tests themselves allow for simple testing of new *MCS* developments. They also help to ensure that new *MCS* versions are compatible with the input files used by older versions. This means that the user is not forced to re-write any tools that make use of *MCS*, unless they want to take advantage of the newly implemented features.

3.2.6 *MCS* usage statistics

As has already been mentioned, *MCS* is used extensively within the *e*Minerals project. This section gives some quantitative details relating to the levels of usage. *MCS* itself does not keep track of usage statistics as there has been no requirement for this information as part of the development and use of my tools. *RMCS*, which is discussed in section 3.3.3, does keep usage statistics as part of its operation. Therefore the following discussion refers to usage of *MCS* through *RMCS* only and does not reflect standalone usage of *MCS*.

	Username					
Date	Awal05	Gtniees	Kat	Mdv	Rty	Total
24/05/2007	0	2	0	29	0	31
25/05/2007	386	0	0	0	0	386
26/05/2007	0	0	0	0	0	0
27/05/2007	0	0	0	0	0	0
28/05/2007	0	0	0	0	0	0
29/05/2007	43	0	8	0	0	51
30/05/2007	42	0	2	0	1	45
31/05/2007	0	0	9	0	106	115
01/06/2007	0	0	0	0	2	2

Table 3.2: Data related to the usage of *RMCS* (and therefore MCS) per user per day for a typical week. Note that the period where no jobs were submitted relates to a bank-holiday weekend during which no use of *RMCS* was made. This data is graphed in figure 3.3.

Figure 3.3 shows the amount of jobs submitted using *RMCS*, and therefore *MCS*, for a typical 9 day period in May 2007. This data is also listed in table 3.2 in order to improve clarity. The total number of jobs per day is separated into the jobs submitted by each user. These figures are typical for usage of my tools within the *e*Minerals project, showing

```
1   # Specify the name of the executable to run
2   Executable      = gulp
3   # Specify the stdin and stdout filename
4   Input = andalusite.dat
5   Output = andalusite.out
6
7   # Specify an SRB directory to get the relevant executable from
8   pathToExe        = /home/codes.eminerals/gulp/
9
10  # Specify a metadata dataset to create all metadata within
11  RDatasetId   = 55
12
13  # Specify a directory to get files from, put files to and
14  # relate to metadata created below
15  Sdir     = /home/user01.eminerals/gulpminerals/
16  Sget     = *
17  Sput     = *
18
19  # Creates and names a metadata data object
20  Rdesc    = "Gulp output from andalusite at ambient conditions"
21  # Specify metadata to get from files with AgentX - get
22  # environment and default metadata only
23  AgentXDefault    = andalusite.xml
24  GetEnvMetadata   = True
```

Figure 3.4: Example of a simple *MCS* input file with line numbers added for clarity. Any lines beginning with '#' are comments and are ignored by *MCS*.

```
1   # Specify the executable to run
2   Executable      = siesta
3   # Instruct condor to not tell us the outcome from the job by email
4   Notification     = NEVER
5   # Specify which file to use for stdin and stdout
6   Input = chlorobenzene.dat
7   Output = chlorobenzene.out
8
9   # Force overwriting when uploading / downloading files
10  SForce     = true
11
12  # Specify an SRB directory to get the relevant executable from
13  pathToExe        = /home/codes.eminerals/siesta/
14  # Specify a list of machines that we are happy to submit to
15  preferredMachineList = lake.bath.ac.uk lake.esc.cam.ac.uk lake.geol.ucl.ac.uk
16  # Specify the type of machine to be submitted to
17  # (throughput for a condor pool and performance for a cluster)
18  jobType          = performance
19  # Specify how many processors to use on the remote machine
20  numOfProcs       = 1
21
22  # Specify a metadata study to create a dataset within
23  RStudyId = 1010
24  # Create and name a metadata dataset to contain data objects
25  RDatasetName   = "chlorobenzene on clay surface"
26
27  # Specify an SRB directory to do some transfers to / from
28  Sdir     = /home/user01.eminerals/clay_surface/
29  # Specify that we want to get every file from within this directory
30  Sget     = *
31
32  # Specify another SRB directory to do some transfers to / from
33  Sdir     = /home/user01.eminerals/chlorobenzene
34  # Specify that we want to put all local files into the specified directory
35  Sput     = *
36
37  # Create and names a metadata data object
38  Rdesc    = "chlorobenzene molecule on clay surface: first test"
39  # Specify metadata to get with Agent-x (Tied to the previous Sdir line)
40  # Get environment metadata
41  GetEnvMetadata = true
42  # Get default metadata from the specified file
43  AgentXDefault  = output.xml
44  # Get z coordinate information and store as zCoordinate in the
45  # metadata database
46  AgentX     = zCoordinate, output.xml:/molecule[1]/atom[last]/zCoordinate
47  # Get lattice vector information and store in the metadata database
48  AgentX     = latticeVectorA, output.xml:/Module/LatticeVector[1]
49  # Get the final energy from the file and store in the metadata database
50  AgentX     = finalEnergy, output.xml:/Module[last]/
    PropertyList[title='Final Energy']/Property[dictRef='siesta:Etot']
51  # Store an arbitrary string of metadata
52  MetadataString = JobContext, "Frst test of molecule height and z separation"
53
54  # Leave the code's stderr on the remote machine, to be uploaded
55  # to the SRB at job end
56  Transfer_Error = false
```

Figure 3.5: Example of a complex *MCS* input file with line numbers added for clarity. Any lines beginning with '#' are comments and are ignored by *MCS*.

how useful these tools are when performing scientific research. They also give an idea of how widely used the tools are within the *e*Minerals project at any one time. Note that the three day period where no jobs were submitted relates to a bank-holiday weekend.

3.2.7 *MCS* examples

Figures 3.4 and 3.5 show two example *MCS* input files with line numbers added for extra clarity. For those familiar with Condor, the roots of *MCS*'s input file can be seen in their

structure.

Figure 3.4 is the simplest example. Line numbers 1 to 5 give the name and location of the executable within the SRB filesystem, and the name of the files to be used for input and output. Lines 15 to 17 provide the interaction with the SRB as discussed on page 75

The other lines of interest specify details of the metadata to be collected. These lines are concerned with the interaction with the metadata database through the use of the *RCommands*. The identification number of the relevant metadata *dataset* into which *data objects* are to be stored, is passed by the '`RDatasetID`' statement on line 11. The '`Rdesc`' statement (line 20) creates a *data object* with the specified name. Its associated URL within the SRB will be automatically created by *MCS*, using the preceding *Sdir* line (line 15).

Figure 3.5 shows a more complex example, which includes the components of the simpler script of Figure 3.4. This script contains, in lines 12 to 20, parameters for the metascheduling task, including a list of specified resources to be used ('`preferredmachineList`'), the type of job ('`jobType`'), and the path to the SRB directory containing the required executable ('`pathToExe`').

This more complicated script involves the creation of a metadata *dataset*, in addition to the creation of a metadata *data object*, as shown in the preceding example. The metadata *dataset* is created by lines 23 and 25, while the *data object* is created by the combination of lines 33 and 38. Note that it is possible to create only one *dataset* per script, but multiple data objects within this dataset, one per `Sdir` statement within the script.

This example also contains, in lines 41 to 50, commands to use *AgentX* to obtain metadata from a created CML file. In this case, the study concerns an investigation of how the energy of a molecule held over a mineral surface varies with its x coordinate, and the repeat distance in the x direction is collected as metadata, stored as '*latticeVectorA*'. This particular piece of metadata collection is specified by line 48, while line 52 specifies a string of information to be stored as metadata, providing context for the submitted job.

3.2.8 A comparison of *MCS* with other tools

MCS is of course not the only job submission tool available, and as such it is important to give a comparison of *MCS* with the other tools available. Other tools include Globus, Condor, JGrid, Nimrod_G, SGS and gLite, each of which will now be discussed.

The available tools can broadly be divided into two types, those which are 'heavyweight' in nature and those that are more 'lightweight'. The heavyweight tools are those that require installation and configuration of a collection of software systems on each of the different resources to use, many of which are dedicated to grid computing. Lightweight tools on the other hand require the minimum of installation on both client and server resources. The remainder of this section will discuss tools of both types, starting with those that can be considered heavyweight, followed by those that are more lightweight in nature.

The most common tools used outside of the particle physics community are the Globus and Condor systems already discussed. These systems while being powerful, do not, in isolation, fit with the *integrated grid* idea as implemented within the *e*Minerals *minigrid*.

Globus has been found to be very complicated and cumbersome to use, which can be hidden from the user by wrapping the commands with simple to use tools. Globus does include tools to transfer data to and from the compute resource. However, these tools require intimate knowledge of all files produced as part of a simulation and do not allow for archiving, etc. as afforded by the use of the SRB.

Condor, on the other hand allows access to Globus resources, therefore providing the required authentication mechanisms. The Condor interface is also easy to learn and use. However, all data transfers to and from remote compute resources must be performed through direct use of the Globus commands, as these have not been wrapped within the Condor system. Again, these data transfers do not allow the direct archiving provided by the *e*Minerals project use of the SRB system. There are two possible methods that could be used to support this archiving with the Condor system. The first possibility is that a tool could be written which uses Globus to retrieve the files and then the Scommands to upload them. Alternatively a tool could be submitted to the remote resource which directly uploads the data to the SRB without transferring it back to the submission machine. This latter method is how *MCS* archives any created data.

An alternative grid submission system is JGrid[10] [37], which aims to make it simpler to create grid enabled GUI systems. To do this, it aims to hide the complexities involved both with the use of the popular Java Swing toolkit[11], and when providing interfaces between the created GUI and an underlying grid submission system implemented using the Jini system[12]. Unlike *MCS* and the *e*Minerals systems that provide a resource ori-

[10]http://pds.irt.vein.hu/en/jgrid/about
[11]http://java.sun.com/j2se/1.5.0/docs/guide/swing/
[12]http://www.jini.org

ented grid, the communications within the Jini system are performed using web services, presenting a system oriented grid to the users.

The use of such systems would require the installation and configuration of the Jini tools on each of the different compute resources. It would also mean that any submission tools using JGrid would not be able to submit jobs to compute resources running other grid software. This is, of course, the same constraint as experienced when using the Globus system. However, Globus currently has a larger market share, meaning that any tools that can interoperate with the Globus system can make use of a larger set of systems than is currently possible using the Jini system.

JGrid provides a GUI interface. However, the use of job submission tools from the command line often fits better with the scientist's working model. This is why *MCS* and the other tools used within the *e*Minerals project, are primarily command line based. As such, JGrid is attempting to solve a problem that *MCS* does not address, although this problem can be addressed by tools which wrap *MCS* such as the parameter sweep tools described below. However, its use would still require the implementation of data management systems such as those implemented within *MCS*.

Another tool providing functionality similar to that of *MCS* is the Nimrod-G tool[13] [14]. The Nimrod-G system provides access to grid systems by interacting with compute resources running the Globus software, and is able to metaschedule across these resources. In that sense it provides very similar functionality to *MCS*.

Nimrod-G is designed to be able to perform more complicated workflows than that handled by *MCS*, enabling the user to specify the workflow for each individual job or set of jobs. It does this by providing two interfaces to the user, allowing the control of its operation. The first interface is a declarative modelling language, similar to other programming languages, which the user must use to code their workflow and the manner in which jobs are to be divided. The second interface is a web browser based portal, which removes the necessity for the user to learn the former interface's language.

As with JGrid above, neither of these interfaces fit the working practices of the *e*Minerals scientists. The first interface requires that the user learn a whole new language before they can submit a job. This does not lend itself to a usable system, and will discourage users from adopting the tool. Meanwhile, the web browser based interface means that the user cannot use their preferred command line interface.

[13] http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm

Nimrod-G addresses a broader range of problems than that experienced within the *e*Minerals project. As such, it has been designed so that the user can construct their own job submission and workflow systems, connecting different codes together as part of a single run. However, this power has come at the cost of increased complexity, making the system more difficult to use than *MCS*, which assumes the structure of the user's workflow internally, hiding the unnecessary details and complexity from the user.

Data archiving and metadata capture are also not handled by Nimrod-G. They would both require that special sub-workflows be developed, handling these functions for the user, just as they have within *MCS*. However, unlike *MCS*, the user would have to combine these sub-workflows with their own developed workflows, adding to the workload of the user. The hiding of these details is one of the strengths provided by *MCS*, allowing the user to concentrate on their research, rather than the tools they are required to use.

The last heavyweight tool that is discussed here is the gLite tool[14] [9]. gLite is the grid middleware software produced and used by the EGEE project discussed previously. As such it is the widest used of all of the tools discussed here, allowing access to approximately 40,000 processors around the world at the time of writing.

gLite aims to address many of the same issues that my tools address including job submission, metascheduling and the staging and management of data. However, gLite does not aim to integrate automated metadata collection with the rest of the job management components to the same degree that is allowed by *MCS*.

gLite is one of the largest grid middleware tools and is very heavyweight in nature, requiring considerable configuration on any resources to which access is to be given. Additionally clients can require large amounts of configuration, although lighter weight GUI based clients are available, as well as a web based portal.

It can be argued that gLite is more complicated to use than *MCS*. However, this is due to the fact that gLite is designed as a much more general purpose tool, implementing features that do not exist within *MCS*. These additional features include the ability to construct much more complicated workflows. Also fine grained control of accounting information is provided, allowing use in service policies and in charging users for consumed resources where appropriate.

The only lightweight tool that will be discussed here is called 'Styx grid services' (SGS)[15] [10]

[14] http://glite.web.cern.ch/glite/
[15] http://www.resc.rdg.ac.uk/jstyx/sgs/

and is typical of all such lightweight tools. SGS allows for a simple wrapping of the user's code. The user is able to call this wrapping, just as they have called the code in the past, but executing it on a remote grid resource. This is achieved by creating an SGS service for each combination of code to be executed, and the resource to execute it on. A wrapper is then written for use from the user's client machine, which parses any command line arguments and file redirections, passing them all to the remote grid service, where the actual code is executed.

This system is suitable for use where the code being executed hardly ever changes, and can be installed and configured on each of the available resources. However, it can not be applied to a project like *e*Minerals, where the code changes frequently, and different users have different versions of the same code. This is of course allowed by *MCS*, which simply requires the user to upload their new executable to the SRB and then to specify the path to the new file in the *MCS* input file.

In addition, tools would still be required to handle the data and metadata archival processes implemented within *MCS*. These processes would have to be combined into a workflow with the SGS job submission, much like the workflow built into *MCS* as standard. This is a common feature of these lightweight tools, and shows that for them to be useful in contexts such as the *e*Minerals project, they must be wrapped into tools that are not so lightweight. Alternatively, the user must handle these processes by hand, in which case traditional, non-grid processes would be just as suitable.

3.3 Bringing the grid to the desktop

3.3.1 Why bring the grid to the desktop?

The main disadvantage faced by users of *MCS* is the need to have the Globus and Condor client tools installed on the machine from which they wish to submit jobs. This means that should the user wish to submit directly from their desktop computer, all of the associated tools must be installed there. For older releases of Globus, installation was a long and difficult process, although more recent versions are much simpler to install. However, as has already been mentioned, Globus can not be installed by users of the Windows operating system, meaning that they are unable to use *MCS* directly on their desktop. For users who face this problem, the solution within the *e*Minerals project has been to provide a group of submit machines, from which users can submit their jobs using *MCS*.

A coupled problem can be created by the use of firewalls, with policies that restrict the opening of the required network ports between potential submission and execution machines. Within the *e*Minerals project it has been possible to negotiate for the required firewall holes to be opened, allowing the necessary access. However, in general it is unlikely that system administrators will allow the required holes, or for submit machines to be placed outside of any institute firewall. This means that users are often unable to access these submit machines from locations from which firewall holes have not been allowed. For example, users will be unable to access their submission machines whenever they are on a sabbatical visit to another institute, or when attending a conference, etc.

This use of submit machines is therefore less than ideal. The ideal solution would be to 'bring the grid to the desktop', allowing users to submit jobs to grid resources, whatever their desktop operating system, and with the minimum of configuration necessary.

Two possible solutions to these issues within the *e*Minerals project are the *e*Minerals *Compute portal* and *RMCS*. These tools will now be discussed, including details of how they work, and the advantages they provide over direct *MCS* submission.

3.3.2 Compute portal

A small group of *e*Minerals project members, including myself, developed the first method used to bring the functionality of the *e*Minerals *minigrid* to the scientist's desktop. This was the development of a web browser based portal, allowing the user to perform all required tasks from their web browser. This portal was named the '*e*Minerals Compute Portal', since it brought the computational resources of the *integrated grid* to the user's desktop.

The *Compute portal*, which has previously been described in [68], allowed the user to perform all tasks that are required, in order to configure and run jobs using the *e*Minerals *minigrid*. It was written to be as generic as possible, with the intention that it could be useful to other projects with similar needs if it proved successful within *e*Minerals. Two screenshots from the *Compute portal* in use during development are shown in figures 3.6 and 3.7.

Development of the *Compute portal* was performed using Java for the server processes, with the user interface being written in PHP, and the separate web pages managed by PHP-Nuke[16], which is a content management system. PHP-Nuke allowed the whole

[16]http://phpnuke.org/

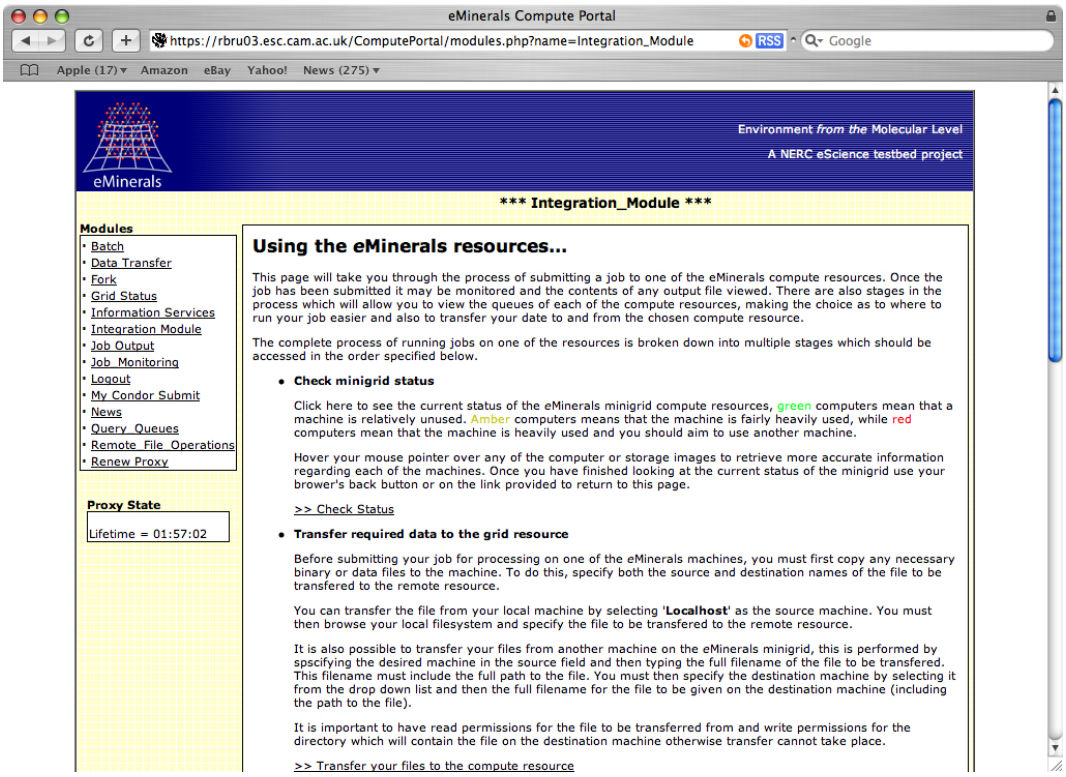

Figure 3.6: Screenshot of the *Compute portal* main menu, allowing the user to access each of the available functions. The main text shown walks the user through the job submission and monitoring process, allowing novice users to use the system without any outside tuition. The links on the left allow experienced users to immediately access any operation they desire.

portal to have a consistent look and feel, while taking charge of ensuring appropriate access to pages that should only be accessible to administrators, etc.

The *Compute portal* was designed to make use of X.509 certificates and security methods to authenticate users and to provide access to the available computational resources. A brief description of these methods and how they have been implemented within the X.509 standard will now be given, in order to show how the *Compute portal* authentication system works.

Public key infrastructure (PKI) relies on the use of pairs of public and private keys. The private key must be kept secure at all times, while the public key can be freely distributed. When given a public key, PKI methods allow the user to be certain that the person or resource being communicated with holds the related private key. This certainty is the foundation for authentication and means that the private key must be kept secret in order to prevent the key owner's identity from being stolen.

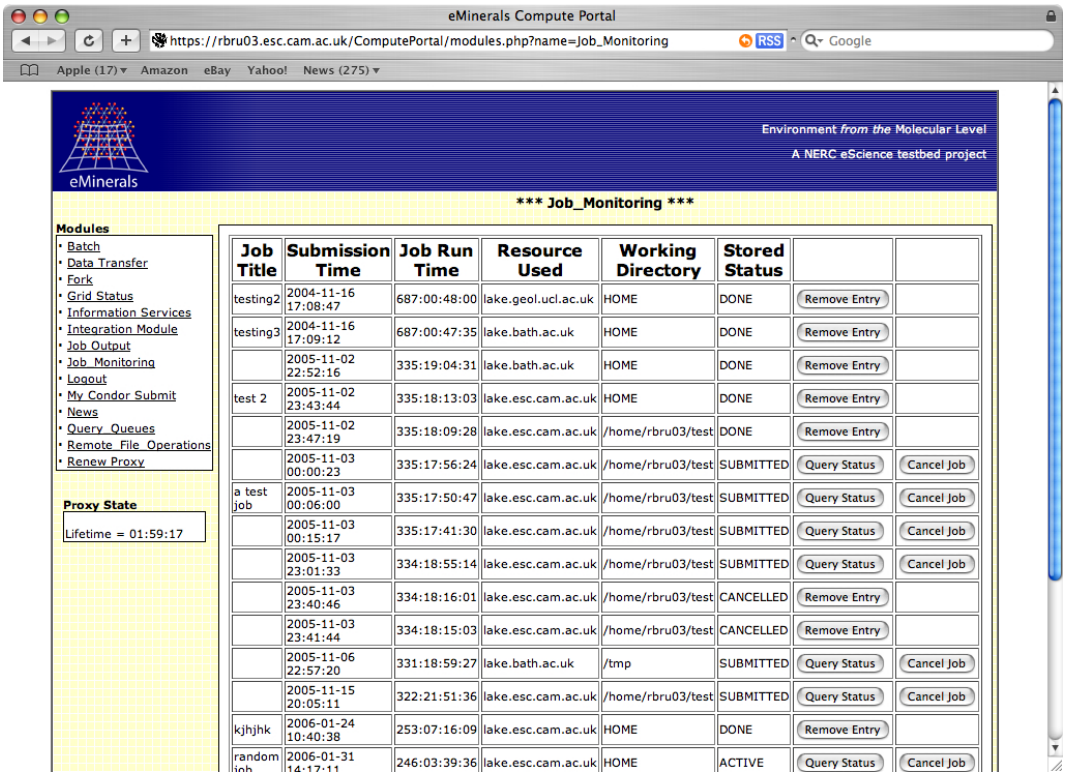

Figure 3.7: Screenshot of the *Compute portal* job monitoring system, showing a list of my submitted jobs during development. These jobs can have their individual status queried, or they can be cancelled if queueing or still running. Old, completed jobs can be removed from the list if desired, or left to provide a complete log of all of the user's jobs.

An X.509 certificate is a secure binding of a public key and a distinguished name (DN). A DN is a unique string that identifies a particular person or resource. Therefore each certificate is bound to a particular person or resource and cannot be transferred.

Authentication methods have been developed so that the private section of the certificate never needs to leave the user's computer and so is kept secure. This is achieved by encrypting a known message using the private key that can only be decrypted making use of the public key. Exchanging these encrypted messages in both directions allows two communicating parties to be sure that the other holds the relevant private key and thus proves their identity.

In normal use the private keys are encrypted and protected in order to further protect them in the case that the user's machine or files are stolen. However, in the context of grid usage, the user's identity may need to be established without any user intervention. Grid middleware typically allows this by making use of proxy certificates. These proxies are not encrypted and so are often restricted to short lifetimes in order to limit the knock

on effects of the theft, or other loss of the proxy.

Additionally, usage of these certificates within a grid environment requires that user credentials, and so the ability to prove identity, are transferred to a remote resource. This allows the remote resource to act on behalf of the user. This transfer is known as delegation.

The *Compute portal* made use of the authentication methods described above to give users access. The user's credentials were then delegated to the machine running the portal. This allowed the portal to assume the identity of the user, submitting and managing their jobs on the available computational resources.

The web browser has become a very popular method for providing interfaces to remote functionality in recent years, due to the relative ease with which systems can be developed. Systems deployed via web browsers can take advantage of the fact that the main part of the GUI is provided by the browser itself and, as such, the developer need only consider the actual workings of the system, ignoring the control of GUI windows, etc. as far as possible.

The other advantage associated with all web browser based systems, is that it is simple to bring the system to the user's desktop. This is because every operating system now comes with at least one web browser installed as standard. This in turn, means that systems need not be ported to each of the different possible operating systems, as is the case with most standalone systems.

As already mentioned, the *Compute portal* was developed to include many different functions, aiming to provide for every possible *e*Minerals *minigrid* user requirement. Some of the features implemented do not seem to fit with the vision of the *integrated grid*. For example, data transfers are handled by the user directly, through the use of gridftp. This is because the portal was initially developed at an early stage of *minigrid* development, before the standardization on the use of the SRB, and before the creation of any of the project metadata tools. The features implemented within the *Compute portal* include:

- Transferring of data to and from the computational resources using gridftp;
- Monitoring of the status of computational and data resources, mirroring the grid status tools described previously in section 2.5;
- Specification and submission of jobs using any *minigrid* computational resource;

- Tracking of submitted jobs, allowing the user to return to any job submitted via the *Compute portal*.

It had been intended that the *Compute portal* would wrap *MCS*, allowing for all of the power of *MCS*, but without requiring the installation of each of its required systems. However, *MCS* from the command line became more popular than expected, with users making the decision that they did not want to use a web based portal.

Additionally, the *Compute portal* did not implement the multiple concurrent job submission functionality described in section 3.4. This functionality has become key to the use of systems as large as the *minigrid* and externally available resources. As such, the *Compute portal* reached the prototype stage, but was never developed into a production system. Development was ceased because the users found that the use of *MCS* from the command line satisfied their requirements, which meant that they did not want to use web based tools to perform the same functionality. This was the case even though the use of *MCS* could require some users to access remote submit machines. In addition, *RMCS*, discussed below, was developed, providing the desired functionality from the user's desktop, whilst still providing a command line interface. The *Compute portal* is therefore presented here purely for historical record.

3.3.3 Remote *MCS* (*RMCS*)

'Remote MCS' (*RMCS*) is the second method employed within the *e*Minerals project to bring the power of the *integrated grid* to the user's desktop. *RMCS* was primarily developed by Rik Tyer, within the *e*Minerals project and is described in detail in appendix D. For the purpose of this discussion, it is important to note that *RMCS* provides a simple web-service based wrapping of *MCS*.

The first advantage of *RMCS* over *MCS* is that the user does not need to install the whole suite of prerequisites that *MCS* needs. This means that *RMCS* is much easier for the typical user to install on their desktop machine. It also means that *RMCS*, unlike *MCS*, can be installed on machines running the Windows operating system.

The second advantage of *RMCS* is that it uses web services to talk between the client and server processes. This means that the client machine only requires firewall holes to a web server on the server machine, on network port 443, which every normal firewall configuration allows. Again, this is not the case with the standard *MCS*, where the actual job submission is performed by the submit machine. *RMCS*, on the other hand, only need

connect to the server processes, with the actual job submission performed from the server machine. As such, the use of *RMCS* often requires no modification of existing firewall rules for the client.

RMCS was designed to be used from a machine that does not have the Globus or Condor systems installed, whereas *MCS* was designed assuming these commands and the user's certificate are stored locally. The use of *RMCS* means that *MCS* is now called remotely from the user's submit machine, meaning that the Globus calls made by *MCS* will not be able to find the user's certificate, since they will never leave the user's desktop machine. Instead, a proxy of the certificate is uploaded to a MyProxy server. The *RMCS* server retrieves a delegation of the proxy, enabling it to submit jobs on behalf of the user.

In order for *RMCS* to work, *MCS* was required to be changed so that the Globus commands invoked by *MCS* could be explicitly directed as to the location of the delegation of the user's certificate, once retrieved from the MyProxy server. This change allowed the Globus commands to correctly find the delegated certificate, and then perform commands on the remote compute resource, submitting the user's job. Without this change, *MCS* would not have been able to find the certificate, or in turn, to decide where to run the user's job, let alone perform the actual job submission.

3.4 Moving from single to multiple job submissions

Many of the problems tackled within the *e*Minerals project are combinatorial in nature. Thus, a typical study may involve running many (up to several hundred) similar simulations concurrently, with each simulation corresponding to a different parameter. Studies of this sort are often known as '*parameter sweeps*', or '*ensemble studies*'. Parameter sweep studies of this sort are well suited to the resources available both in the *e*Minerals *minigrid*, and in other grid resources, where access is given to large numbers of moderate to high power processors.

I have developed a set of tools to make the task of setting up and running such parameter sweeps within a single study relatively easy. The user must supply a template of the simulation input file, and information regarding the values of the parameter(s) to be varied. They must then run a single command, which creates a set of directories in the SRB, one for each set of parameter values. These directories contain generated input files with the specified parameter varied. Additionally, a commensurate set of directories on the users local computer is created, containing the generated *MCS* input files. The actual

process of submitting the jobs is then performed by running a second command, which walks through each of the locally stored *MCS* input files and submits them independently from one another.

As should be clear from the discussions related to *integrated grids* in chapter 1, the essence of such grid systems is much more than just the ability to create and run many jobs concurrently. Rather, *integrated grids* must include some sort of data management and analysis capability if they are to be useful to the user in the longer term. To this end, the parameter sweep tools include commands allowing for the monitoring and resubmission of failed jobs, if required. They also include support for combining output data, automatically generating graphs from the data, allowing the user to monitor any trends.

The parameter sweep tools also include a Java GUI, from which the user can control all parameters of their sweep, generating the input files and actually submitting the created jobs, monitoring their status as they run. This GUI was developed to cater for those users who prefer to interact with software via graphical interfaces, rather than command line tools, but performs exactly the same functionality, and in fact, calls the command line tools behind the scenes. However, the *e*Minerals users found that they preferred the command line tools to the GUI and so the GUI will not be discussed any further.

To summarise, a full workflow can be constructed, starting with the parameter sweep information, and creating multiple sets of input files, submitting and executing all the jobs. Data are then returned, and finally, particular data of interest extracted, generating graphs showing trends in results from all created jobs. The details of each of the steps in this process will now be given, along with the algorithms implemented within the tools.

These tools are provided on the CD included with this thesis. They can be found in the `parameter_sweep_tools` directory. Instructions for their use are given in the README.txt file in that directory. The description of each of the commands below is accompanied by the names of the commands and where they can be found on the included CD.

3.4.1 Creating parameter sweep jobs automatically

As mentioned above, for the creation of a whole sweep of simulation input files, the user must simply provide a template input file, and must specify parameters and values to vary over. The user is presented with a directory structure within the parameter sweep tools, including directories to contain template files (`parameter_sweep_tools/templates`), ex-

ecutables (`parameter_sweep_tools/executables`), and other required files such as pseudopotential files for quantum mechanical codes (`parameter_sweep_tools/toTransfer`).

The user provided template file must include all input data as required by the individual jobs, but with the values to vary replaced by keywords, for which the tools are able to search. A configuration file (`parameter_sweep_tools/config/variable_config`) is then used to specify the keywords to search for, and the values to replace them with. The values are specified as a start value, end value, and a number of steps to change from the former to the latter. For example, to vary pressure within a sweep from 0–10 GPa in 1 GPa steps, the user would need to specify start and end values of '0' and '10' respectively, and that 11 calculations should be generated. This would create jobs with pressures 0,1,2,...,10 GPa.

The parameter sweep tools are able to handle sweeps in n-dimensions. That is, the user may vary up to 'n' different parameters, where 'n' is an integer, greater than 0. Each parameter to vary may have its own range over which to sweep, with the tools taking care of managing the appropriate value generation. When more than one parameter is varied at once, the tools will create a nested directory structure with the required *MCS* input files. Each value of a parameter will be combined with every value of every other parameter within the created sweep. This is easiest to visualise for a 2-dimension parameter sweep, where, for example, x and y position could both be varied over 10 values each. This would result in the creation of 100 jobs, with the x and y positions, once plotted, forming a grid pattern with all possible combinations of x and y considered.

Since these tools were designed to make use of the *e*Minerals *minigrid*, all input files and created data are transferred to the compute resource via the SRB. As such, the job creation command (`parameter_sweep_tools/setupJobsNDSweep.pl`) takes care of uploading everything into the SRB. The tool creates a directory on the SRB to contain all of the created jobs. Within this directory, a suitable directory structure is created with one directory per job. If the user wishes more than one parameter to be varied at the same time, then the directory structure will create a nested directory structure, grouping jobs that have equal parameters and allowing for simplified browsing by the user. This uploading of the required files to the SRB, allows *MCS* to access the files as part of executing the jobs on the various computational resources available to the user.

Once the sweep of jobs has been created, the next task is to submit them to appropriate resources. This process will now be discussed.

3.4.2 Submitting sweeps of jobs

Once all of the sweep jobs have been setup and the *MCS* input files have been created, the process of submitting jobs can be trivial. The submission process simply needs to walk over the directory structure created by the previous setup command, submitting any *MCS* input files found using the locally installed *MCS*. However, simply submitting in this manner makes it difficult to account for jobs, and to keep track of which of the created Condor jobs relates to each sweep job.

Due to this lack of accountability, the submission command within the parameter sweep tools (`parameter_sweep_tools/submitJobs.sh`) uses a less trivial method. The command monitors the return code from each invocation of *MCS*. If the *MCS* call fails, the error message is reported to the user. However, if the call succeeds and the job is submitted to the underlying Condor system, then the job's unique identifier is parsed out from the *MCS* output string. This identifier is then stored, with the path to the successfully submitted job's directory in a file, known as the '*jobList*' file, which both the user and the tools can check in order to trace individual jobs if desired.

Once the jobs have been submitted to the desired *grid* resource, the issues that the scientist must concern themselves with move from setting up and submitting jobs, to the monitoring of these jobs. This monitoring process and the commands provided to assist the user will now be described.

3.4.3 Monitoring jobs on this scale

Once automated job submission moves from a few jobs to the range allowed by the parameter sweep tools described here, manual monitoring of the jobs quickly becomes unmanageable. The running of even as few as ten jobs concurrently becomes difficult. This difficulty is due to the fact that *MCS* submits several jobs using the Condor-G system for each job submitted by the user. Therefore, for the user to be able to check that their sweep of jobs has finished they must check that none of the jobs in the Condor system's queue relate to their submitted *MCS* jobs. In addition, once the jobs are known to have completed, the SRB directory containing these jobs' files must be identified before they can be analysed.

Another problem when monitoring queued and running jobs, is seen when the user runs more than one set of jobs concurrently. This means that the user must keep track of which Condor jobs relate to which parameter sweeps if they are to be able to analyse the

produced output from a sweep as soon as it has finished. This completion time may not correspond to the completion time of other sweeps, meaning that jobs will still exist in the user's queue even though one sweep has completed, making it difficult to simply look at the Condor system's queue in order to decide when a sweep has indeed completed.

The parameter sweep tools help to combat the complications involved with submitting jobs on this scale, by providing commands to assist the user in monitoring their jobs. Firstly, a pair of commands is provided, checking that all jobs within the sweep have been successfully submitted, resubmitting them if necessary. The first command (`parameter_sweep_tools/findNotSubmitted.pl`) reads the *jobList* file described in section 3.4.2. Any job in that file, which does not have a related job identifier, has not been submitted successfully. When any such jobs are found the user is informed so that they know that the job submission needs to be attempted again. Once the user knows that there was a problem in submitting at least one of the jobs, they can run the second command (`parameter_sweep_tools/submitUnsubmittedJobs.pl`), which identifies the unsubmitted jobs, attempting to submit them again, updating the *jobList* file with job identifiers as appropriate.

Secondly, a command is provided that allows the user to check that jobs within a sweep completed 'successfully' (`parameter_sweep_tools/findNotWorked.sh`). This does not mean that the command checks that the simulation has been performed correctly, or that satisfactory results have been obtained. Rather, it walks through the related directory structure within the SRB and checks for expected output files. If the output files exist, then the job has completed 'successfully', otherwise a problem with the job is assumed, which the user must then investigate themselves.

Unfortunately, it has been proven that it is not possible to programatically decide whether a given run has completed successfully in the generic case. This can be inferred from Rice's theorem [48] below. The inferences that can be made from this theorem are why the problem has been left to the user, who can deduce whether an individual run has completed successfully, based on their knowledge of the output from the particular code in question.

Theorem 3.1 (Rice, 1953) *Any functional property of programs is undecidable.*

Once jobs have started to finish, the user's interest moves from submitting and monitoring the jobs, to analysing the generated results. My work on automating this analysis will now be detailed.

3.4.4 Post-processing job output created by parameter sweeps

Now having the tools for setting up, running, and storing the resultant data files for large numbers of jobs, the scientist faces the problem of extracting the key information from the resultant deluge of data. Although the required information will differ from one study to another, there are certain commonalities among all parameter sweep studies.

The use of CML to represent output data from simulation codes, means that tools can be created to automatically parse the output files. These tools can look for data they recognise, even when the format of the whole output file is not 'understood' by them. One such tool was developed as part of the parameter sweep tools and allows the user to have graphs automatically created, showing any trends in their data. These tools are discussed in detail in chapter 4, and have since been further developed into '*pelote*' and '*ccViz*', discussed in [77]. However, their roots in the parameter sweep tools will be described now for completeness, regarding parameter sweeps.

The graphing functionality provided with the parameter sweep tools allows the user to specify the creation of as many different graphs as they require. These graphs can be of the following types:

- Single-line plots, showing evolution of a parameter through a set of jobs;
- Multi-line plots, showing the evolution of a parameter through each of multiple sets of jobs. For example, see figure 3.8;
- Phase diagrams, showing the subset of jobs with a higher value for a parameter within each job. This is used where two or more sweeps of jobs can be combined with the same value calculated for each job. For example, the modelling of two different phases of the same mineral, see figure 3.9;

These plots are created using Scalable Vector Graphics (SVG)[17], which is another XML based language. As such, the command provided with the parameter sweep tools applies a relevant transformation written using the eXtensible Stylesheet Language for Transformations (XSLT)[18]. This transformation harvests the specified data from the produced CML output files, formatting the information into a graph for the user's viewing. However, the transformation itself is complicated and will be described in more detail in chapter 4.

[17] http://www.w3.org/Graphics/SVG/
[18] http://www.w3.org/TR/xslt

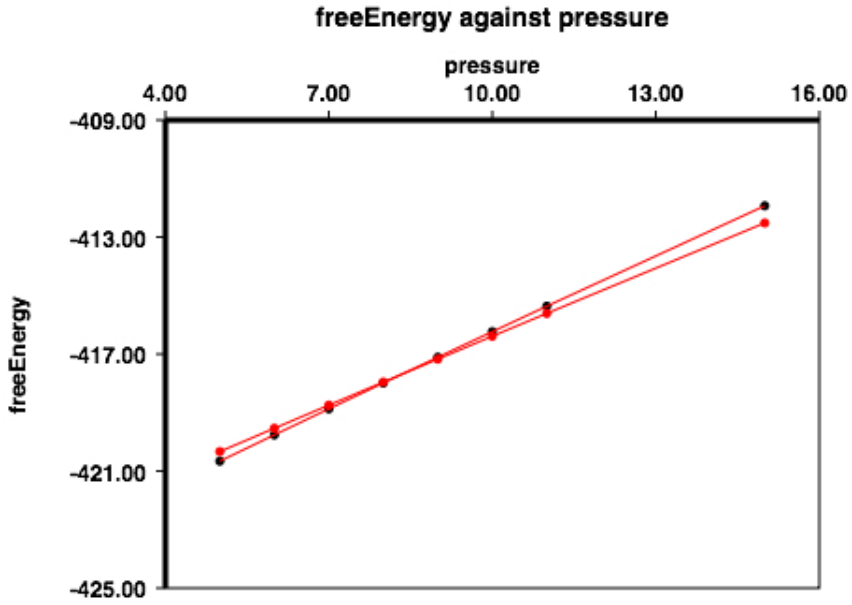

Figure 3.8: An SVG graph automatically created as part of a parameter sweep. The graph shows the different values of final energy from two sets of calculations, allowing the comparison of two phases of the same mineral. The data in red corresponds to the final energy for phase A, while that in black corresponds to the energy for phase B. The user is able to see that for this temperature, a phase transition occurs at the point where the two lines cross one another.

These created graphs are not designed to be of a quality suitable for use in research papers directly. However, they can easily be loaded into graphics programs such as Adobe Illustrator[19], and adjusted to a format and style appropriate for such use. Without modification the graphs are still accurate enough to provide the user with a view of any trends within the produced data. These trends can then be investigated further, perhaps involving the creation of an additional sweep of jobs, concentrating in more detail on an area shown as interesting within the initial graphs.

The graph creation command was developed prior to the use of the SRB for data management. As such, all data was stored locally and the command simply traversed the local directories, identifying data to be plotted. The command was designed such that it would create graphs from all available data, allowing the user to start analysing their data whilst they are still being generated. Any job that had completed, and so returned its output file to the submission machine, was analysed and plotted, whilst other jobs that were still running, and so had not returned their output files, were ignored.

[19]http://www.adobe.com/products/illustrator/

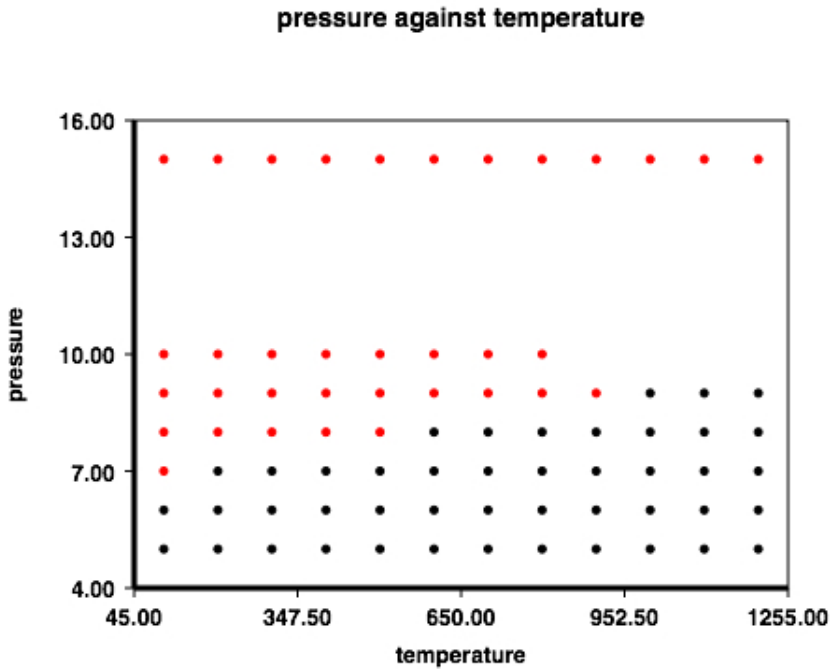

Figure 3.9: An SVG phase diagram, automatically created as part of a parameter sweep. The red dots represent combinations of temperature and pressure at which phase A is the more stable phase of the mineral. Conversely, the black dots represent combinations leading to phase B being more stable. The user might now concentrate an additional parameter sweep around the transition between phases A and B being the stable phase, in order to consider in more detail exactly when the phase transition occurs.

Figures 3.8 and 3.9 show two of the different types of graph created by these tools. The first figure relates to a multi-line graph, showing the evolution of energy within a parameter sweep, where the pressure was varied for two different phases of the same mineral. The latter figure illustrates a simple phase diagram, where two different phases of a mineral are compared, showing that which is more stable for any particular combination of pressure and temperature. The data in both figures are incomplete, because the graphs were created whilst jobs were actually still running.

Once the user has reached this stage, they will need to perform more analysis on the results from the jobs submitted. This analysis can then be used to guide further parameter sweeps, which can be run using the same tools as have just been described.

3.5 Expanding beyond the *e*Minerals *minigrid* resources

*e*Minerals project members have found the ability to use other available grid resources a great aid to their research work, providing them with extra computational resources. This requirement for extra computational power has meant that the project submission tools have had to be adjusted, in order to ensure that they work with external resources. *MCS* was the only tool described above that had to be changed to function with these new resources, because *RMCS* and the parameter sweep tools simply rely on *MCS* to provide the actual job submission functionality.

MCS itself was developed to be as generic as possible and so only required minimal changes to allow its use with external grid resources. These changes typically related to different eccentricities related to the scheduling system running on each of the different resources. In fact, *MCS* was able to use many external grid resources straight away, without requiring any changes, including both the NGS and Camgrid resources as soon as the metadata tools were installed and suitable binary executables built.

The first set of resources requiring changes to *MCS* were the NW-Grid machines mentioned earlier. Some of the machines require various environment variables to be set when submitting jobs, instructing the scheduler as to where within the cluster to attempt to schedule the job. *MCS* was simply required to be extended, such that environment variables could be set when appropriate. These environment variables are stored in the *MCS* database, and are used automatically, as required, when submitting a job.

The other main required change to *MCS* was the incorporation of support for multiple queues on the same physical resource. Within the *e*Minerals *minigrid*, the computational resources have been configured to be as simple to use as possible. As such, the resources only have one queue each, and all *e*Minerals users are able to run on any available queue. This lack of complication related to queues meant that *MCS* was developed with the assumption that each resource would operate in this way.

When it became time to start applying *MCS* to external resources such as the NW-Grid resources, it was clear that the model of one queue per physical resource was not implemented on all grid resources. As such, *MCS* was adapted to be able to query and submit to several queues on the same machine.

The only other effort involved in configuring *MCS* to function with external resources is that appropriate commands must be installed on the resource, allowing *MCS* to query the

available queues. With PBS resources, such as those found within the *e*Minerals *minigrid*, the command is already provided by the scheduler. However, Condor based resources, such as those incorporated into the Camgrid system mentioned previously, need to use a specially written command. Similarly, the SGE resources provided by the NW-Grid, required that their machine status command was wrapped for use via *MCS*, allowing the different queues to be separated correctly. These resource status commands are not at all difficult to construct, and allow *MCS* to function correctly, metascheduling between and submitting to, all resources available to the project members.

3.6 Conclusions

The problem of accessing grid resources, without making either unreasonable demands of the scientist user, or forcing users to compromise what they can achieve in their use of grid computing, is addressed by each of the tools described in this chapter. The tools are aimed at different users, with different interface types and requirements on the user's chosen submission machine.

The different tools range from the powerful tools for single job submission (*MCS* and *RMCS*), through not so powerful tools, which can be used from anywhere (the *Compute portal*), to tools automating the whole process of creating, submitting and monitoring large sets of jobs.

Each of these tools form layers around the key *MCS* tool, which has been described in this chapter. This layering has led to each of these tools keeping usability as a key consideration, ensuring that the tools are easy to use. This ease allows the user to concentrate on their research, and simply rely on the tools to function as expected. The tools are therefore, enabling, rather than distracting for the user.

This chapter also shows how tools developed for moderate size grid systems, such as the *e*Minerals *minigrid*, can be applied to larger grid systems in a simple manner. Tools such as this, which can work on all scales of system, from the smallest cluster to the largest grid, will be very important for grid computing to achieve mainstream usage, especially tools that are as easy to use as *MCS* and *RMCS*.

The next chapter describes my work on extraction and visualisation of information from the simulation codes used within the *e*Minerals project. This work has been used, and indeed was initiated, during the development of the post-processing techniques described in section 3.4.4.

Chapter 5 will then discuss the application of the tools described in this chapter to concrete science use cases.

Chapter 4

Information extraction and visualisation

This chapter is concerned with the development and use of methods that enable information to be extracted from simulation code output. Once this information has been extracted, it can be used as metadata, or visualised for the user, allowing easier access to their data.

This work is essential for users of grid systems such as the *e*Minerals *minigrid*, because the exposure of users to more computational resources leads to the ability to perform more and more simulations. With the performance of more simulations comes the need to be able to analyse the results created. Interoperability, information retrieval, and visualisation are all very important in enabling the scientist to more easily process results, helping them to make use of systems on the scale afforded by *integrated grids*.

The first tool discussed in this chapter concerns the extraction of information and its plotting into two-dimensional x-y plots. The discussion below will first introduce information extraction, explaining why it is important, and how it differs from traditional data extraction methods. Different methods of information extraction will then be discussed as well as uses for the retrieved information. The algorithms used to plot the extracted data will then be detailed.

Another method that can be used to extract information is based around the use of ontologies. This is the method used within the metadata extraction functionality of *MCS*. Ontologies are a relatively new concept that can be difficult to understand and use. Therefore, the final piece of work discussed in this chapter is a tool designed to allow for more simple use of these ontology-based methods, '*ontologyViz*'.

4.1 Introducing information extraction and visualisation

Although information and data are often used as synonyms, they do not actually have the same meaning. In fact, two definitions for data and information given by Morville in [44] are:

> Data. A string of identified but unevaluated data.
>
> Information. Evaluated, validated, or useful data.

This definition of data is clearly cyclical, therefore a clearer definition is used within this thesis, which is '*a string of identified but unevaluated statistics or facts*'. The above definition of information is also slightly different to that used in this thesis, where it is defined as '*data with meaning, or context*', where context is defined in [57] as:

> Parts that precede or follow a word and fix its meaning.

Data, once it has lost its meaning, is often useless. For example, the value '10.3 Å' does not, in isolation, tell the reader anything. It is only once this value is associated with the property to which it applies, that it can be considered useful. For example, relating this value of 10.3 Å to the length of a particular molecule, is information, making the value meaningful.

Using this definition of information leads to the definition of information extraction used within this thesis, which is '*the process of retrieving both data and the related context*'. It should be noted that information extraction is not simply a new name for data extraction; rather it is a superset of the functionality. This difference is important, because data extracted without care and attention will quickly lose any meaning and value, resulting in large quantities of effectively meaningless data. Information extraction, on the other hand, takes care of as much of the context retention as possible, helping to ensure that data can be understood in the months and years following its creation.

An example data extraction process follows, demonstrating the drawbacks associated with such a system and highlighting the need for information extraction processes. Within this example some terms will be used that a scientist does not typically associate with the extraction of data and, as such, they will be defined here. The first of these terms is 'audit trail', which is taken to mean '*a documented path through which the related*

information can be traced, showing where this information comes from'. The second term is 'accounting process', which is '*the process performed to create the previously defined audit trail*'.

A typical data extraction method used by a computational scientist would be to use the '`grep`' command to search through all relevant data files, writing any found data into their logbook. Scientists applying a more automated process may load this data straight into a spreadsheet program, which could then be used to plot the data, attempting to find trends. However, it is not unheard of for a scientist to manually copy their data back from their logbook onto their computer, and then perform any manipulation or analysis.

This process is understandably error prone and does not provide a suitable audit trail for the created data. Whilst the errors introduced by manually copying and manipulating data are an obvious problem to the scientist, they pose a relatively low risk, since the scientist already concentrates on maintaining data integrity as part of the process. The lack of a suitable audit trail for the data, and the associated loss of the data's context is much more serious, since this is the information required by the user in order to show how their data was produced. Without this information, their data cannot be checked by others. However, this audit trail is often neglected by scientists, due to the desire to find exciting trends in their data, which leads to mundane accounting processes being ignored in favour of more interesting activities.

The challenge of maintaining the audit trail is addressed within the information extraction process by the automation of the collection and storage of context as an integrated part of extracting the data. Using systems in this way, where any extracted data are connected, via metadata, to the details of the related simulation, provides part of the required accounting information. Additional information then comes from relating any input data to previous simulations from which these data are derived. The combination of these pieces of information allows any data to be traced back to its origin. This means that, should the user need to retrace their steps, they can simply follow the trail of metadata, retrieving all of the context they require.

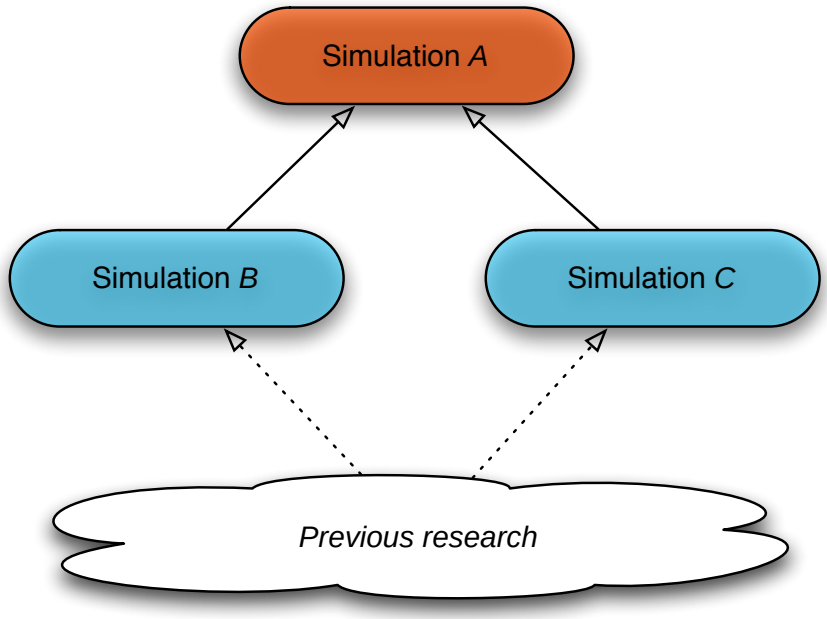

Figure 4.1: A demonstration of the audit trail for a simulation '*A*'. This simulation depends directly on simulations '*B*' and '*C*' for data. In turn these simulations depend on previous research for their data. This previous research is unidentified in this figure.

A more concrete example of this process is shown in figure 4.1. In this figure the data in question is that output from simulation *A*. This simulation requires data from two previous simulations *B* and *C*, which both model some prerequisite of the system in simulation *A*. Simulations *B* and *C* both have their own prerequisites, which are not identified in this figure, other than the unqualified connections to '*previous research*' shown at the bottom of the figure. In real usage this path could be followed, eventually leading to the initial source of the data.

An important component in this information extraction and storage process is that it should occur, as far as possible, automatically. The scientist's main aim must of course continue to be their research. Automating the extraction processes, in a manner that does not interfere with the scientist's working method, means that users can take advantage of the benefits without being distracted from this aim.

The following discussions will assume the use of XML based output file formats, since this is the format primarily used within the *e*Minerals project. However, some of the techniques can also be applied to traditional text file formats, although not in such a generic manner.

4.2 Two-dimensional *x*-*y* plots of data

4.2.1 Background

XML data formats are now very common within all forms of computerised data and information exchange. As such, most modern programming languages provide support for the handling and manipulation of these data formats, allowing the data to be read, modified and written very easily.

These languages generally implement two standard APIs allowing the programmer easy access to the data contained in an XML document. These APIs are known as the 'Document Object Model' (DOM)[1] and the 'Simple API for XML' (SAX)[2]. Both of these APIs, while allowing access to all of the data within an XML document, are aimed at use by programmers and so require tools to be written in order to make use of them. Therefore another standard was developed, XSLT, which allows for more simplified manipulation of data within XML documents. XSLT can be thought of as a `grep` for XML data, allowing the user to search through and extract data from the document. It does not require the user to program any tools other than an XSLT stylesheet. This stylesheet specifies how to locate the information required, and what to do with it once it has been found.

4.2.2 Representing two-dimensional data

Two-dimensional (2D) data can be viewed in many different ways. The two most natural methods for numerical data are tables, or 2D *x*-*y* plots. While tables are useful for representing any form of 2D data, they do not assist the reader in analysing the data held. *x*-*y* plots on the other hand are much more useful for performing this analysis, allowing the user to immediately see any trends in their data.

This usefulness of *x*-*y* plots led to the requirement for a method to draw such plots automatically from XML data. This has been tackled by the work discussed in the following sections, where XSLT is used to extract the information directly from simulation code output files, drawing plots of this information.

[1] http://www.w3.org/DOM/
[2] http://www.saxproject.org/

4.2.3 Extracting information with XSLT

These pure XML based tools involve the application of XSLT transformations to the data files, extracting the required information. These tools have the benefit that they are directly based on standards, and can be performed by any number of XML processing engines. This allows the user to read in the data file and transformation. The transformation can then be applied to the data, and the required information retrieved.

However, these tools do have disadvantages associated with them. One disadvantage is the complexity involved; there is simply too much of a learning curve. Although users may be able to create a simple script using standard Unix tools such as `grep`, the corresponding tools to extract data with XSLT would be too complicated. These methods are prone to error, and very difficult to debug, should such an error occur. Even more seriously, any small change in the data format being transformed, can cause the transformation to break, meaning that the required information can no longer be retrieved.

In addition, any change to the information required by the user, will mean that large sections of the transformation will potentially need to be rewritten. These changes relate to both the actual selection of the required data, and to its processing in the manner desired by the user. This, coupled with the fact that the typical user cannot develop or modify these tools means that developer effort is required to implement any changes the user requires, which of course is not ideal from either the perspective of the user or the developer.

It may seem that these disadvantages outweigh the benefits. However, that need not be the case for the majority of applications involving XML formatted data. In the case of the *e*Minerals project, where the XML output file formats are consistent across the simulation codes used, and where these formats only change very infrequently, the application of XML processing is very useful and easy to maintain. As such, this process is applied by the *x*-*y* plotting components of the parameter sweep tools described in section 3.4.4, and whose algorithms will be discussed in detail below. These tools show how XML based systems can be applied to the extraction of information, and its subsequent visualisation.

4.2.4 The use of visualisation within a grid environment

As has already been mentioned, expecting the user to manually work through each of their different simulation runs, visualising each output, is not practical on the scale allowed by the use of grid systems. Even when the user is only making use of a relatively modest

sized resource, such as the *e*Minerals *minigrid*, they can quickly find that the number of jobs executed, and the resultant deluge of data, becomes too much to handle manually. This means that the visualisation transformations should occur automatically, combining simulation data where appropriate.

The consideration of visualisation within parameter sweep simulations, is exactly the same as that required for the use of visualisation within grid systems as a whole. As such, the following discussion relates to all visualisation within grid systems, but with specific examples referring to parameter sweep simulations.

Within the context of parameter sweep simulations, the visualisation of job output has been integrated into the parameter sweep tools discussed previously. This integration means that as soon as the sweep of jobs has been submitted, the user can run a single command, collecting all available data within the sweep and plotting them, showing any trends. Many of the transformation tools used within the *e*Minerals project, were in fact originally developed as part of the parameter sweep tools.

Visualisation of data across whole sweeps of simulations has two uses. Firstly, it allows the user to quickly check that their simulations are performing as expected, creating sensible and feasible results. Secondly, the user is able to see trends across the whole sweep of jobs, which means that they can identify areas of interest to which future computational resources can be directed.

This final point is very important, because the use of grid systems, and the associated visualisation of whole sweeps of generated data, allow the user to consider many more data points than has been the case in the past. As an example, a scientist considering a property varying over a large range would be able to divide the problem into perhaps ten to twenty different calculations. Their conclusions must then be drawn from this small number of calculations, possibly leading to important properties being missed. Using grid technologies, the scientist would be able to divide this same problem into perhaps several hundred calculations, each executed concurrently. Automatic visualisation would then allow for the retrieval and consideration of data from all of these calculations. This means that the research performed using these technologies can be of a much higher resolution than that of research performed in a traditional manner.

Without automated visualisation tools, such as those described below, it is very difficult for the scientist to retrieve the required data from all runs within a sweep of simulations. This means that visualisation is key to the scientist's ability to cope with the data deluge, even in an *integrated grid*, where there is already so much other work dedicated to data management. The management of this deluge, is therefore key to improving the science

that can be performed by computational scientists, and can be greatly assisted through visualisation.

4.2.5 An introduction to pure XML 2D plotting

One of the most important uses for visualisation tools is to allow the user to quickly and easily see how the value of a particular property changes throughout the progression of a single, or set of related simulations. The parameter sweep tools discussed in section 3.4.4 include tools to automatically process any generated CML output files, creating SVG plots or 'graphs' of this data. The method applied and implemented by these tools will now be discussed.

These tools are based around the direct transformation from the output CML into SVG, using XSLT. There are no other tools to take an XML data file and convert it into SVG graphs. This lack of existing tools means that the transformation must take care of the complete graphing process, from extracting the data, right through to deciding upon the scale of, and actually creating appropriate axes. Finally, the actual data points and suitable lines must be placed on the graph.

These tools became very useful and were applied to the general visualisation of data created within the *e*Minerals project, in addition to the sweeps of simulations for which they were originally developed. As such, they were taken and developed further, incorporating them into the '*ccViz*' tool. The tools have also been developed into a state where they can be used as a standalone tool, called '*pelote*'. Both of these subsequent systems are described in [77].

4.2.6 Graph drawing algorithm

The graph drawing tools work by accepting from the user a term identifying the data to be drawn on the y-axis and in some cases an associated term for the x-axis data. In other cases, the x-axis data are automatically inferred from the data files being transformed, with one data point for each simulation step.

The applied transformation then processes the whole data file, finding all data identified by the term, or terms specified. Once all of the data to be graphed is found, the transformation processes them, calculating the minimum and maximum values in both the x and y directions. These values are then rounded off as appropriate, to produce the end

values for each of the two axes. These axes are then drawn, ensuring that the axes cross one another at zero, if it appears on the graph. If it does not, then the axes are placed at the end of the graph closest to zero. For example, for a graph where the y-axis only contains values between 10 and 15, the x-axis would be drawn to cross the y-axis at the bottom of the graph, corresponding to the value of 10 on the y-axis.

Once the axes themselves have been drawn, they must be labelled and divided into an appropriate scale. In order to do this the range of values on the axis is simply divided by five, and a value put on the axis at the related positions along the length. The next step taken by the transformation is to draw each of the data points on the graph. For this to happen, the coordinates of each data point are calculated, based on the coordinates of the origin of the graph. A point is then drawn at this position. Lines can also be drawn, connecting the calculated points, in the same manner as the points are drawn, should this be requested by the user.

Multiple series of data can be drawn on the graph by repeating the above procedure for each of the desired series. The transformation applies different styles to each of the series, allowing the user to identify them. As mentioned previously the created graphs can easily be loaded into a suitable graphics program and the styles changed if desired. This allows the user to adjust the generated graphs to suit their own taste. Examples of the graphs drawn using these tools can be seen in relation to the analysis of parameter sweep jobs in figures 3.8 and 3.9.

4.3 Using ontologies for information extraction

4.3.1 An introduction to ontologies

Ontologies are a relatively new concept in the field of information extraction and are currently not very common outside of information management research laboratories. They allow for the definition of the meaning associated with concepts. These concepts are then connected together to express knowledge held about them. These connections are often represented as a triple, which is a combination of a subject, predicate and object, where the subject and object are two ontological concepts and the predicate is the connection between the two. An example triple is '*a dalmation is a type of dog*', where the two concepts are '*dalmation*' and '*dog*'. A further triple might associate a '*dog*' with having '*four legs*', etc. The combination of these triples can be used to create a web of knowledge, which can then be used to apply meaning to the terms used.

Ontologies are useful because they enable the software using them to access information in files based on its meaning, rather than just its location or simple keywords. This in turn means that the user of the software is able to access the information in the files using this same knowledge, which is much more natural and intuitive than searching through files line by line for keywords, or for a comment or some other pointer.

An ontology is often represented within computing systems as a combination of two files, which are both often marked up using XML. The first file uses the Web Ontology Language (OWL)[3] to specify the different ontological concepts. The second file then uses the Resource Description Framework (RDF)[4] to define how these concepts are related to one another. Ontologies written in this manner can be very difficult to understand, especially for inexperienced readers. As such, ontology visualisation tools are often necessary, helping the user to understand what the ontology allows them to do.

4.3.2 Extracting information with *AgentX*

While ontologies do define the mapping between different concepts, they do not provide the functionality to actually follow these mappings. This means that this functionality must be provided by other tools, such as *AgentX*, which is described in appendix A. *AgentX* allows the user to navigate through XML data, based on the meaning of the data as stored in an ontology. When using *AgentX*, the user does not need to understand XML at all. In fact, all the user needs to know is the list of available concepts to choose from, and their respective meanings. With this knowledge, the user can then navigate through the logical structure of their data, extracting any information they require on the way.

AgentX provides two separate mapping functions. Firstly, it uses the ontology and XML document structure to define mappings between different concepts, allowing the user to select concepts in turn, based on these mappings. Secondly, it maps from these logical concepts onto the actual document structure. This mapping allows the user specification of a concept to be translated into an understanding of the fragment of the document that must be isolated in order to select the specified concept.

Within the *e*Minerals project, some of the mappings between concepts are implicitly stored, and are not actually represented in the ontology. In order for the user to understand and grasp these mappings, they must be visualised, showing the user what concepts

[3] http://www.w3.org/TR/owl-features/
[4] http://www.w3.org/RDF/

can be selected at each stage. This is the problem tackled by *ontologyViz*, which is discussed below.

The obvious advantage of using *AgentX* to extract information from XML based data files, is that the user does not have to be able to read the data marked up in XML format. In addition, the user need not even consider the structure of the data format. Instead, all work can be performed through the consideration of the simulation and its logical evolution. For example, to get information related to each step of a simulation, the user must simply loop through each of the steps, retrieving the data by their dictionary reference, which was explained previously in section 3.2.3. This does, of course, require that the user has a knowledge of the dictionary references used, but this can simply be provided in the simulation code's user manual.

However, the use of *AgentX* does not come without disadvantage. One such disadvantage is that it is not trivial for the user to learn and understand how it can be used. This is because the user will not have previous experience in tools of this type. However, learning and understanding *AgentX* is not insurmountable and can easily be seen to provide many benefits, encouraging the user to invest their time and effort learning. The most important aspect of the *AgentX* system that the user is required to learn, is the information held in the ontology. This is simplified through the use of *ontologyViz*, below.

The main use of *AgentX* for information extraction within the *e*Minerals project is its incorporation into *MCS*, where it underpins the metadata collection functionality. *MCS* was developed in such a way that it hides the complexity of *AgentX* usage from the user. This allows for the simple specification of a path to the required information within the user's data. This path specification has been detailed in section 3.2.3. *MCS* then takes care of the use of *AgentX*, retrieving the information required and then storing it, with context, in the *e*Minerals project metadata database.

4.3.3 Visualising ontologies

Introducing ontology visualisation

Ontologies are a completely new concept for many of the *e*Minerals project members, especially the scientist members. As such, it has been found necessary to provide the users with assistance in the understanding of how ontologies can be used. Requiring the user to read the ontology by hand is not practical, especially when you consider that users

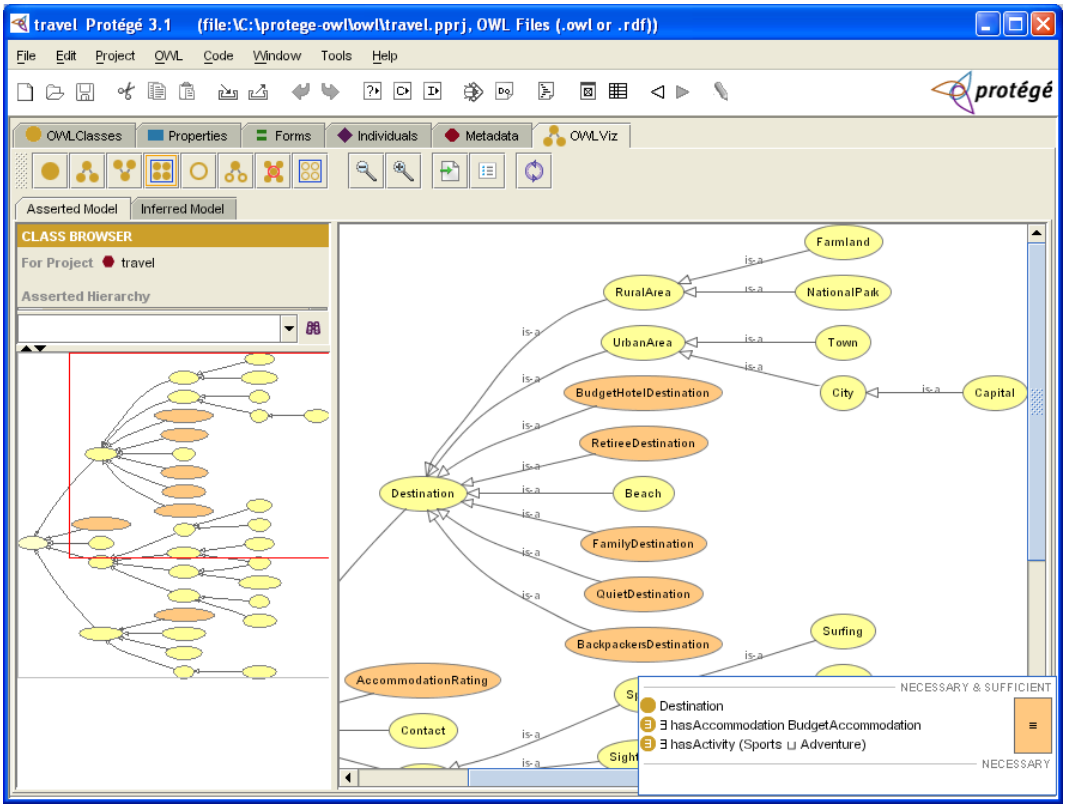

Figure 4.2: A screenshot showing a typical directed graph visualisation of an ontology, as created by the Protégé tool. This image is taken from the Protégé website.

may have no experience of XML at all, let alone the use of XML for ontology specification. Therefore, visualisation of the ontology is very important.

There are currently many ontology visualisation tools available for use, which represent the whole ontology as a directed graph, allowing the viewer to see how the different concepts defined by the ontology link to one another. One well known tool that allows for the visualisation of ontologies in this manner is the Protégé tool[5]. However, these directed graphs are often very complicated and difficult to navigate. An example ontology visualised using Protégé is shown in figure 4.2, which shows just how complicated these graphs can be.

The user specification of metadata to be collected using *AgentX* via *MCS*, relies on the user being able to understand the different concepts that *AgentX* allows to be selected at any one time. With this method of usage, the user is not required to understand the whole structure of the ontology involved. This means that the visualisation provided by

[5]http://protege.stanford.edu/

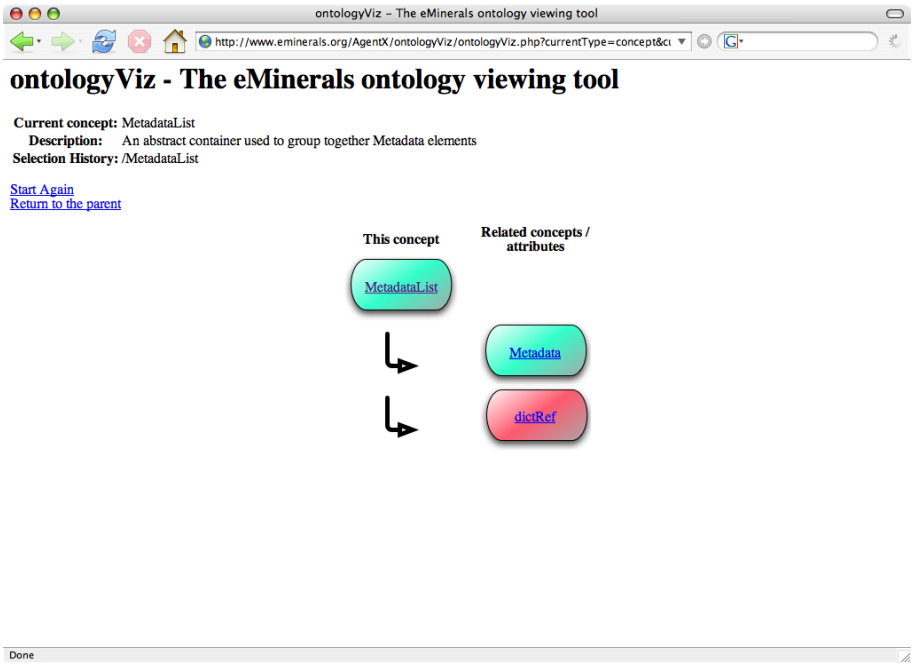

Figure 4.3: A sample *ontologyViz* screen. The user has selected the *MetadataList* concept, which is shown at the top of the page. The other objects shown are those that the user is able to select once a *MetadataList* concept has been selected. The objects in green on the right represent sub-concepts held by the current concept, whereas those in red represent attributes of the current concept.

tools such as Protégé is not really suitable for helping the user to understand how to use the ontology because it is still too complicated.

The type of visualisation that is actually required to assist the user with the use of ontologies in this manner must allow the user to walk through the ontology, selecting the required concepts in turn. No existing tools allow for visualisation of ontologies in this manner, which led to the development of a tool to provide this functionality, called *ontologyViz*, which is discussed in the following sections.

ontologyViz

ontologyViz[6] was developed so that the *e*Minerals scientists could fully grasp the details of the different available ontological concepts, without requiring them to learn several different XML technologies. *ontologyViz* helps the user to understand the *AgentX* ontology

[6] *ontologyViz* can be found and used at http://www.eminerals.org/AgentX/ontologyViz/

by allowing them to interactively navigate their way around it. An example of a typical *ontologyViz* screenshot is shown in figure 4.3.

Using *ontologyViz*, the user is able to explore the ontology, viewing the different concepts available for selection at each stage. *ontologyViz* keeps track of the different relationships between the concepts, which allows the interface to show the concepts available for use following the selection of a concept. For example, the screenshot shown in figure 4.3, shows that the user is able to select either the '*Metadata*' concept, or the '*dictRef*' attribute[7] of the currently selected '*MetadataList*' concept. A full walkthrough example, explaining in detail the usage of *ontologyViz* is given below.

At all times during the user's tour of the *AgentX* ontology several pieces of information are displayed to the user. Firstly, the list of the concepts selected in order to reach the current concept is displayed. This selection history is constructed with the same format as used by the metadata capture lines within *MCS*, meaning that the user can simply copy it into their *MCS* input file, as part of the job submission process.

Another useful piece of information presented to the user is the definition of the currently selected concept. Researchers in different areas of computational science often use the same term with different definitions. This means that it is important for the user to be able to retrieve the meaning applied by *AgentX* and its ontology, in order for them to be able to retrieve the information in which they are interested.

The *ontologyViz* interface presents the current selection at the top of the screen followed by any related concepts and attributes. Any of the presented objects can be selected by the user, which results in the redrawing of the information, with the newly selected object at the top, followed by the concepts that can now be selected. In addition, the selected object is appended to the existing selection history, allowing the user to trace the route they have followed through the ontology. Finally, the displayed definition is updated, to contain the definition of the currently selected object.

ontologyViz architecture

The *ontologyViz* system is built on a two layer architecture. The first layer is that presented to the user, which is a webpage implemented using PHP. The second layer is a database back end, which stores all of the information about the ontology for presentation

[7] As previously mentioned, the *dictRef* attribute allows for the specification of the context of a value within a CML file. For example, the SIESTA dictionary reference '`siesta:verbosity`' is used to identify the input parameter specifying whether or not SIESTA should create verbose output.

to the user. This architecture means that the contents of the database can be replaced with the details of other ontologies, allowing *ontologyViz* to be used to visualise other ontologies in the same manner.

The database contains four tables. The first two tables contain the details of the concepts and attributes that the ontology defines. The third table stores a list of relationships between any two concepts, while the fourth lists the relationships between a concept and its attributes. The interface presented to the user by *ontologyViz* accesses the database, retrieving all information related to the currently selected concept or attribute, as well as the concepts and attributes related to the current selection.

The backend is provided by the use of a database instead of using the ontology itself, because not all of the connections between the ontological concepts are actually stored within the ontology. Some of these connections are actually implicit in the structure of the CML data files, and are specified in the *e*Minerals CML standard, documented in appendix B. Therefore a representation was required that brought together all of these connections into one place. A database was chosen for this representation since databases can easily be accessed from PHP based interfaces.

This implicit storage of connections is not normally the case in systems that make use of ontologies. In fact, ontologies are normally used to represent these connections. However, the ontology used within the *e*Minerals project does not include all of these connections in order to allow *AgentX* to be more flexible in its operation when making use of the ontology.

A final important point, regarding the *ontologyViz* architecture, is that the user interface for *ontologyViz* is presented to the user via their web browser. This means that users are able to access *ontologyViz* from any computer, unlike other ontology viewers, which often require the user to install and configure them. In addition, users already understand how to use their web browser and, as such, can concentrate on the information provided by *ontologyViz*, whereas other tools will require that the user learn how to navigate the provided interface.

ontologyViz is in a state of continual development, with the ontology itself changing frequently and the definitions still gradually being collected. It is however a very useful tool for the *e*Minerals users, allowing them a much better understanding of the power afforded by *AgentX* and the metadata collection lines within *MCS* input files. The following section will provide a detailed example of the use of *ontologyViz*, highlighting the important features.

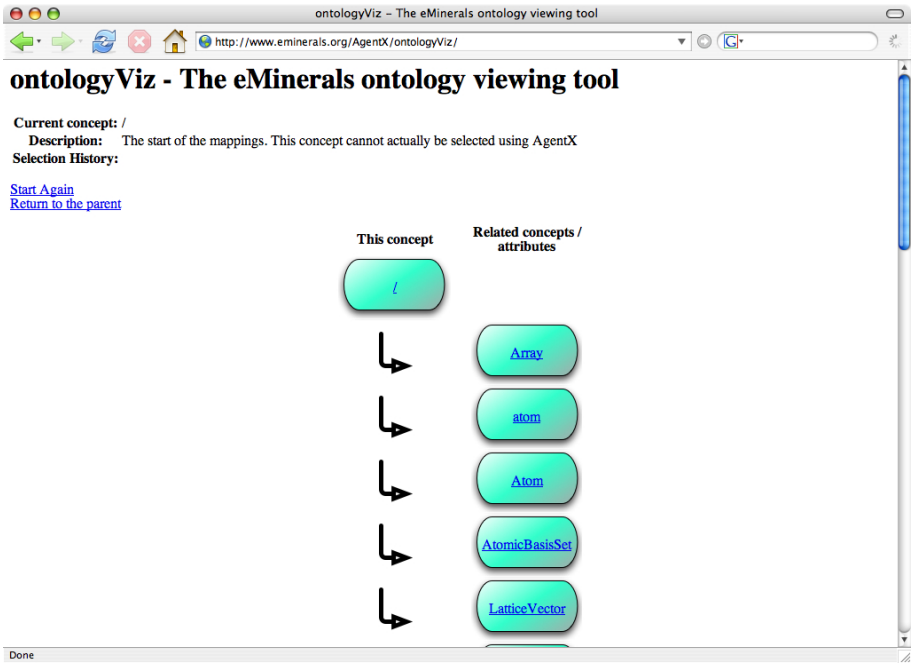

Figure 4.4: A screenshot of the initial *ontologyViz* state, showing the selected '*root*' object.

Example *ontologyViz* usage

In order to clarify the usage and power afforded by *ontologyViz*, this section will present an example of the process followed by the typical user. This process will lead to the determination of the path required when instructing *MCS* in how to retrieve some information from the output file of a SIESTA simulation. The piece of information that the example will retrieve, is the final calculated position of the first atom within a molecular system.

Initial state - When the user first accesses *ontologyViz*, they are presented with the interface shown in figure 4.4, where the '*root*' ('/') concept is selected. This is an artificial concept, which does not exist within the ontology, but is simply used to provide a point from which the user can select real concepts. The usage of the concept in this manner is explained to the user, via the description at the top the screen, which simply states the meaning as '*The start of the mappings. This concept cannot actually be selected using AgentX*'.

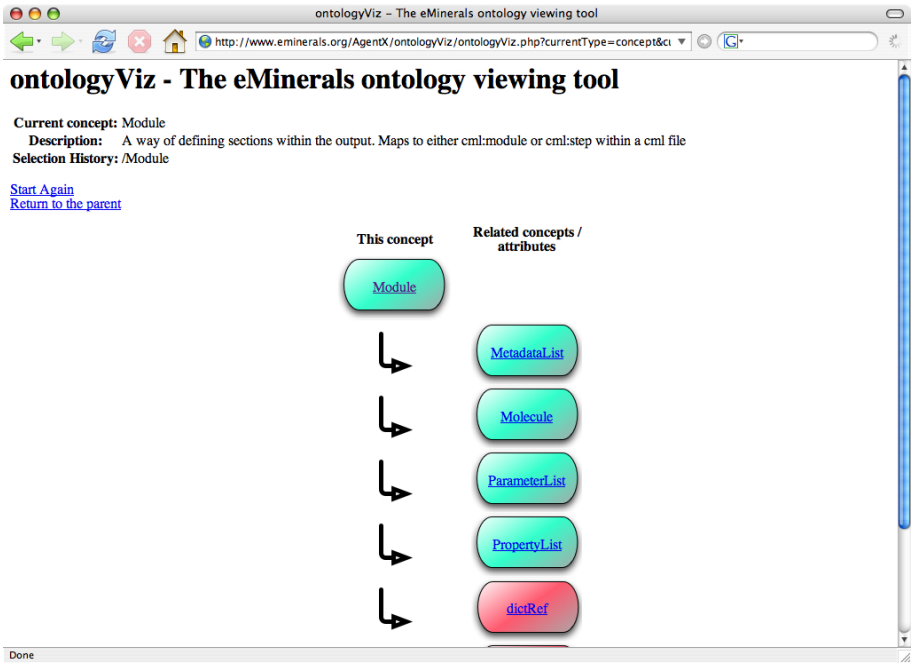

Figure 4.5: Example step 1 – screenshot of the *ontologyViz* interface, showing available concepts and attributes after the selection of the *Module* element which is displayed at the top of the screen.

Step 1 - Below the *root* element in figure 4.4, each of the concepts and attributes that can now be selected are displayed. Holding the mouse over any of these objects will display the definition applied to that object by the *AgentX* ontology. The user can then click on the desired object to select it, and cause the interface to be redrawn.

The use of *ontologyViz* in this context does require some knowledge of the structure of a CML file, because, while some concepts relate directly to typical computational chemistry terms, others relate to simulation details for which the user may not immediately understand the usage. An example of this is the '*Module*' concept, which is used to divide simulation output into individual steps. A simulation containing one hundred simulation steps, would have one *Module* for each of these steps, allowing the user to identify details related to each step.

For this example, the user is interested in details related to the final state of the system, which requires them to select the last *Module* concept. Figure 4.5 shows the redrawn *ontologyViz* interface once this concept has been selected. Features to note in the redrawn interface include the selection history, which now shows '*/Module*', and the definition of the current object, which has changed to '*A way of defining sections within the output. Maps to either cml:module or cml:step within a CML file*'.

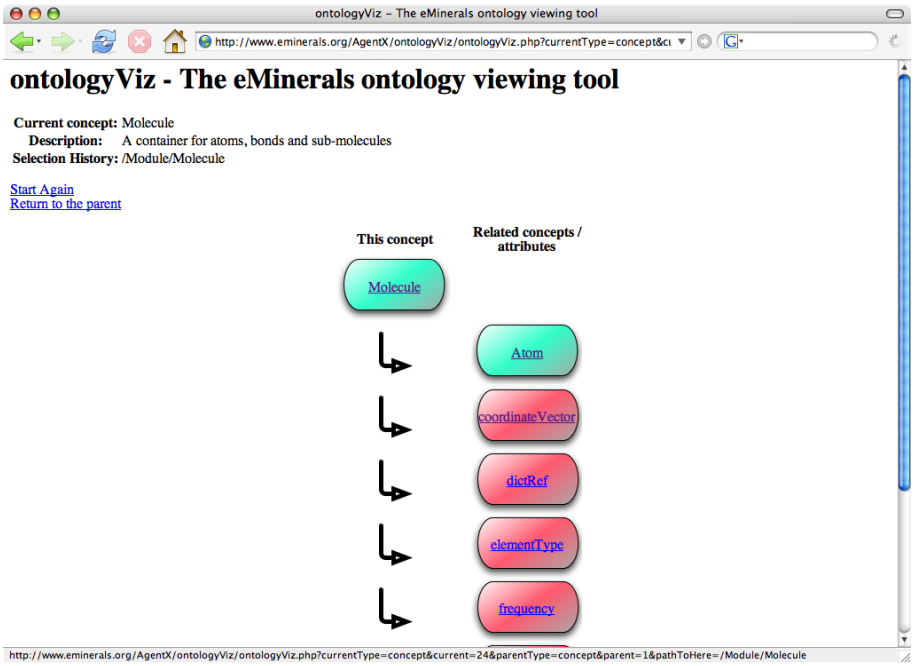

Figure 4.6: Example step 2 – screenshot of the *ontologyViz* interface, showing the selected *Molecule* concept and each of the concepts and attributes currently available for selection.

One limitation of *ontologyViz*, which is shown by this example, is that it does not allow the user to refine their selection by occurrence position when more than one object matches the specified selection. For example, it is unlikely that there would only be one *Module* in the whole CML file. When *AgentX* reads in the data it assigns an index to each of the available objects in the form of a number between 1, and the number of matched objects. Within *MCS*, the user can specify the index to be selected as either a number, or by the string '`last`'. The specified index must be wrapped with square parentheses. So to select the 10th *Module*, they would need to specify '*Module[10]*'. Alternatively, to select the last *Module* element as required for this example, they would need to change the *Module* part of their selection history to read '*Module[last]*'.

Step 2 - As shown in figure 4.5, once the *Module* concept has been selected, the user can see that the '*Molecule*' concept is available for selection. This concept, as can be seen by moving the mouse over it, is used to contain atoms and atomic bonds. This definition would imply that the user needs to select *Molecule* next, in order to retrieve the information they desire.

Selecting the *Molecule* concept leads to the interface being redrawn to look as shown in figure 4.6. Again the selection history has been adjusted and now shows '*/Module/Molecule*'.

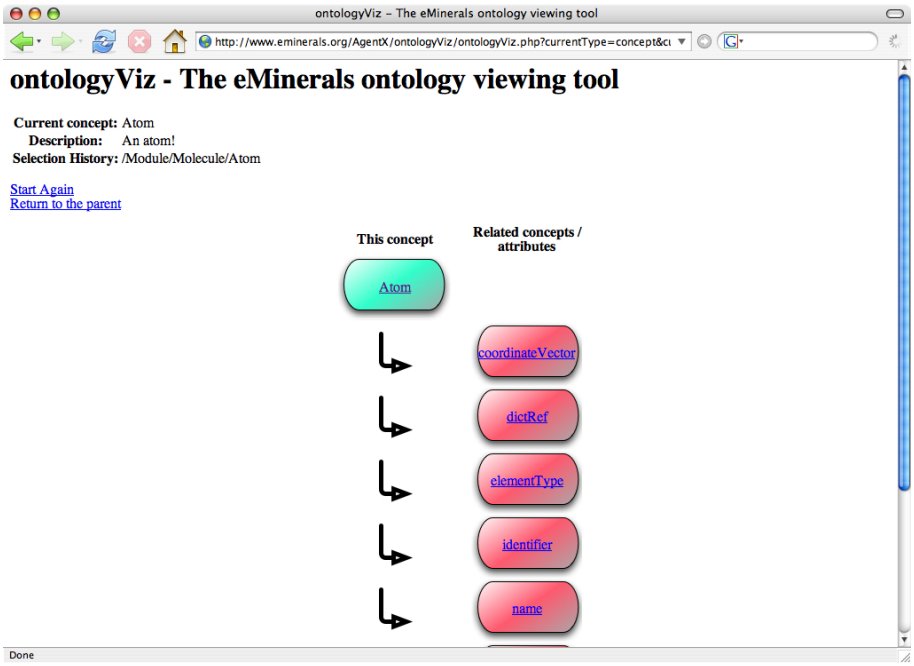

Figure 4.7: Example step 3 – screenshot of the *ontologyViz* interface, showing the selected *Atom* concept, with each of the attributes available for selection. The lack of available concepts shows that the *Atom* concept has no sub-concepts within the ontology.

In addition, the ontology's definition of the *Molecule* concept is now shown.

Step 3 - Once the *Molecule* concept has been selected, the only available object relating to the information required by the user as shown by *ontologyViz*, is the '*Atom*' concept. Therefore, for the purpose of this example, the user must select this concept in exactly the same way as with the previous steps, which leads to the screenshot shown in figure 4.7.

At this stage, the selection history, and therefore, the path to select the current element using *MCS*, is shown to be '*/Module/Molecule/Atom*'. In addition, the definition of the *Atom* concept is displayed, ensuring that the user is selecting the information they require. All attributes of the *Atom* concept that are available for selection by the user are then shown. There are no related concepts available for selection, because the ontology specifies that the *Atom* concept can have no sub-concepts.

Step 4 - The user can now perform one of several different steps, depending on the exact information required. They can select each coordinate individually, or they can select the complete vector of coordinates. The latter of which is the only option that can be seen in figure 4.7, where the '*coordinateVector*' attribute is shown. Scrolling down

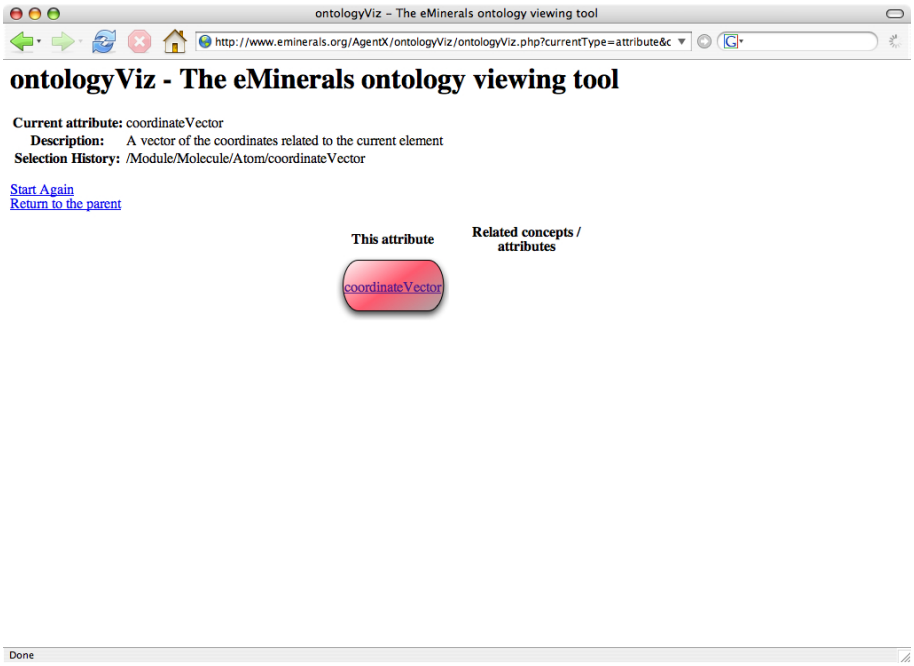

Figure 4.8: Example step 4 – screenshot of the *ontologyViz* interface, showing the selected *coordinateVector* attribute. The lack of any further concepts or attributes shows that the user cannot perform any more selections after selecting this attribute.

in the interface, presents the user with individual coordinate selection options, including '*xCoordinate*', '*yCoordinate*', and '*zCoordinate*'.

For this example, the user requires the complete coordinate vector, and so clicks on the *coordinateVector* attribute, leading to the screenshot as shown in figure 4.8. This shows that once the user has selected *coordinateVector*, they can perform no further selections. As with each of the above steps, the selection history is displayed, this time showing '*/Module/Molecule/Atom/coordinateVector*'. Also shown is the definition of the current object, which in this case is '*A vector of the coordinates related to the current element*'. This definition informs the user that using the selection path they have constructed will indeed result in the selection of the value they require.

Step 5 - Once the user has reached this stage, they can simply copy the selection history directly into an *MCS* input file. However, as part of this copying, the user must add three pieces of information that are dependent on the exact usage required by the user and so cannot be known in the generic case.

Firstly, the user must enter the name to be assigned to the selected value when it is

stored as metadata. For this example, the name given will be '`coordinates`', although in proper usage, a less generic name should be used, in order for the information to more effectively retain its context.

Secondly, the user is required to specify the name of the file to retrieve the information from. This obviously cannot be known by *ontologyViz*, as it is not concerned with actual data, only the ontology applied by *AgentX*. For this example, the filename will be '`filename.xml`', which must be followed by a '`:`' for *MCS* to correctly identify and parse the information presented to it.

The combination of these two additional pieces of information, with the selection history from *ontologyViz* will result in the following line:

```
AgentX = coordinates,filename.xml:/Module/Molecule/Atom/
         coordinateVector
```

This line is still not quite the complete statement required for *MCS* to select the information that the user requires. This is because, as explained above, *ontologyViz* is not able to handle refinements based on the index of the object to be selected. As specified at the start of this example, the user was to select the final coordinate of the atom, which relates to the last *Module* object within the output data. Therefore, the final addition to the created line is to add '`[last]`' to the *Module* selection, which results in the following valid *MCS* statement:

```
AgentX = coordinates,filename.xml:/Module[last]/Molecule/Atom/
         coordinateVector
```

This line, once entered into an *MCS* input file, will result in the collection of the coordinates of the atom, as required by the user. Therefore, this example is now completed.

4.4 Conclusions

Traditional data extraction methods do not scale to the levels required when using grid technologies. This means that information extraction and its visualisation are very important for users to be able to analyse the created data. Therefore, this chapter has described the automatic creation of graphs from both single simulation runs, and from

whole sweeps of simulations, on this scale. These tools started life as part of the parameter sweep tools described previously in chapter 3, and have since been developed into general purpose tools and integrated with the other visualisation tools developed within the *e*Minerals project.

The other main topic of this chapter has been the visualisation of ontologies using *ontologyViz*. An ontology is a complicated system, which is not easily understood by the untrained eye. As such, visualisation by tools such as *ontologyViz*, which hide the intricacies and technicalities of the ontology from the user are very important for ontologies to be used by scientists. This concealment allows the user to work with concepts and terms that they understand from their day-to-day research, with the tools considering the mapping from these terms into the actual document structure. The user should not even need to know that they are working with an ontology, which is often the case for users of the metadata collection systems implemented within *MCS*.

The following chapter will now discuss the application of the tools described in chapter 3, in accessing both the *e*Minerals *minigrid* and external resources. These tools have been used to perform real scientific research, including a number of case studies, some performed by other project members using the tools I have developed. However, the main studies described, have been performed by myself, in order to demonstrate and develop my tools.

Chapter 5

Case studies

5.1 Introduction

This chapter discusses some of the scientific research performed using the tools that have already been described in earlier chapters. The main case study to be discussed is one that I have performed myself in order to test and further develop my tools. Other case studies, performed by myself and other *e*Minerals project members, will also be briefly detailed where these studies have used different aspects of the tools from the main case study.

The main case study discussed here, as with much of the research performed within the *e*Minerals project, concerns the adsorption of pollutant molecules onto mineral surfaces. Specifically, it considers the adsorption of PCDD molecules onto a dry calcite surface.

Additionally, a study of cation ordering within the octahedral layer of a clay will be discussed. This study has been performed to show how the tools discussed in previous chapters can be easily and quickly applied to the creation, management and analysis of a typical combinatorial challenge.

Other case studies are also discussed that have made use of different aspects of the tools and systems discussed in earlier chapters. The following sections will discuss each study's scientific and computational backgrounds, and why they are important. The methods employed within these investigations will then be detailed, followed by the results obtained. Conclusions will then be drawn relating to both the scientific results and the experience of using the tools described previously.

5.2 PCDD adsorption onto calcite

5.2.1 Background

This case study concerns the adsorption of the PCDD family of pollutants onto the $(10\bar{1}4)$ surface of dry calcite. The PCDD family includes several long-lived, toxic congeners, which are very harmful to humans. Calcite is a naturally occurring mineral, which is found in large quantities in soils throughout the world. Therefore, the study of the adsorption of the PCDD family onto the surface of calcite is important in understanding the transport of these pollutants within the environment.

To understand the adsorption of pollutants onto mineral surfaces it is first important to learn the key properties of both the pollutant and the mineral involved. Therefore, this study consists of three components. The first component is the family of pollutant molecules. The second component is the calcite surface. Finally the pollutants must be combined with the surface, calculating the associated energy. Thereafter, the adsorption energy can be determined, which allows further work to calculate retardation factors[1]. These can then be used to help develop better remediation strategies for contaminated areas.

The main aim for this case study was to test the applicability of the job submission and management tools discussed in chapter 3, including both *MCS* and the parameter sweep tools. The testing of the tools using real scientific research challenges is important because it ensures that the tools can be used in these situations by other users. This cannot be thoroughly tested using smaller test cases such as those created as part of the initial tool development process.

The pollutant - Polychlorinated dibenzo-p-dioxins (PCDDs)

Polychlorinated dibenzo-p-dioxins (PCDDs, $C_{12}O_2H_xCl_{8-x}$), are a family of halogenated organic molecules, which are classed as pollutants due to their toxic nature and longevity. PCDDs, as with other pollutants, build up in the human food chain due to bioaccumulation and biomagnification. This accumulation leads to some serious health risks including skin diseases, liver damage, and an increased risk of cancer. These health risks have led to close monitoring of PCDDs and their release into the environment.

[1]The retardation factor is the factor by which the pollutant that does not adsorb onto the surface migrates through the soil compared to the migration of the pollutant that does adsorb.

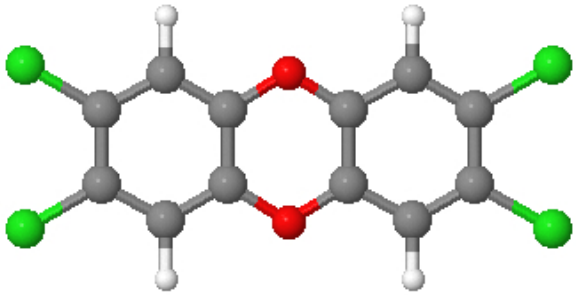

Figure 5.1: A representation of 2,3,7,8-tetrachlorodibenzo-p-dioxin, the most toxic of the PCDDs. Chlorine atoms are shown in green, hydrogen in white, oxygen in red and carbon in grey.

Large quantities of PCDDs were released before their dangers were realised, and they are still common in the environment, particularly in agricultural, industrial and brown-field sites. Sources of PCDDs include many herbicides, notably 'Agent Orange', which was used extensively by the US military during the Vietnam war. PCDDs are often found as impurities within flame retardant materials, and are released when organic material is burned in the presence of chlorine, such as in coal fired power stations, metal forges and vehicle engines.

There are 76 different PCDD congeners, with the difference between any two being the number of chlorine atoms and their location within the molecule. A typical PCDD can be seen in figure 5.1, which shows the most toxic of the PCDDs, 2,3,7,8-tetrachlorodibenzo-p-dioxin. Some stages of the work within this case study perform sweeps over all of the different congeners, while others consider only two of them.

The mineral surface - Calcite ($CaCO_3$)

Calcite ($CaCO_3$) is a very common carbonate mineral within soils, and is one of the most widely distributed minerals on the surface of the Earth. It occurs within sedimentary rocks and is commonly known as chalk, limestone or marble. The fact that calcite is so common means that any pollutant released into the environment is likely to come into contact with it. As such, the study of calcite and the associated adsorption of pollutants onto its surfaces is very important.

Calcite can present several different surfaces for adsorption, depending on its crystal

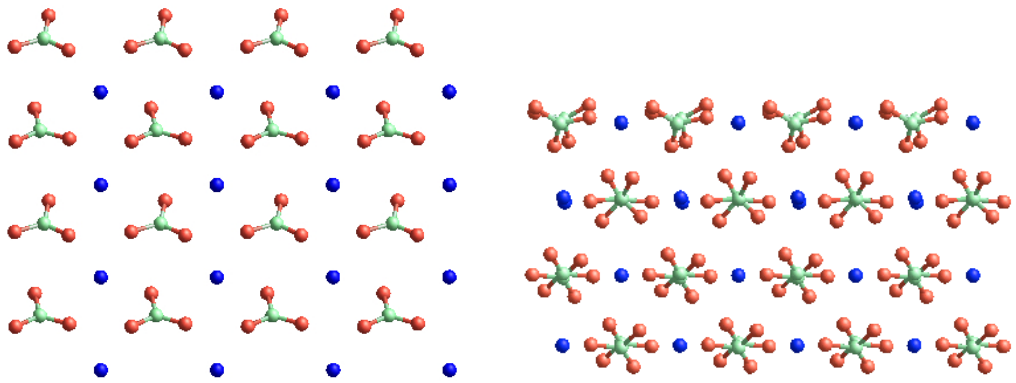

Figure 5.2: Representations of the $(10\bar{1}4)$ surface of calcite. The left-hand image shows the surface from directly above. The right-hand image shows a side-on view of the calcite slab, looking along the y-axis. Both images show the optimised surface structure as modelled within this study. Oxygen atoms are shown in red, calcium in blue and carbon in green.

structure and the orientation in which the mineral has been cleaved. The most common surface of calcite in the environment is the $(10\bar{1}4)$ surface [21]. This is the surface considered by this case study, which can be seen from above and the side, in figure 5.2.

Pollutants on mineral surfaces

The study of pollutants and minerals in isolation, while important, does not help to explain either the reason that some pollutants are found in higher concentrations in both the environment and the human food chain, or the binding of the pollutants and their movement in the environment. To be able to understand these issues, it is important to understand how strongly each of the different pollutants adsorbs onto mineral surfaces. Those pollutants that do not adsorb onto minerals may be transported within underground water flows, either making their way into plants growing in the contaminated soil, or eventually running into streams and rivers. Both of these eventualities can then lead to the pollutants entering the human food chain.

A representation of one of the PCDD congeners placed above the calcite $(10\bar{1}4)$ surface can be seen in figure 5.3. This figure shows the initial state of many of the calculations performed as part of this case study, and will be discussed in detail in section 5.2.2.

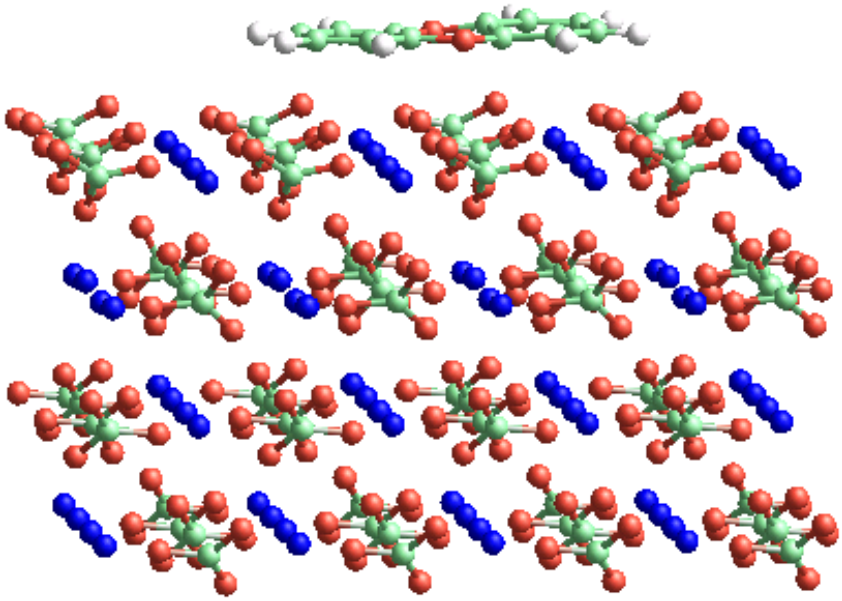

Figure 5.3: A representation of a side-on view, looking along the y-axis towards the origin, of the fully-chlorinated PCDD molecule placed above the $(10\bar{1}4)$ calcite surface.

Simulation code and method applied

All of the simulations performed for this case study used the SIESTA ab-initio simulation code. SIESTA applies DFT, as described earlier in section 1.4.1, to a molecular system. It optimises the geometry of the system, obtaining the associated energy. The DFT implementation makes use of periodic boundary conditions, in which the molecular system specified by the user is repeated to infinity in all three dimensions. Periodic boundary conditions allow each of the atoms in the initially specified system to feel the influence of those in the periodic images, preventing errors due to edge effects where the system is artificially cut off.

The first step in this study was to model a calcite unit cell, optimising its geometric configuration. The resulting optimised calcite unit cell was then used to construct a slab of calcite with the appropriate surface exposed. Next, each of the PCDD molecules were modelled in isolation, optimising their structure. This optimisation allowed for the following steps in which these molecules are placed above the calcite surface.

Two representative congeners were then scanned across the surface without any geometry optimisation and the systems' total energies calculated. This allowed the construction of a contour plot of system energy with regard to pollutant location. One contour plot was

created for each orientation of each congener considered and can be seen in figures 5.5 and 5.6 respectively. The contour plots show where the congeners should be placed above the surface to lead to the lowest system energy, which is the location at which the congeners are most likely to adsorb to the surface.

These two congeners were then modelled at their respective preferred location on the surface, optimising the geometry of the whole system. This allowed the calculation of adsorption energy for these systems. Analysis of the combined results allows for the determination of the importance of pollutant location above the surface when considering adsorption. The method used to calculate the adsorption energy, and so to determine the actual scientific results of this study, will be discussed in section 5.2.2 where this stage of the study is described.

Initially the case study was planned to fully consider all seventy-six PCDD congeners. However, the consideration of the two representative congeners took longer to simulate than first expected, which meant that there was not enough time, and associated computational capacity, to consider the other congeners to the same level of detail. The work that was performed was enough to show that the tools could successfully be applied to studies of this sort. It has also allowed for more accurate predictions of the time required to study the remainder of the simulations, which can be performed by other project members who may follow the steps determined and discussed here.

5.2.2 Performing the study

The computational challenge

Each of the different stages of this study require their simulations to be configured, managed and analysed differently from one another. However, there are some commonalities between all of the stages, which will now be discussed. The different methods applied to each stage individually will then be detailed as part of their discussion below.

All of the jobs performed for this study made use of the infrastructure built for the *e*Minerals *minigrid*. Some of the simulations were executed using computational resources within the *minigrid*, while those that required more powerful resources were executed using NW-Grid resources.

Each of the different stages of the study involve the creation of both SIESTA and *MCS* input files. Jobs created and executed individually had their input files created by hand,

while those run in sweeps had their files created automatically by tools including the parameter sweep tools described in section 3.4.

All jobs were submitted using *MCS*, which was automated by the parameter sweep tools when required. *MCS* was configured to collect both environment and default metadata, as defined in section 3.2.3, for each submitted job. It was also configured to make use of *AgentX* to capture metadata from SIESTA's output CML file, with all metadata stored in the *e*Minerals metadata database using the *RCommands*. The collected metadata includes information about the jobs' submission and execution environments, submission and completion times, the standard SIESTA metadata, and each of the input parameters with which SIESTA was invoked.

Other metadata related to the progress of the simulation was not captured by *MCS*. This was because the quantities that would be useful to check that simulations had performed as expected are not output at any single point within the simulation. Rather, the interesting feature of these quantities is their evolution throughout the simulation, which *MCS* is not able to capture. Post-processing techniques can be applied to the created data resulting in graphs like those discussed in previous chapters. However, these graphs cannot be stored by the *e*Minerals metadata systems.

The methods used to configure, submit and monitor the jobs, as well as how the created results were analysed will now be discussed for each individual step.

Dividing the case study into steps

The study of PCDD molecules and their adsorption onto the calcite surface is a complex system. As such, it must be divided into several steps in order for the different sections of the system to be investigated appropriately. These steps are:

(i) Model the calcite unit cell, comparing results with experimental data.

(ii) Construct models of slabs of calcite with different thicknesses. Compare the calculated energies and simulation time for the different numbers of layers, finding the optimum thickness.

(iii) Construct the calcite surface supercell for use in the calculation of adsorption energy, giving a surface large enough to place the pollutant congener above the surface in later steps.

(iv) Construct and optimise each of the different PCDD congeners to be used below.

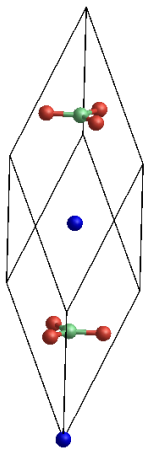

Figure 5.4: The modelled calcite unit cell.

(v) Scan two representative PCDDs over the calcite surface in two different orientations, creating energy maps for the surface.

(vi) Combine the representative PCDDs with the surface in their optimal position and optimise, calculating the surface adsorption energy.

These steps must be performed in this order, because each step depends on the outcome of that preceding it. The following sections will now describe each of these steps in detail, explaining both the computational steps taken, and the scientific basis and results for each stage.

Step (i) - Model calcite unit cell

The first stage in the modelling of calcite is the creation of a simple system representing the unit cell of the calcite crystal. The SIESTA input file was created using an approximate structure calculated by previous work discussed in [5]. The input file included commands instructing SIESTA to vary the system's lattice parameters, optimising the calcite structure. It was then combined with suitable pseudopotential files, to create a complete set of SIESTA input files. The created calcite unit cell can be seen in figure 5.4.

Each of these files was then uploaded to the SRB and a suitable *MCS* input file created. The simulation was then executed, with the end results providing optimised unit cell dimensions, the crystal structure, and the atomic forces present in the fully optimised system. Suitable metadata was collected, relating to the system's input parameters and the calculated energies, assisting with retrieval of this data at a later stage.

Property	Calculated value	Experimental value	Error (%)
Lattice parameter a	5.040 Å	4.989 Å	+1.0
Lattice parameter b	5.040 Å	4.989 Å	+1.0
Lattice parameter c	17.033 Å	17.062 Å	-0.1
Lattice parameter α	119.749°	120°	-0.2
Lattice parameter β	90.155°	90°	+0.2
Lattice parameter γ	89.845°	90°	-0.2
C–O bond length	1.303 Å	1.294 Å	+0.7
Volume	375.655 Å^2	367.8 Å^2	+2.1

Table 5.1: A comparison of the values calculated in step (i) of the case study with values published in [54].

The results were analysed to check that the simulation correctly converged by viewing the evolution of the total energy of the system. The system was found to have properly converged, providing a final, optimised structure. The atomic forces were then retrieved and were found to be small, at just 0.005 eVÅ^{-1}, which confirmed that the system had converged.

The lattice parameters a, b, c, α, β, γ, which are defined in [22] as the system's repeat distance in the a, b and c directions, and the angles α, β, γ between the directions, were compared, along with the system volume, with values published in [54], and were seen to be very close to those found experimentally. In addition, the length of the bond between the carbon and oxygen atoms, the C–O bond length, was measured. Errors were calculated to be 1% or less, for each of the different lattice parameters and the bond length, which was deemed acceptable and meant that the optimised structure could be used for later stages of the case study. The calculated and experimental lattice parameters, volume and C–O bond length are shown in table 5.1.

Step (ii) - Construct and parameterise the calcite slab structure

This step involved taking the optimised calcite unit cell resulting from the previous step and creating a slab of calcite consisting of several layers with the $(10\bar{1}4)$ surface exposed. This is necessary because modelling the PCDD congeners above a single layer of calcite will lead to incorrect behaviour within the modelled system.

The modelling of a calcite slab with too few layers will lead to mistakes for two reasons. Firstly, atoms in the calcite surface will interact with those in lower layers. Therefore, the lack of these lower layers will result in errors in the electron density of the surface

Number of layers	Number of atoms	Approximate run time (days)	Approximate surface energy (Jm^{-2})
4	40	4	0.353
5	50	7.5	0.300
8	80	12.5	0.217
Published results	N/a	N/a	0.59

Table 5.2: A comparison of the different runtimes for calcite slabs of different thicknesses with calculated approximate surface energies per formula unit. The bottom energy corresponds to that published in [20], calculated using potential methods.

and consequently in the calculated adsorption energies. Secondly, the geometry of the surface will be altered by the effects of the lower layers. The lack of these lower layers will therefore lead to the modelled surface not representing the correct geometric structure exhibited by the real surface.

The creation of a system with too many layers should also be avoided. This is because the extra layers will result in the simulation taking longer to execute, but will not lead to the results being significantly more accurate than the system with just enough layers.

The number of layers that are required is dependent on the actual materials being modelled and so cannot be predicted without performing some tests. This step must therefore discover how many layers are required to accurately model the surface, while creating simulations that do not take too long to execute. In order to find this optimum number of layers simulations were created modelling four, five and eight layers respectively. The atoms in the bottom two layers of these slabs were not allowed to move, ensuring that only one side of the slab represents the real surface, which leads to a requirement for smaller calculations.

It is also important that the vacuum gap above the surface is large enough to ensure that the molecule does not interact with the underside of the slab in its periodic image in the z-direction. This was ensured by creating a vacuum gap of approximately 25 Å, which has been found to be large enough by previous work discussed in [76].

Due to the relatively small number of jobs required for this parameterisation phase, the SIESTA input files were each created manually using the GTK Display Interface for

Structures (GDIS) program[2]. All of these files were then uploaded to the SRB, and *MCS* input files created using a previous input file as a template. The only difference between each *MCS* input file related to the directory within the SRB to be used for data transfers and archiving. Each of the simulations were then submitted for execution using the resources of the *e*Minerals *minigrid*.

The first observation made about the results of the different simulations is that they took vastly differing amounts of time to execute. Table 5.2 shows the runtimes for each of the simulations. The shortest of the simulations modelled just four layers of calcite, whilst the longest modelled eight layers. The difference in the length of these two simulations was several processor-days, which shows why this parameterisation is necessary for later runs to be feasible computationally.

Next, an approximation of the surface energy was calculated for each of the simulations, relating to the $(10\bar{1}4)$ surface of the calcite structure. This energy, γ, was calculated using:

$$\gamma = \frac{E_S - E_B}{A} \tag{5.1}$$

where, for each different slab, E_S is the total system energy, E_B is the total energy of the same number of atoms in the bulk calcite structure, and finally, A is the surface area. These calculations led to the approximate energies presented in table 5.2. These energies are approximations of the true surface energy, due to the bottom two layers of the modelled slabs not being allowed to move, as discussed above.

In addition to these calculated energies, the surface energy calculated using potential methods, as published in [20], is given for comparison. This value is quite different to those calculated by this case study, which was to be expected since the potentials used within [20] were not parameterised for surface environments. This lack of parameterisation leads to a larger distortion of the surface and in turn a higher energy than that calculated here. Therefore, the difference between results calculated here and those published in [20] are not as significant as might first be perceived.

The simulation modelling eight layers was deemed to be too expensive for further consideration of the system to be feasible, therefore it was decided that further simulations should consider one of the smaller systems. The decision was made that the small difference in the energies resulting in the other two simulations did not warrant the additional computational expense and that the simulation modelling four layers of calcite was thick

[2]GDIS, available from http://gdis.sourceforge.net, is a program used for the display and manipulation of isolated molecular, or periodic systems [51].

enough for the purposes of this study. As such, the following simulations all use a four layer thick calcite slab, representing a compromise between computational expense and the degree of accuracy of the results obtained. However, it should be noted that any future work considering this system should again consider the different number of layers in the system to assess whether the four layer system is appropriate for the research being performed, or whether a thicker slab is required.

Step (iii) - Construct complete calcite slab for use in the following simulations

Although the thickness of the surface slab was determined by the previous step, the periodic cell containing the surface used for those simulations was not large enough to accommodate the pollutant molecule in either the x or y direction. The size of the created surface supercell must be large enough to ensure that the molecule does not interact with its periodic image in both the x and y directions since this would lead to the calculation of spurious energies.

As such, the next step was to construct such a supercell from the previous four layer slab. This was performed using GDIS, which provides functionality for creating such structures.

On the basis of previous results published in [76], the decision was made to create a surface which was approximately 5 Å longer than the longest dimension of the largest PCDD congener. The largest congener is the fully chlorinated PCDD, which has a length of 10.3 Å. This led to the requirement for a surface measuring approximately 16 Å in both the x and y directions. The created supercell combined four unit cells in the x direction with two unit cells in the y direction, and measured 19.306 Å in the x direction, by 15.353 Å in the y direction.

This constructed surface was then optimised using SIESTA, resulting in the structure for use in later steps. This optimised surface can be seen in figure 5.2. It is worth noting that the creation of the calcite supercell almost directly provided the optimised structure, which shows that the smaller surface does not miss any wider ranging facets of the structure.

This step only involved a single simulation, therefore the GDIS created input file was combined with the pseudopotential files and uploaded to the SRB manually, with the *MCS* input file also created by hand. The job was then submitted using *MCS* directly and executed using the *minigrid* resources.

Step (iv) - Modelling the pollutants

This step involves the creation of molecular configurations representing each of the different congeners of the PCDD family, with one simulation created per congener. Each of the simulations were then submitted and executed in parallel to one another, optimising the structure of each molecule and ensuring that bonds between atoms in the molecule are the correct length and orientation, etc.

The creation of the different congeners is a task that cannot be performed using the previously described parameter sweep tools. This is because the creation process is very specific, requiring an in-depth knowledge of mathematics and chemistry. Generic tools cannot hope to implement these processes as a part of their standard functionality. This meant that the job creation process was not performed using my sweep tools. However, the job submission and monitoring processes were still performed by these tools.

Due to the complications involved, the actual creation of the molecular configuration of each PCDD was performed by making use of a separate tool. This tool was written for use in previous studies of the PCDD family within *e*Minerals [76]. It considers each of the different sites at which chlorine atoms can occur, creating the different configurations, taking into account symmetry effects, where certain different chlorinated sites on the molecule can be considered the same. The tool takes care of the whole simulation creation process, creating a SIESTA input file for each of the different congeners. The congeners were each created within a periodic cell, which was specified to have the same dimensions as that created for the previous step of this study, in order for the calculated energies to be comparable.

Once the required files had been uploaded to the SRB, the jobs were submitted using the parameter sweep tools described in section 3.4. Before submission, the parameter sweep tools were used to create *MCS* input files, configuring the simulations so that they were submitted to the NW-Grid resources. These resources were used because they are more powerful than any computational resource available as part of the *e*Minerals *minigrid*, and as such, would lead to simulations being performed much more quickly.

The simulation of each different PCDD congener within this step provided optimised structures. These structures were then used in later stages, where they were combined with the surface created by the previous steps, creating the complete system to be modelled.

Each output file was checked to ensure that the modelled congener was correctly optimised, and that final structures were achieved. These checks were performed by ensuring

Bond	Average calculated length (Å)	Published length (Å)	Error (%)
C–C	1.400	1.40 ± 0.003	+0.0
C–O	1.390	1.41	-1.4
C–H	1.088	1.08 ± 0.06	+0.7
C–Cl	1.723	1.70 ± 0.01	+1.4

Table 5.3: The calculated lengths of each bond within the PCDD family. Averages are for each bond in the fully-chlorinated and fully-protonated congeners. The published values are taken from [72], except for the C–O bond length, which is taken from [55].

that the simulations finished before completing the maximum number of allowed steps, as set in the SIESTA input files.

Those simulations that were found not to have optimised, or even to have not run, were resubmitted for further processing. This was done by changing the SIESTA input files so that they would continue from where the previous simulation stopped, where appropriate. They were then resubmitted using one of the parameter sweep commands discussed in section 3.4.

The average length was calculated for each of the different bond types in both the fully-protonated and the fully-chlorinated congeners. These values were then compared with lengths published in [55] and [72]. In addition, standard deviations were calculated for each of the bond lengths and found to be approximately zero. The average bond lengths can be seen in table 5.3, which shows that the calculated values were close enough to published values to continue using the modelled structures.

Step (v) - Parameterising the pollutant above the surface models

The next step taken as part of this study was to place two representative PCDD congeners above the calcite surface, in two different orientations and at many different locations. This was performed in an attempt to find a location at which these two congeners prefer to adsorb onto the surface. The existence of such a location would be important because it would significantly reduce the amount of computation required for later steps, since each congener could simply be modelled at this location. However, should no such location exist, then any further work considering the other seventy-four PCDD congeners would also be required to consider every possible position and orientation across the surface, in order to calculate accurate adsorption energies for these congeners.

The two congeners used for this stage were the fully-protonated and fully-chlorinated PCDD molecules. These two were chosen because they are at the two extremes of the PCDD series. As such, they are likely to exhibit the greatest differences in adsorption position and orientation, with all other molecules in the series exhibiting behaviour somewhere in between.

Of the many locations above the calcite surface, the one at which the PCDD molecule is most likely to adsorb is that with the lowest calculated system energy when considering the molecule above the surface. In order to find this location, the surface vectors were divided by ten in both the x and y directions. Each of the two molecules were then placed above the surface at a height of 3.0 Å above the topmost surface atom, in two different orientations at each of the 100 locations. This height was chosen after consideration of similar previous work, where this distance was found to be approximately that at which the PCDD molecules will adsorb to the surface of pyrophyllite [76]. The two different orientations were at 90° to one another, with one orientation placing the long length of the pollutant molecule along the x-axis and the other along the y-axis. In combination this involved four separate sweeps of 100 simulations, totaling 400 separate simulations performed for this step.

Each of the different simulations performed a single point calculation, which means that the simulation simply calculates the energy of the system, without attempting to optimise it. This allows the total system energy to be calculated according to the placing of the molecule at each point on the surface, for each orientation of both pollutant molecules.

These four sweeps were created by adjusting the parameter sweep tools to make use of another script, which, given two simulation output files, will combine the molecular configurations in each of the simulations to make one combined configuration. The two sets of output files used for this process were those taken from steps (iii) and (iv), above.

Once the sweep tools had been modified to use this script instead of their own parameter-varying functionality, it was simply a case of running the job creation command. This command created each of the SIESTA and *MCS* input files, uploading the SIESTA files, including the suitable pseudopotentials, to the SRB. This creation process would not have been feasible manually, due to the sheer number of files that would need to be created, which would lead to a high possibility of human errors in these files.

The job submission and monitoring commands were then used, submitting each of the simulations to be executed. These simulations each required a large amount of processing power, due to the number of atoms modelled within the system. As such, they were submitted to the NW-Grid resources, where each simulation was run on sixteen proces-

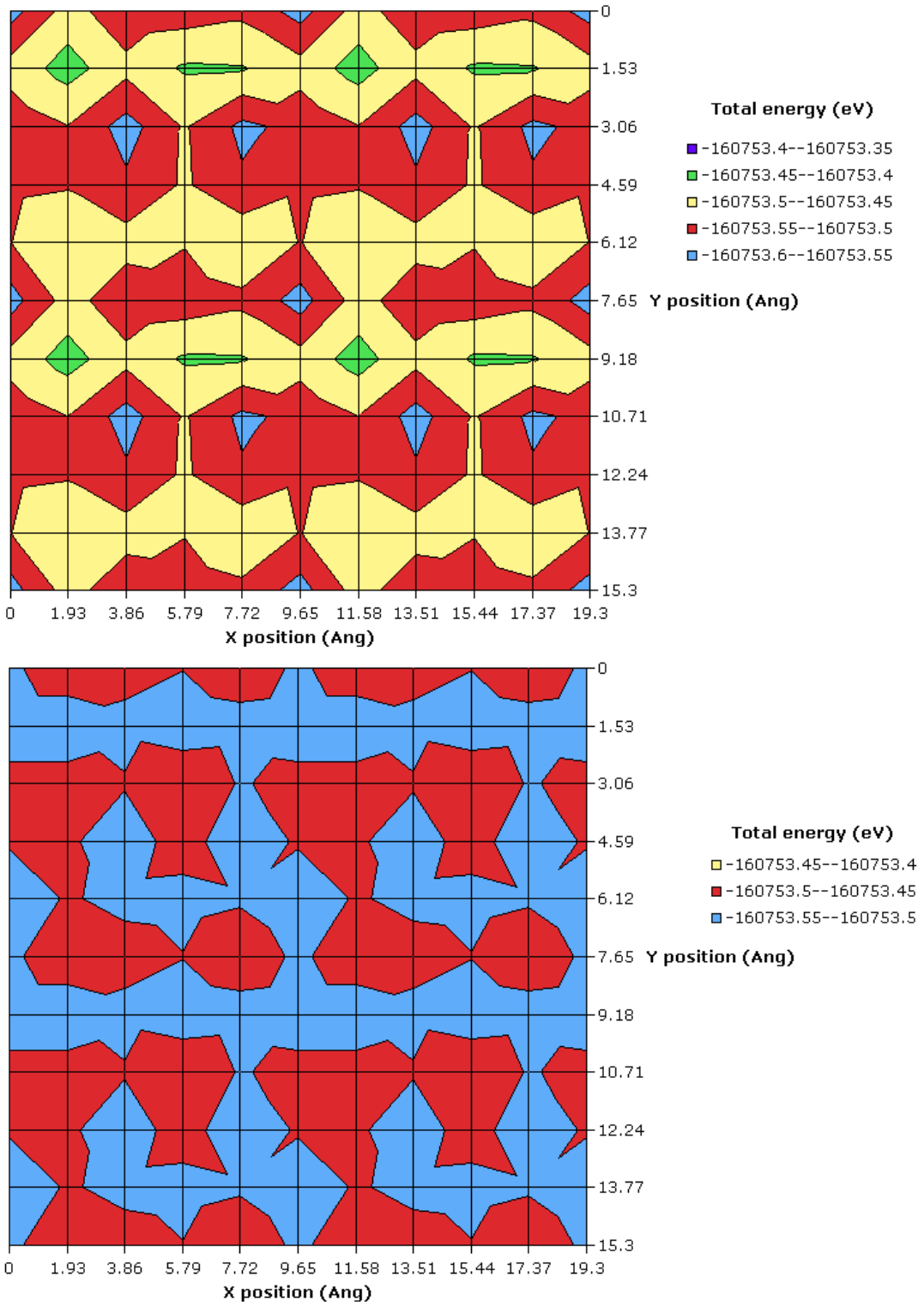

Figure 5.5: Contour plots showing the evolution of system energy calculated when scanning the fully-protonated PCDD across the calcite surface. The upper plot relates to the molecule aligned with its long axis along the x-axis of the calcite surface, while the lower plot relates to the molecule aligned along the y-axis. The x and y axes represent position across the surface in the x and y direction from the origin respectively. Each grid-point on the plots corresponds to a single simulation.

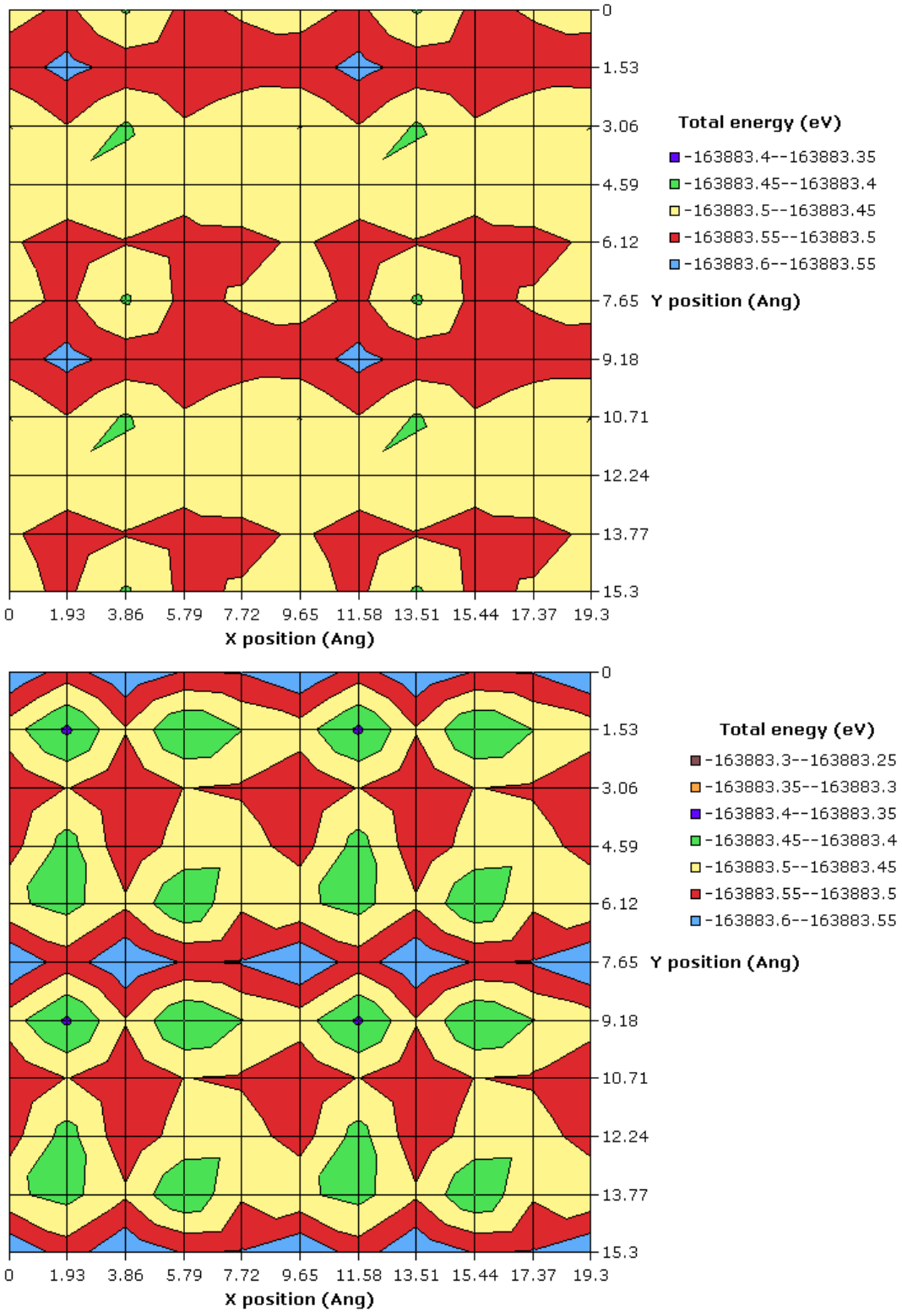

Figure 5.6: Contour plots showing the evolution of system energy calculated when scanning the fully-chlorinated PCDD across the calcite surface. The upper plot relates to the molecule aligned with its long axis along the x-axis of the calcite surface, while the lower plot relates to the molecule aligned along the y-axis. The x and y axes represent position across the surface in the x and y direction from the origin respectively. Each grid-point on the plots corresponds to a single simulation.

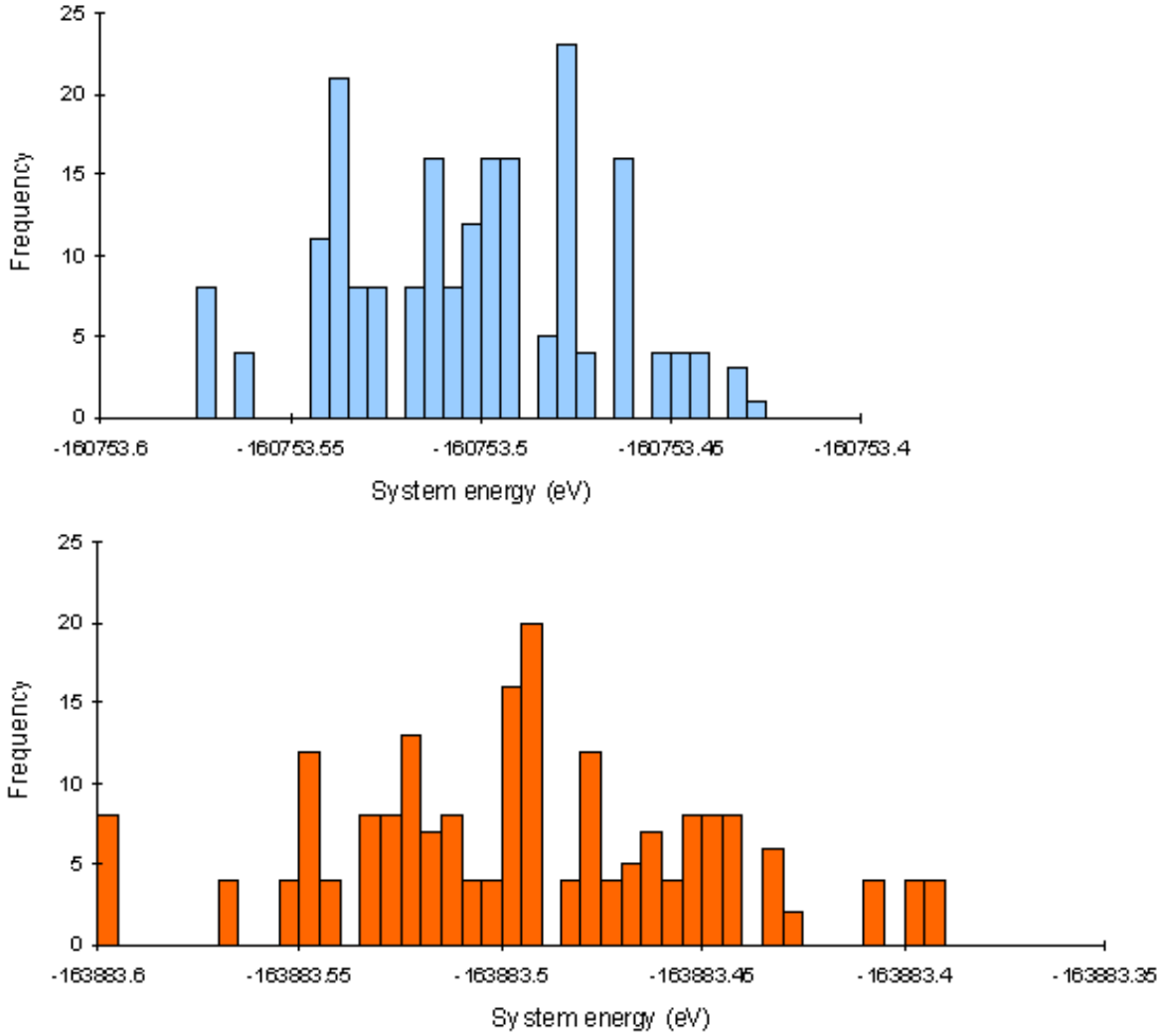

Figure 5.7: Histograms showing the distribution of system energy calculated when scanning the PCDD congeners over the calcite surface. The upper graph relates to the fully-protonated PCDD, while the lower graph relates to the fully-chlorinated PCDD. Each graph combines the two orientations of the congeners.

sors. This is equivalent to each simulation using the whole of one of the *Lake* clusters described in chapter 3. However, better performance was achieved due to faster network interconnects on the NW-Grid resources.

Once all of the simulations had completed the system energy from each run was retrieved, with each sweep of runs collated together. This data was then manipulated, creating contour plots of the system energy at each location. These plots allow trends to be seen in the calculated energies. The created plots, one for each orientation of each molecule, can be seen in figures 5.5 and 5.6. It can be seen in each of these plots that the total energy follows a repeating pattern across the surface, which is due to the repeating nature

Pollutant congener	E_{Min} (eV)	E_{Max} (eV)	$\lvert E_{Max} - E_{Min} \rvert$ (eV)	E_{Ave} (eV)	Standard deviation
0CDD	-160753.57	-160753.43	0.14	-160753.50	0.033
8CDD	-163883.60	-163883.39	0.21	-163883.49	0.046

Table 5.4: The minimum (E_{Min}) and maximum (E_{Max}) energies for each of the pollutants considered in step (v) of the case study, ignoring the orientation of the pollutant. The range of energies across the surface, which is equal to the absolute value of the difference between the minimum and maximum energies ($|E_{Max} - E_{Min}|$) is also given, allowing comparison with the results from case study step (vi). '*0CDD*' and '*8CDD*' represent the fully-protonated and fully-chlorinated PCDD respectively.

of the underlying surface structure.

In addition to these plots, the mean system energy was calculated for each molecule above the surface by averaging the total energy from each run. This is reported, with the standard deviation and energy range, in table 5.4. Each pollutant had a lowest system energy calculated at several locations above the surface. However, one location led to minimum energies for both congeners. This location is 0 Å from the origin in the x-direction and 7.65 Å in the y-direction, where the origin is shown in the top left of the surface in figure 5.2. It should be noted that the molecules were not in the same orientation when these minimum energies were found at this location.

Figure 5.7 shows two histograms of the variation of system energy for the two congeners as they were scanned across the calcite surface. One histogram was created per congener, combining the different modelled orientations. These show that the location is fairly important, because while the range of energies experienced is large they are distributed fairly evenly.

The magnitude of the numbers in table 5.4 and figures 5.5 and 5.6 only have meaning when considered relative to one another. In any computational investigation such as this, the magnitude of any number is not important. Only when considered in relation to another similarly calculated number is the value important. As such it is not the system energy that is informative, but rather the energy range arising from the translation of the PCDD molecules across the surface and the standard deviation from the average within that range.

The minimum energy value is outside of one standard deviation from the average, so at this preliminary stage it seems that the position of the molecule above the surface has significant affect on the energy of the system. Once the adsorption energy had been

obtained the significance of the position of the molecule was again considered. This reconsideration is presented below in table 5.7 and in the related discussion.

As can be seen in table 5.4, the difference between the minimum and average energies is quite a significant amount, at around 2 times the standard deviation of the results. In addition, the energy range is equal to approximately one quarter of the adsorption energies calculated in step (vi) below. These comparisons with the standard deviations and the actual energies mean that the location of the PCDD molecule above the surface is important. As has already been mentioned, the congeners modelled here were chosen because they are at the extreme ends of the PCDD series. This means that the behaviour of the other congeners is likely to be between that seen for these two. Therefore the discovery of a common location at which these two congeners have a minimum calculated energy implies that this location is also likely to lead to a minimum calculated energy for the other congeners.

Step (vi) - Combining the pollutants and the surface

As noted above, a common location was found, resulting in a minimum calculated system energy when either congener was placed there. As such, for the rest of the study the pollutants were modelled at this location. For this step of the study, the SIESTA input files from step (v) related to this location were used, with a single change instructing SIESTA to perform a full optimisation of the system.

These jobs were then submitted, again using the NW-Grid resources. Each job was executed using thirty-two processors to reduce the time needed for the runs. This would not be possible within the *e*Minerals *minigrid*, since it does not include any resources that allow parallel jobs to run on this scale.

Once the two jobs completed they were checked for convergence, ensuring that the simulations were correctly optimised, producing a stable system. The final system energy was then extracted, allowing for the calculation of the adsorption energy between the molecule and surface. This adsorption energy, E_A, is given by:

$$E_A = E_{M+S} - (E_M + E_S) \tag{5.2}$$

where, E_{M+S} is the final system energy for the simulation of the molecule above the surface, and E_S and E_M are the system energies for the surface and the molecule separately. E_{M+S} was calculated within this step, while the latter two energies were calculated by steps (iii) and (iv) respectively. The calculated energies are shown later in table 5.7,

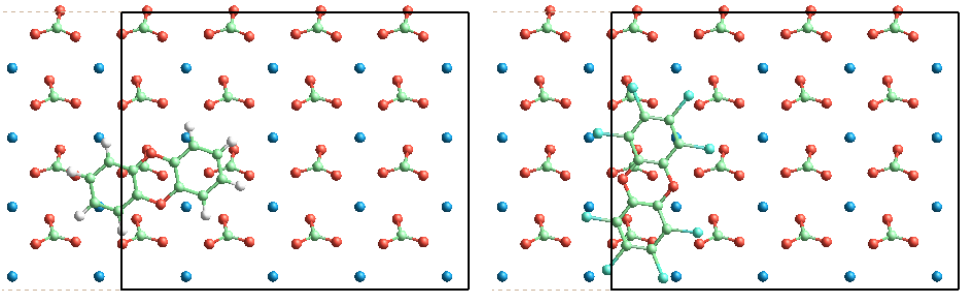

Figure 5.8: The fully-optimised PCDD congeners above the calcite surface, in the location providing the lowest system energy. Both images show a section of the neighbouring periodic image allowing the molecule to be seen completely above the modelled surface. The black square contains the modelled supercell, while those atoms outside of the square are in the periodic images. The left image represents the fully-protonated congener, while the right image represents the fully-chlorinated PCDD.

where they will also be discussed.

5.2.3 Overall results

Scientific results and future work

Figure 5.8 shows the optimised location and orientation for each PCDD congener above the calcite surface. These positions are those leading to the adsorption energies discussed below. The two congeners appear to be attracted to their respective position for slightly different reasons, which will now be discussed.

Analysis of the optimised system containing the fully-protonated PCDD congener above the calcite surface has shown that there is an attractive force between the phenyl rings' electron density and the calcium atoms in the surface. However, the rings' electron density will also try to avoid the oxygens in the surface. Therefore the molecule is oriented such that the oxygens in the surface are in the centre of the molecule's phenyl and dioxin rings where the electron density is lowest.

The Pauling electronegativity of chlorine atoms in the PCDD congener is 3.16, while for the carbon atoms it is 2.55 [1]. This higher electronegativity of chlorine compared to the carbon atoms means that they are slightly negatively charged, which will reduce the electron density of the molecule's phenyl rings. This means that it is not as important for the phenyl rings in the fully-chlorinated congener to avoid the oxygen atoms in the

C–O distance (Å)	
0CDD	8CDD
3.074	3.041
3.103	3.134
3.159	3.134
3.236	3.159
3.345	3.342
3.357	3.407
3.373	3.466
3.384	3.599
3.399	3.617
3.404	3.674
3.548	3.791
3.683	3.828
Average (Å)	
3.339	3.443

Table 5.5: The calculated distances between each carbon atom in the molecule and its nearest oxygen atom in the calcite surface, and their averages, showing the variability in distance for the two congeners. '0CDD' represents the fully-protonated PCDD, while '8CDD' represents the fully-chlorinated PCDD.

0CDD H–Ca distance (Å)	8CDD Cl–Ca distance (Å)
3.721	3.501
3.735	3.603
3.800	3.650
3.812	3.845
4.046	4.257
4.161	4.349
4.314	4.437
4.587	4.585
Average (Å)	
4.022	4.028

Table 5.6: The calculated distances between each hydrogen or chlorine atom in the molecule and its nearest calcium atom in the calcite surface, and their averages, showing the variability in distance for the two congeners. '0CDD' represents the fully-protonated PCDD, while '8CDD' represents the fully-chlorinated PCDD.

Congener	E_S (eV)	E_M (eV)	E_{M+S} (eV)	E_A (eV)
0CDD	-157866.408	-2855.677	-160722.812	-0.728
8CDD	-157866.408	-5985.627	-163852.901	-0.866

Table 5.7: Optimised adsorption energies for the fully-protonated (0CDD) and fully-chlorinated (8CDD) PCDD congeners on the calcite surface. E_S and E_M are the system energies for the surface and congener in isolation as calculated in steps (iii) and (iv) respectively. E_{M+S} is the system energy considering the congener above the surface in step (vi), and E_A is the calculated adsorption energy.

calcite surface. The slight negative charge on the chlorine atoms also means that they are attracted to the calcium atoms in the surface. This attraction is more powerful than the repulsion experienced by the phenyl rings and so dominates the positioning of the fully-chlorinated congener.

This explanation of the differences between the two congeners is supported by analysis of the distances between atoms in the molecules and the calcite surface. Table 5.5 shows the minimum distances between the carbon atoms in the molecule and an oxygen atom in the surface, while table 5.6 shows the minimum distances between hydrogen or chlorine atoms in the molecule and a calcium atom in the surface. These tables list the calculated values in order of increasing distance, along with the respective average for each congener.

Simple consideration of the average distances given in these tables is not enough to justify the positioning of the PCDD congeners, because the average distances do not vary much between the two congeners. Rather, the variation in the actual distances must be considered. It can be seen that for the distance between carbon and oxygen atoms does not vary greatly for the fully-protonated congener. This leads to a much larger minimum distance showing the repulsion between the phenyl rings and the oxygen atoms in the surface. Conversely, the variation is much greater for the distances between carbon atoms in the fully-chlorinated congener and oxygens in the surface, showing that this repulsion is less important in the positioning of this congener.

Consideration of table 5.6 and figure 5.8 show that while the chlorine atoms in the molecule are attracted to the calcium atoms in the surface it is not possible for all of the chlorine atoms to be placed above a calcium atom. This means that the congener is positioned to maximise the number of chlorine atoms that are above calcium atoms, while minimising the distance between the remaining chlorine atoms and their nearest calcium atoms.

The two congeners were allowed to move up and down above the surface as a part of the final optimisation. This resulted in the fully-protonated congener moving to 2.61 Å above the topmost atom in the calcite surface, while the fully-chlorinated congener moved to 2.64 Å above the top most atom in the calcite surface. This correlates with values obtained from previous work considering the adsorption of PCDDs onto pyrophyllite [76], where the congeners were found to adsorb at a height of 3.65 Å above the surface.

The adsorption energies calculated for the position found to have the lowest energy in step (v) are shown in table 5.7. This table also includes the calculated system energy for the congeners and surface modelled in isolation in steps (iii) and (iv), as well as the calculated energy for the system considering the congener above the surface in step (vi).

The adsorption energies were considered in relation to the results obtained from the single point calculations in step (v), the results of which are detailed in table 5.4 and figures 5.5 and 5.6. The energy range obtained from the single point calculations is approximately 25% of the adsorption energies indicating that adsorption position has a significant impact on the adsorption energy.

The difference between the two calculated adsorption energies can be attributed to the number of chlorine atoms in the two molecules. This trend was expected, since it has also been seen in previous work on PCDD adsorption onto pyrophyllite [76], which has since been verified in similar work by Martin, et al [42]. Both of these other studies found that the adsorption energy was dependent on the number of chlorine atoms in the congener, with more chlorine atoms leading to stronger adsorption energies. These studies also found that the different congeners with the same number of chlorine atoms have approximately the same adsorption energy to one another.

Additionally, the adsorption energies calculated within this work are in a similar range to those calculated by Martin, et al [42]. They calculated adorption energies ranging from approximately -1.2 eV for the fully-protonated congener through to approximately -1.45 eV for the fully-chlorinated congener. The differences between their results and those presented here are to be expected due to the different adsorption surface and also due to differences in the simulation method used.

Although the results achieved so far provide good evidence that the adsorption energy will vary with the number of chlorine atoms in the PCDD, further calculations are required in order to assert these conclusions with more conviction. However, the main aim for this study has been to test the applicability of my tools to a true scientific challenge. This aim has successfully been achieved with each of the steps required in order to setup, perform and analyse the calculated results all identified. This knowledge has now been passed

on to other *e*Minerals project members who are continuing the study, investigating the other PCDD congeners.

Grid computing results

This case study has shown that the systems and tools that I have developed can successfully be applied to a real scientific study. All of the simulations performed were submitted using *MCS*, which shows that it is a powerful tool that can be applied to real challenges.

Many of the simulations performed for the early stages of this case study were run on the *e*Minerals *minigrid*, which shows how an *integrated grid* can be applied to scientific research. Those simulations performed using resources outside of the *minigrid* still accessed these resources in exactly the same manner as the *e*Minerals machines, exemplifying the portability of the tools and showing how heterogeneous resources can be accessed in a completely uniform manner.

The choice of resources to use for each of the jobs was based on the computational requirements of the different simulations. The smaller jobs, that could be executed using a single processor, were performed using the *minigrid*, with the metascheduling functionality within *MCS* determining exactly which computational resources to use. The larger jobs, which typically required the use of more than one processor at a time, were sent to the NW-Grid resources where executables can be run in parallel more efficiently than on the *minigrid*. The particular machines used within the NW-Grid system were also chosen automatically using the *MCS* metascheduling functionality.

A little adjustment enabled the parameter sweep tools to make use of external scripts to create simulations, showing that they can be applied to very specialised simulation creation, such as the positioning of a molecule above the surface of a mineral. The tools have been shown to be generic in their job submission and management processes, with every simulation performed as part of this study managed using them.

5.3 Additional case studies

5.3.1 Modelling cation ordering in clay

I have also applied my parameter sweep tools to the modelling of cation ordering within the octahedral layer of a representative dioctahedral 2:1 phyllosilicate clay. Figures 5.9

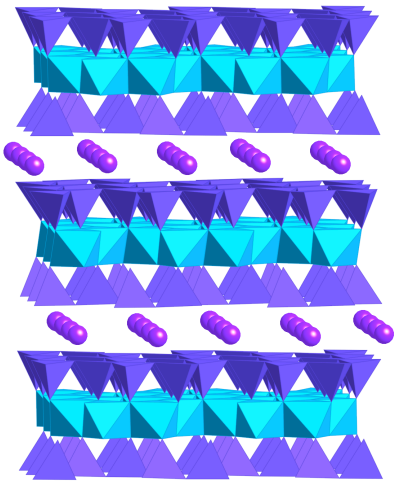

Figure 5.9: Side view of a typical dioctahedral layer silica such as the clay modelled in section 5.3.1. Layers of tetrahedra can be seen (shown in purple), separated by layers of octahedra (shown in blue), or interlayer cations (shown as spheres). The vertices of each polyhedron are oxygen atoms. The tetrahedra contain silicon or aluminium in the centre, while the octahedra contain aluminium or magnesium. This image was created by Prof. Martin Dove for use in publications related to this work.

and 5.10 show the structure of the clay being modelled from the side and above respectively. The key points in these figures are the layers of tetrahedra (shown in purple) and octahedra (shown in blue), and the interlayer cations (shown as spheres). The vertices of the polyhedra are oxygen atoms, with each oxygen in an octahedron shared between two octahedra. The tetrahedra contain silicon or aluminium in the centre, forming SiO_4 and AlO_4 units. The octahedra contain aluminium or magnesium, surrounded by six oxygen atoms. The interlayer cations are typically potassium, sodium, magnesium, or calcium. The system considered here is an octahedral layer, with 75% of the octahedra containing aluminium and 25% magnesium. Only the cations within the octahedra are actually modelled.

This project was chosen in order to demonstrate how quickly and easily the tools discussed in chapter 3 can be applied to a real scientific study, automatically performing large sections of the necessary analysis as part of the simulation submission and execution process.

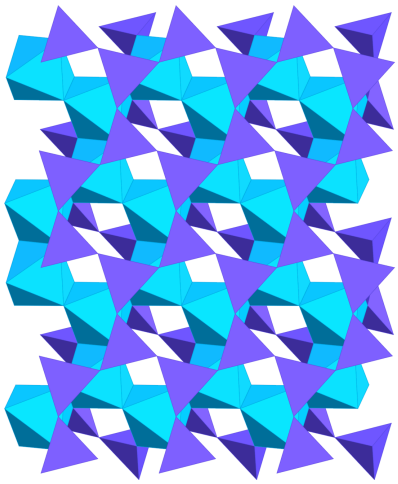

Figure 5.10: The top view of a layer of the structure shown in figure 5.9. Six-membered rings of the tetrahedra and octahedra can clearly be seen. The purple rings are made up of tetrahedra containing silicon or aluminium atoms surrounded by four oxygens. The blue rings contain aluminium or magnesium atoms surrounded by six oxygen atoms, each of which is shared between two neighbouring octahedra. This image was created by Prof. Martin Dove for use in publications related to this work.

The challenge

As temperature increases, the modelled clay is expected to undergo a phase transition, where the material becomes disordered. This change is expected because it has been seen in other clays as discussed in [47], [52] and [53]. The understanding of when and how the phase transition occurs is important but cannot be determined experimentally because the clay will decompose before the order can be measured. The phase transition can be identified computationally by measuring the level of cation ordering within the system, which is the approach taken by this study.

This study aimed to find the temperature at which the phase transition occurs (T_c), by modelling the system at a large number of different temperatures around the range at which the transition is expected to occur. The sheer number of simulations performed, each considering a different temperature, will lead to the discovery of T_c with an accuracy that could not easily be achieved using traditional methods alone without a large risk of human error.

Method applied

The Ossia[3] simulation code, which is described in [11] and [71], was used to model the octahedral layer of the clay at many different temperatures. Ossia takes a user-specified system and calculates statistics regarding the order of the system using a Monte Carlo approach.

Historically, Ossia operated by modelling a range of different temperatures in parallel to one another, using MPI to setup and complete the simulation. However, as each of the different modelled systems are completely independent from one another, they can be performed as separate simulations. This fact, and the level of processing power available from typical standalone computers, meant that Ossia was modified to perform each of these simulations individually. It was also extended to create a CML output file, which allows for easy metadata collection using *MCS* and the other tools discussed in chapter 3.

Scientific method

Prior to the simulations discussed here the octahedral layer of the material had been modelled using the SIESTA simulation code. As has already been mentioned, each of the octahedra within the layer contain either an aluminium or magnesium atom, surrounded by six oxygen atoms. SIESTA was used to calculate values for the exchange interaction between two neighbouring octahedra which both contain magnesium atoms. The exchange interaction is defined formally in [11]. However, for the purposes of this work it can be defined as the energy associated with placing the same element in two neighbouring octahehdra. A positive exchange interaction value shows that the modelled system prefers that two neighbouring octahedra do not contain the same element, while negative values show the opposite, that two neighbouring octahedra should contain the same element.

Previous studies of this type, including [47], have made use of exchange interaction values calculated using the GULP[4] simulation code [35]. This study is the first that has been performed with exchange interaction values calculated using SIESTA. SIESTA has been used in order to assess whether the exchange interaction can only be calculated using methods such as those implemented by GULP.

[3]Ossia is a Monte Carlo simulation code, which has been designed and implemented in order to study systems with atomic ordering phase transitions, solid solutions with no long-range ordering, and also non-convergent ordering processes.

[4]http://www.ivec.org/GULP/

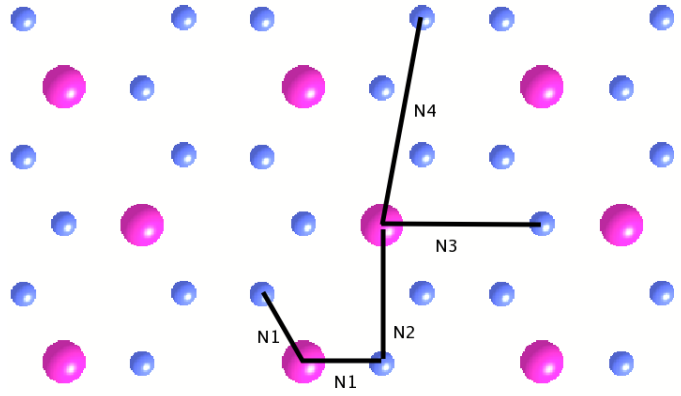

Figure 5.11: A representation of an octahedral site and its neighbours within a section of the modelled octahedral layer. Each sphere represents a distinct octahedral site, while the lines labelled Ni link a site and one of its i^{th} nearest neighbours.

Neighbour level	Exchange Interaction
1^{st} nearest	0.6146
2^{nd} nearest	0.1725
3^{rd} nearest	0.0949
4^{th} nearest	0.00278

Table 5.8: The calculated exchange interactions for the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} nearest neighbours within the octahedral layer.

Each octahedral site within the modelled structure has several neighbours. Exchange interactions were calculated for 1^{st}, 2^{nd}, 3^{rd} and 4^{th} nearest neighbours within the octahedral layer. Figure 5.11 shows a representation of an octahedral site in the modelled surface, with its 1^{st}, 2^{nd}, 3^{rd} and 4^{th} nearest neighbours identified. The calculated values for the exchange interaction for each of these neighbours can be seen in table 5.8.

As can be seen in table 5.8 the values calculated for the 1st and 2nd nearest neighbours were found to be much larger than that for the 3rd and 4th nearest neighbours. This means that it is less likely for two magnesium based octahedra to be 1st or 2nd nearest neighbours than it is for them to be 3rd or 4th nearest neighbours.

These calculated exchange interactions were then used to determine the location of magnesium within the different octahedra leading to a fully ordered system using Ossia. These locations are identified by applying the following equation:

$$E = N_1 J_1 + N_2 J_2 + N_3 J_3 + N_4 J_4 + c \tag{5.3}$$

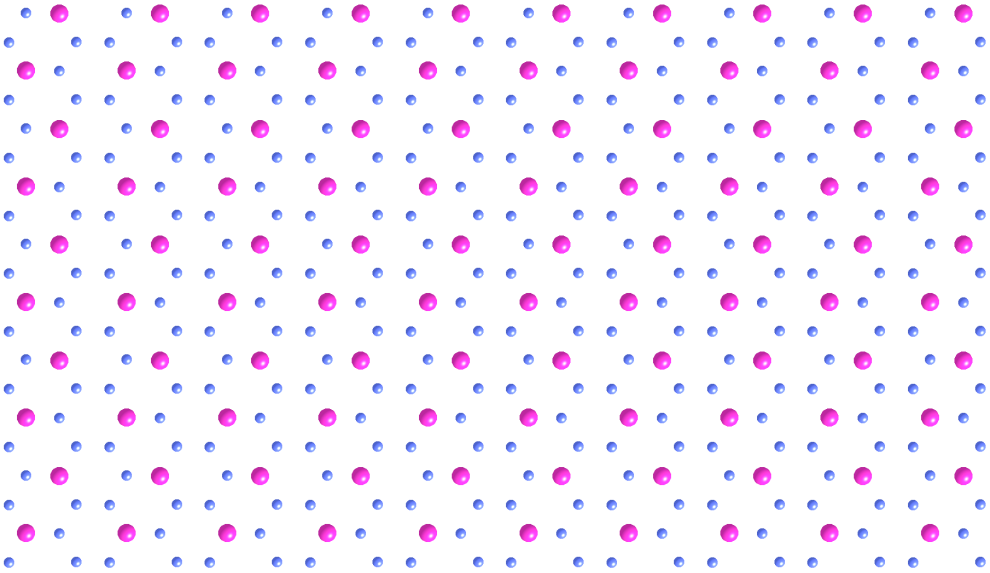

Figure 5.12: A representation of the fully-ordered octahedral layer that formed a starting configuration for each of the Ossia simulations discussed here. Each sphere corresponds to an individual octahedra in the layer. Those in pink contain magnesium, while blue spheres contain aluminium. This image was created by Prof. Martin Dove for use in publications related to this work.

where E is the total system energy, N_i is the number of octahedral sites that should have i^{th} nearest neighbours containing the same element, J_i is the calculated exchange interaction for i^{th} nearest neighbours and c is a constant. This equation is applied to a number of different atomic configurations, with that leading to the lowest calculated value for E being the configuration used for the rest of this work.

This fully-ordered system can be seen in figure 5.12. Ossia then modelled this system, calculating a value called the order parameter for the system. The order parameter allows for the determination of the level of order in the system, with the phase transition occurring as the system changes from ordered to disordered.

The order parameter is calculated using the following algorithm. First, Ossia notes the locations of the magnesium atoms within the initial system, which herein will be called magnesium sites. For each of these magnesium sites, the occupancy, f, is defined as $f = 1$ if the location is occupied by a magnesium atom, and $f = 0$ otherwise. The average occupancy is then calculated for all of the these magnesium sites. In the fully ordered

system, this average occupancy is 1, while for a system that is completely randomly distributed, the average value is 0.25. The order parameter for each magnesium site, q, is then calculated as follows:

$$q = \begin{cases} \frac{4(f - 0.25)}{3} & \text{for each magnesium site;} \\ 4(0.25 - f) & \text{elsewhere} \end{cases} \tag{5.4}$$

The total order parameter, Q, is then calculated as the instantaneous average of all values of q. q, and therefore, Q range from 1 for a fully ordered system, to 0 for a completely disordered system.

Another method to identify T_c is to measure the susceptibility, χ. This is calculated using the formula:

$$\chi = \frac{(\langle Q^2 \rangle - \langle Q \rangle^2)}{T} \tag{5.5}$$

where $\langle Q^2 \rangle$ is the average of the square of the order parameter throughout the Monte Carlo simulation, $\langle Q \rangle$ is the average of the order parameter throughout the simulation, and T is the simulation temperature.

χ measures the fluctuations in the system order parameter. $\chi \to \infty$ as $T \to T_c$, which implies that $\chi^{-1} \to 0$ when $T \to T_c$. Therefore, the consideration of the values of Q and χ^{-1} directly give T_c.

Computational method

The combination of the requirement for a large number of independent simulations, and Ossia's ability to create its output in CML format, makes this area of study a perfect application for the parameter sweep tools described previously. These tools were used, taking a template Ossia input file and the range of temperatures to model, creating and submitting the sweep of simulations. The temperature was initially varied between 0 and 5800 K, with 200 individual simulations. This allowed the system to be considered at a high resolution, leading to the determination of T_c to a high accuracy.

Analysis of the results from this initial 200 jobs showed an approximate temperature range containing T_c. In order to identify T_c with more accuracy an additional 200 jobs were run, considering a temperature range of 0–1160 K. Finally, a further 50 jobs were run, considering the temperature range of 290–350 K, increasing the resolution still further.

Each job was set up and submitted using the standard parameter sweep tool functionality. They were run using the NW-Grid resources, which were chosen because they provide a

large amount of processing power, allowing for the whole of each sweep of jobs to be run concurrently. The simulation execution process involved the capture of large amounts of metadata from each of the submitted jobs. This metadata was captured automatically, using the functionality provided by *MCS* and the parameter sweep tools. The metadata that was collected includes the following pieces of information:

- Energy, and its value squared;
- Heat capacity;
- Order parameter, $\langle Q \rangle$, and its square, $\langle Q^2 \rangle$;
- Susceptibility, χ;
- The inverse of the susceptibility, χ^{-1};
- Simulation temperature.

These values were collected by specifying several metadata collection lines within the parameter sweep tools' configuration file. These lines were then placed into the *MCS* input files by the tools when they were created. The *MCS* metadata collection lines follow, where `trans.xml` is the name of each created Ossia output file:

```
AgentX = Energy,trans.xml:/Module[last]/PropertyList/
                Property[dictRef=‘ossia:Energy’]
AgentX = EnergySquared,trans.xml:/Module[last]/PropertyList/
                Property[dictRef=‘ossia:EnergySquared’]
AgentX = OrderParameter,trans.xml:/Module[last]/PropertyList/
                Property[dictRef=‘ossia:OrderParameter’]
AgentX = OrderParameterSquared,trans.xml:/Module[last]/PropertyList/
                Property[dictRef=‘ossia:OrderParameterSquared’]
AgentX = HeatCapacity,trans.xml:/Module[last]/PropertyList/
                Property[dictRef=‘ossia:HeatCapacity’]
AgentX = Susceptibility,trans.xml:/Module[last]/PropertyList/
                Property[dictRef=‘ossia:Susceptibility’]
AgentX = InverseSusceptibility,trans.xml:/Module[last]/PropertyList/
                Property[dictRef=‘ossia:Stiffness’]
```

This metadata included all of the information required to identify T_c, except for the actual temperature modelled, which was collected as part of the default metadata collection

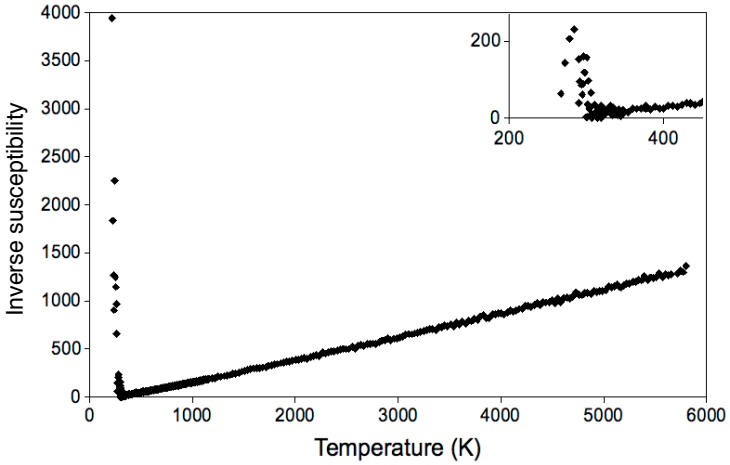

Figure 5.13: A graph of the calculated inverse susceptibility, χ^{-1}, for all modelled temperatures of the octahedral layer of the studied clay. The point where $\chi^{-1} = 0$ corresponds to T_c, the temperature at which the phase transition occurs. The inset image shows the point around T_c in more detail.

performed by *MCS*. Therefore, the actual simulation output did not need to be considered as part of the results analysis; rather all such analysis could be performed through the use of the project metadata tools. These tools allowed for the retrieval of all of the metadata relevant to the analysis of the results for every simulation performed as part of this study. Once retrieved these values could be plotted directly allowing trends in the data to be seen straight away. The output files were of course archived and so could be retrieved and checked if something were to go wrong.

The collection of the squared system energy and order parameter allow for further analysis of the results with a minimum of effort. Since these squared values are calculated by ossia itself, it makes sense to capture the value as metadata directly, rather than to recalculate the value as part of any further analysis.

Traditional analysis of this data would either require the user to manually work though several hundred data files, or to write their own script that does so automatically. Both processes would require that all relevant data be collected and then collated. However, the use of the metadata tools makes the results analysis much simpler, with all of the required data collected together, with meaning, in one single place.

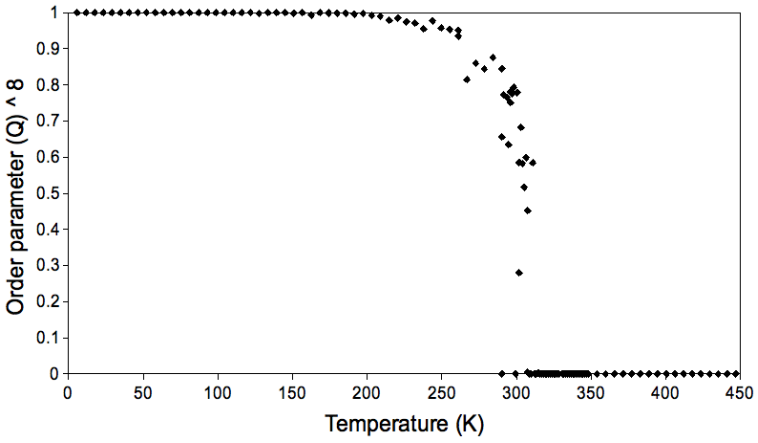

Figure 5.14: A graph of the calculated order parameter to the power of 8, Q^8, for modelled temperatures of the octahedral layer of the studied clay. The point where Q^8 falls to zero corresponds to T_c, the temperature at which the phase transition occurs.

Results and conclusions

As has already been mentioned, the analysis of the data created by each of the Ossia simulations was performed purely using the project metadata tools. The data was retrieved from the metadata database and graphs of the pertinent data plotted. The result of plotting χ^{-1} can be seen in figure 5.13.

Theory shows that the system order parameter, $Q \propto (T_c - T)^{\frac{1}{8}}$ for two-dimensional systems, when T_c is close to T. This means that $Q^8 = 0$ when $T_c = T$. Plotting Q^8 for all simulated temperatures will therefore give T_c, with any trends in the data being much more pronounced than a similar plot of Q against temperature.

The evolution of Q^8 with changing temperature can be seen in figure 5.14, where it is plotted. Careful analysis of the combination of the trends shown in figures 5.13 and 5.14 show that $T_c \approx 307.5\ K$.

This study has shown that in a very short space of time it is possible to configure and run a large number of related simulations. Each of these simulations, when executed using *MCS* and the parameter sweep tools, was able to collect all of the important calculated data as metadata, allowing for very simple analysis of the results. The collection of the data in this manner means that collaborators were also able to access the extracted values, performing any analysis they wanted, without requiring the data extraction to be repeated by each collaborator.

In the past this study could have been considered using Ossia in parallel on large HPC type resources, where one simulation ran all of the required temperatures. However, the trivially parallel nature of these calculations, where each individual temperature can be modelled independently of the others, is more suited to the resources made available by grid systems.

Once the move had been made to the use of grid systems, it would not have been feasible to create so many separate simulations without the use of the parameter sweep tools. Also, the submission and management of such a large number of jobs would have been difficult, if not impossible. Finally, it is unlikely that the analysis of the created data could have been performed on the scale allowed by the automated metadata collection, which means that it would have been difficult to determine T_c to the same level of accuracy as has been allowed by this study.

5.3.2 Simulations of amorphous silica under pressure

A third case study to which these tools have been applied as part of their development and testing is the modelling of amorphous silica under varying pressure. This work was performed in collaboration with other project members and an undergraduate student within my department, Lucy Sullivan. Along with myself, each of the collaborators made use of the parameter sweep tools described previously to create and manage their simulations. Lucy's involvement was important to the study, because she was a complete novice to grid computing and so provided a good test of the level of usability associated with these tools for a user without any prior experience, or knowledge of grid systems.

The challenge

Amorphous silica exhibits strange behaviour when exposed to increasing pressure [66]. The volume of most materials will be reduced under increasing pressure, which results in the material becoming stiffer and harder. Amorphous silica does not behave in this manner; rather, it becomes softer before then hardening.

One theory explaining this behaviour is that when under pressure the bonds between the silica tetrahedra will initially deform, accommodating changes in volume, rather than deforming the tetrahedra themselves. This deformation leads to a change in the compressibility of the structure, which in turn changes how the structure reacts to the pressure. Previous work has shown that under high pressure these bonds break, leading to re-

duced compressibility [65]. Conversely, under high negative pressure, these bonds will be stretched, again leading to reduced compressibility. The combination of these effects implies that there must be a compressibility maximum in between the two extremes, leading to the unusual behaviour exhibited.

Method applied

The study of silica using a version of DL_POLY_3 with CML support to perform molecular dynamics simulations allows for the consideration of silica under many different pressures, determining the structure for each value. My parameter sweep tools were applied to this problem, performing several sweeps across the range of pressures.

Some of these sweeps were performed by Lucy and other project members, while others were performed by myself. This sharing of the work meant that the collaborative sections of the *minigrid* were required to be used on a scale that had not been attempted within the *e*Minerals project prior to this study. These tools allowed the data and associated metadata, which were each collected and archived automatically using *MCS* and the parameter sweep tools, to be shared between the collaborators, as well as assisting with associated communication.

This case study required the configuration and management of simulations in exactly the manner for which the parameter sweep tools were designed, which meant that they could be applied directly to the system without any modification. They were used to create and execute several hundred simulations over a pressure range of -5 GPa to +5 GPa. The performance of this many jobs, covering a pressure range of this size, is something that would not have been possible prior to the use of grid technologies, and in particular tools such as the parameter sweep tools.

In addition to varying the pressure, it was necessary to confirm the produced results by using two different interatomic potentials 'TTAM' [67] and 'BKS' [70]. Each of these potentials was applied to three defect-free amorphous silica molecular structures. One considering 1536 atoms (512 SiO_2 tetrahedra), while the other two considered 12288 atoms (4096 tetrahedral units). This led to six sets of jobs, each considering the whole pressure range given above.

In an effort to reduce the user involvement in the analysis of the output from each of the DL_POLY simulations, the simulations were combined with certain analysis steps. This was achieved by instructing *MCS* to call a simple script, which itself invoked two DL_POLY simulations, followed by some data manipulation performed by a bespoke Perl

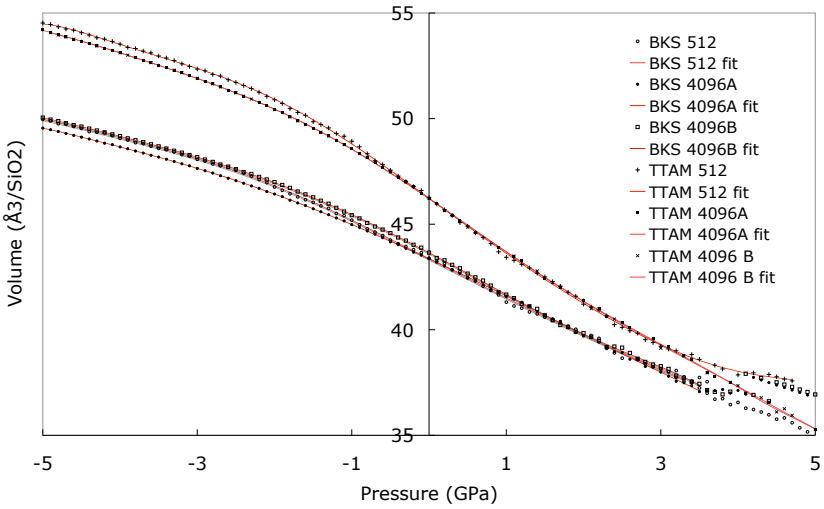

Figure 5.15: A graph showing the calculated volume of each model of amorphous silica with varying pressure. The changing gradient corresponds to the unusual behaviour of amorphous silica under pressure. The shown curves are lines of best fit through the calculated data. This image was created by Prof. Martin Dove for use in publications related to this work.

program, and then analysis of the atomic vibrations experienced within the simulation using the GASP analysis code[5].

The GASP analysis compared each stored configuration with every other stored configuration created as part of the simulation process. The differences between the configurations were then partitioned into motion caused by rigid rotation and translation of the tetrahedra and distortion of the tetrahedra. The analysis was performed automatically as part of running each job, meaning that the user was not required to perform the process manually.

Attempting to analyse the output data from each of the many simulations using the the user's own desktop computer would be untenable, simply due to the sheer number of simulations involved, and the amount of computational power required to perform each analysis. To address this problem, the analysis was performed as part of the submitted job, using the computational power of each of the execution machines, spreading the load and making the analysis feasible.

[5]GASP is a system developed within Cambridge University to help with the analysis of output from other simulation codes. It is described in detail in [73], [74] and [75].

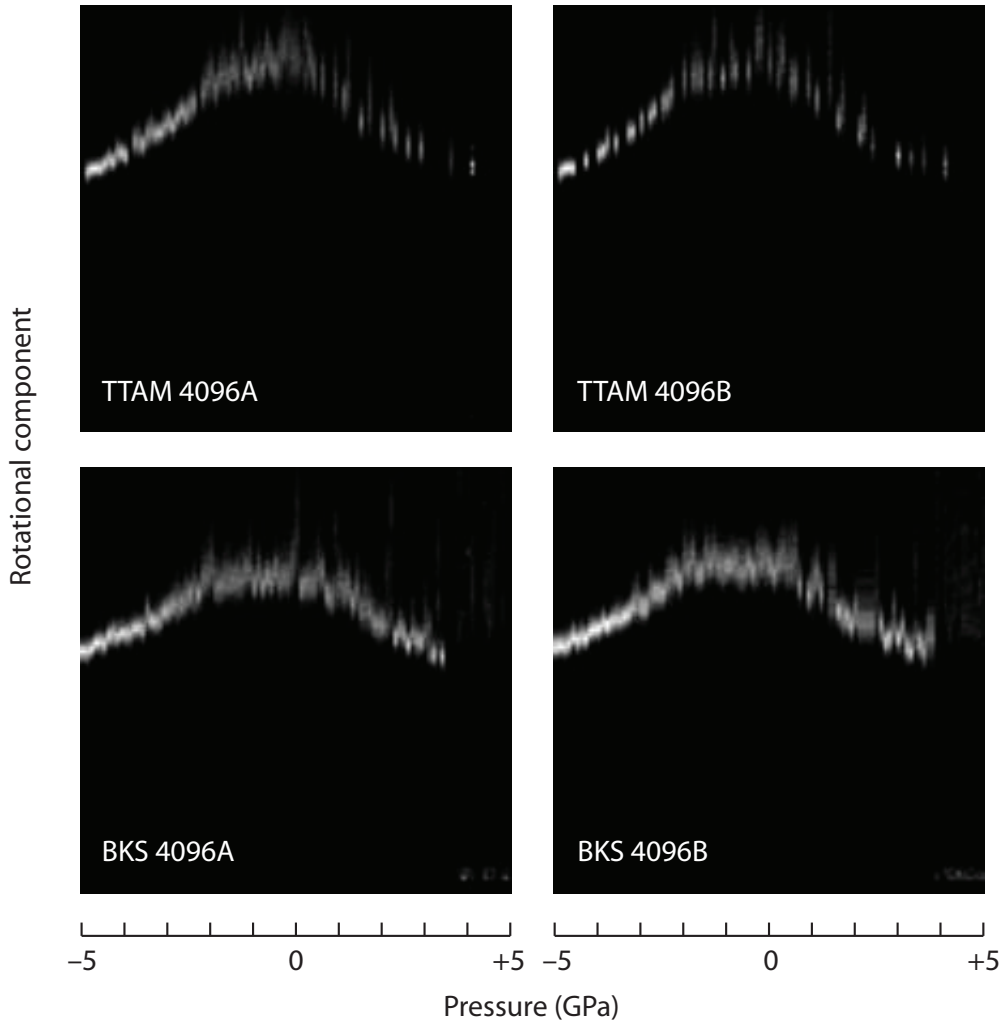

Figure 5.16: Four collections of histograms of the mean-square rotational components of the instantaneous atomic displacements for the two different potentials for both sizes of modelled amorphous silica systems. There is a separate histogram for each pressure, and in this representation higher values of the histograms are in white and zero values are black. Pressure, varying from -5 GPa to +5 GPa, is shown along the horizontal axis, with displacement squared on the vertical scale. This image was created by Prof. Martin Dove for use in publications related to this work.

Results

The analysis of all of the performed simulations showed how the volume of the amorphous silica structure changes with pressure. This was easily shown by visualising the calculated volume for each sweep of runs as shown in figure 5.15. The changing gradient of the line passing through the data points corresponds to the unusual behaviour of silica under pressure, with a steeper gradient corresponding to a softer material.

The visualisation alone does not show the reason for the unusual behaviour, only that this behaviour was exhibited as expected. This is the reason that the analysis performed using GASP was required. This analysis allowed for consideration of the measurement of the vibration both between repetitions of the silica structure, and within this structure itself.

Figure 5.16 shows the results of this analysis as a set of histograms, which show two types of dynamics: the fast vibrational motions, which are represented by the peaks in the histograms that stretch across the range of pressures, and slow/rare localised large-amplitude rotations of groups of tetrahedra that are shown as the long high-value tails in some of the histograms. It can be seen from figure 5.16 that the rotational flexibility has its maximum in the intermediate pressure range where the compressibility has its maximum, concordant with the hypothesis of the origin of the compressibility maximum. This analysis is described in much more detail in [26].

In conclusion, the work performed for this case study showed that initial increases in pressure result in deformation of the arrangement of the silica tetrahedra. This in turn explains the unusual behaviour exhibited by amorphous silica as predicted.

This study has shown that the parameter sweep tools can be directly applied to a large combinatorial study, enabling the archiving and sharing of data and metadata between several collaborators. It has also shown how *MCS* can be used with a complicated script to combine simulation execution and analysis as part of a single job, all performed on the remote computational resource. Additionally, it has shown that the tools can easily be applied by users with no previous experience of grid computing technologies. This means that the users can concentrate on their research, without being required to learn complicated job submission systems.

5.4 Other applications of my tools within *e*Minerals

5.4.1 An introduction to other uses

The following sections detail other uses of my tools within the *e*Minerals project. I have not been directly involved with these tasks other than to provide the tools that were used and any required instruction in their use. They have been included here to further show the wide range of research to which these tools can and have been applied. Although

these tasks do not include all research that has been performed using my tools, they do represent the different ways in which they can be used.

5.4.2 Parameterising models of PCB molecules

A study performed by other *e*Minerals project members that has made extensive use of *MCS* and the parameter sweep tools is the consideration of the adsorption of polychlorinated biphenyl (PCB, $C_{12}H_xCl_{12-x}$) molecules onto the pyrophyllite (001) surface, which is discussed in detail in [6]. This study is currently ongoing within the *e*Minerals project, but has already made extensive use of the tools discussed in this thesis in order to parameterise the required simulations. The use of these tools has allowed for the consideration of the system at a previously infeasible resolution, which has in fact led to the discovery of an unexpected computational artifact that traditional methods would probably not have identified.

As with the study of the adsorption of PCDD molecules onto the calcite surface described above, a large amount of preliminary simulations are required before the actual calculation of relevant adsorption energies can be calculated. The pyrophyllite surface has previously been constructed for the work discussed in [76]. The work discussed here considers the parameterisation of the PCB molecules themselves, with all of the involved simulations being performed using the SIESTA simulation code.

This parameterisation was divided into two stages. Firstly determining the size of the box containing the PCB molecules ensuring no self-interaction between periodic images. Secondly, the PCB molecules are able to rotate around their central bond. The ease of this rotation is expected to greatly influence the adsorption energy of the molecule onto the surface and consequently it is very important that it is adequately described in the simulations.

The first stage of the parameterisation was performed by taking the PCB molecule with the largest space-filling contribution in a planar configuration. This PCB is the 3,4,5,3′,4′,5′-hexachloro biphenyl, which measures approximately 10 Å in length and 5 Å across the chlorophenyl rings. This molecule was then placed in a box measuring 15 Å×10 Å×10 Å, with the molecule aligned along the long axis of the cuboid. The two shorter sides were kept equal to allow for free rotation around the central C–C bond. The lengths of the sides were then increased to 25 Å for the longer side and 20 Å for the other sides in 0.5 Å increments. Each of these different box sizes were then modelled, leading to over 100 different simulations. Each simulation performed a single point calcu-

lation on the system, determining the degree of interaction across the periodic boundary conditions.

This high resolution sampling of box dimensions meant that an unexpected phenomena was experienced in the results. It was found that the fineness of the grid over which the system's energy was calculated was insufficient for the PCB molecule, resulting in periodic fluctuations in the energy corresponding to the distance between grid points, which is known as the 'eggbox' effect. While the mesh cutoff is always tested for convergence before starting production calculations, it was not expected that the eggbox effect would have such an input to the calculated values. The effect was removed by fine-tuning the grid size. This tuning was performed by running a total of fifteen sweeps across the box dimensions as described above, varying the grid size between each sweep, searching for convergence.

As part of each simulation the modelled lattice vectors were collected by *MCS* and stored as metadata allowing for simple indexing of the data at a later stage. In addition, environment and default metadata were collected, providing an audit trail for all of the created data. This metadata capture was very useful because over 1000 simulations were performed for this parameterisation stage, which would easily have become unmanageable using traditional techniques.

The result from this parameterisation of box size was that a mesh cutoff of 300 Ry and a combination of fcc and box centre grid cell sampling led to the eggbox effect being minimised. In addition, box dimensions of 15.9 Å×12.7 Å×12.7 Å were found to be sufficient to prevent self-interaction between periodic images. 1815 simulations were performed in order to parameterise the system to this resolution. It is probable that traditional methods would not have identified the eggbox effect at all, let alone resulted in the system being parameterised to remove these effects.

The second parameterisation stage, the calculation of the torsion energy within the molecule, ensuring that the rotation around the central bond is correctly modelled, was performed by additional parameter sweeps. The ortho-2,2′-dichloro biphenyl PCB was modelled, fixing the molecule's torsion angle. This fixed angle was then varied over 360° in 5° increments around the central C–C bond. These simulations were configured and submitted using the same method as the above parameterisation.

Analysis of the results from the second parameterisation phase showed that the barrier to rotation within the PCB molecule is highest at 0°, and that there is a minimum in the calculated torsion energy at approximately 70°, which matches results in the literature, both experimental [49] and computational [80]. This minimum is only slightly lower than

the broad minimum within which it sits, between 70° and 130°. The kinetic barrier to rotation is around 1.25 eV at 180° and 3.25 eV at 0°, although a more detailed sampling is needed at 0° in order to determine this value accurately.

5.4.3 Studies of pollutant arsenic ions

Each of the above case studies consider the use of *MCS* and the other tools to perform whole sweeps of simulations. However, the tools are equally applicable to smaller scale simulation studies where each simulation is configured and submitted manually.

An example study that makes use of *MCS* in this manner, which has been performed by other *e*Minerals project members, is the consideration of the incorporation of arsenic into pyrite (FeS_2), which is discussed in detail in [28]. This study was conducted by placing the arsenic in iron or sulphur sites within the pyrite structure. The substitution for sulphur was performed in three different ways. Firstly, by the formation of AsS groups. Secondly, by forming As_2 groups, and finally, by substituting one arsenic atom for one S_2 group. Each of these different incorporation methods was studied under simple oxidising and reducing conditions.

These separate methods and conditions led to the requirement for the study of eight separate systems. These systems must each be modelled in several stages in a similar fashion to the study of PCDDs on a calcite surface above. The full simulation of arsenic incorporation into pyrite is a very large, computationally expensive system. As such they were performed using HPCx[6], which is a very powerful HPC resource. However, the simulations used to set up and configure these large systems were performed using the UCL Condor pool, which is a contributed part of the *e*Minerals *minigrid*. These latter simulations were performed using *MCS* directly, with the required input files created by hand. This shows that *MCS* can be applied to simulations such as these, which must be performed manually and in small numbers.

5.4.4 Simulations of uranium oxide

Another application of my tools is the investigation of the electronic and magnetic structure of uranium oxide (UO). This work was performed by other *e*Minerals project members working at Daresbury Laboratory. The SIC-LMTO-ASA method is applied [62],

[6] http://www.hpcx.ac.uk

which is an implementation of DFT that is able to more accurately model strongly correlated electron systems. This method allows for the simultaneous investigation of the magnetic ordering in the system and the localisation of the f electrons.

This study considered ten different lattice parameters, for three different valency scenarios for both ferromagnetic and anti-ferromagnetic uranium configurations, resulting in a total of sixty simulations. The groundstate was determined to occur in the ferromagnetic U+5 valence configuration, with a lattice parameter, a = 4.773 Å, which is close to the published value of a = 4.92 Å [50]. The next stage of the study is to consider the entire pnictide and chalcogenide series (UN, US, UP, UAs, USb, UBi, US, USe and UTe), using the method outlined here.

Each of the simulations required to model UO were submitted using *RMCS*, which makes use of *MCS* to perform the actual job submission. The jobs are each performed using NW-Grid resources, with *MCS* metascheduling between the available clusters. The simulation code used does not create output in an XML format and so cannot have metadata collected from its actual output as part of the *MCS* submission. However, environment and user specified metadata are both collected as part of each job, providing accountability and allowing the user to trace their created data.

5.4.5 PCDD adsorption onto pyrophyllite

The final study discussed in this thesis is one that has been performed by Arnaud Marmier who is an *e*Minerals project member based at the University of Bath. This study involves the continuation of previous work discussed in [76], considering the adsorption of the PCDD pollutant molecules discussed earlier onto the (001) surface of pyrophyllite.

This investigation involves the use of N different molecular dynamics simulations, where each simulation considers a PCDD congener placed at a different height above the pyrophyllite surface. The relative accuracy of the investigation is proportional to the number of simulations performed, N, and it has been found that in order to achieve suitable accuracy 300 different simulations were required per congener. Consideration of all seventy-six PCDD congeners therefore leads to approximately 23,000 simulations, each of which requires approximately twenty-four hours of processing time, which makes this study particularly suitable to the HTC resources typically provided by grid systems.

Each of the different simulations were submitted using *MCS*, which was wrapped by tools written especially to manage this study by Arnaud. The simulations were all performed

using the NW-Grid computational resources ensuring that the large number of required simulations could all be performed, with *MCS* metascheduling between the different clusters available.

5.5 Conclusions

This chapter has described the application of the job submission and data analysis tools discussed in previous chapters to some real scientific research topics within the *e*Minerals project. Most of these different research topics would not have been feasible prior to the use of grid techniques and the tools discussed in previous chapters. Those that could be considered in the past have been made easier and more accountable by the use of these tools.

In addition, it is unlikely that each of these topics would have been investigated to the level of detail demonstrated here, without the use of the parameter sweep tools. While it would have been possible to create the necessary simulation input files in the past, the analysis of any created data would not be feasible manually. The combination of the metadata collection functionality of *MCS* and the automatic visualisation of simulation output such as that provided with the parameter sweep tools is key to the ability to analyse studies on this scale. This has been shown by the study of cation ordering in the octahedral layer of a clay, where all analysis was performed using the captured metadata alone.

The main study discussed in this chapter is the adsorption of PCDD molecules onto the surface of calcite. This study is representative of many of the studies performed within the *e*Minerals project. It found that the position of the PCDD congeners above the calcite surface is an important contributor to the adsorption energy. The two PCDD congeners modelled above the surface both exhibited a minimum in the system energy when placed at the same location above the surface. Since these two congeners are at the extreme ends of the PCDD series, it is expected that all other PCDDs will also exhibit a minimum energy at this location, where they will adsorb to the surface most strongly.

This case study has also shown that the strength with which the PCDD molecule adsorbs to the calcite surface is dependent on the level of chlorination of the molecule. Higher levels of chlorination lead to larger adsorption energies and as such a strong binding to the surface. This was expected, because it correlates with results for the PCDDs above the surface of pyrophyllite, discussed in [76].

Other studies to which the job submission tools have been applied have also been introduced, showing how the tools can be used in different situations and with different aims. *MCS* provides the key job submission functionality required by each of the different submitted jobs, and is used in all of the different scientific research topics considered by the *e*Minerals project. The case studies in this chapter show how *MCS* and the parameter sweep tools have been used by the project scientists, allowing them to perform their research using the emergent grid infrastructure without being distracted by learning how to use complicated submission systems.

Chapter 6

Conclusions

This thesis has described my efforts to make the idea of the *integrated grid* a reality, combining the important components from computational, data and collaborative grids. These systems are each linked together in order to allow the user full access to the power provided by grid technologies, while still remaining usable and not interfering with the user's research.

In order to test that the *integrated grid* idea can be used effectively within scientific research a first implementation was developed, called the *e*Minerals *minigrid*. The *minigrid* integrates several computational clusters and Condor pools, with data and metadata management provided by the SRB and RCommands respectively. Access to these resources is strictly controlled and allowed only through the use of grid middleware provided by the Globus toolkit.

The *minigrid* has many users, most of whom are members of the *e*Minerals project, although other collaborators are also allowed access to the available resources. These users have made use of the *minigrid* to perform simulations relating to each of the different research topics studied by the *e*Minerals project, a subset of which have been discussed in this thesis. The resources provided are mainly used for long running serial simulations, although they are occasionally used for small scale parallel simulations where the communication between processors is kept to a minimum.

Currently the *minigrid* acts as a resource oriented system, which has started to be seen to reach the limits of scalability in terms of the number of jobs that can be performed at any one time. This has not limited the usefulness of the *minigrid* because these limits are not experienced when submitting jobs on the scale allowed by the *minigrid* resources. However, these limitations are starting to be experienced when users move from systems

the size of the *minigrid* onto much larger resources. This could be addressed in the future by moving to a service oriented architecture, where all communications between resources make use of web services, which are starting to be seen to scale better than the current model.

I developed *my_condor_submit* (*MCS*) to provide a simple and uniform interface to each of the resources within the *minigrid*. *MCS* provides powerful metascheduling functionality allowing the user to simply submit their job to be executed, relying on *MCS* to discover and make use of any available computational resource.

MCS also provides powerful metadata capture functionality, making use of ontology-based libraries to extract available information from the user's data. This functionality has been seen to make the user's analysis of their research much easier. In some cases it has allowed all analysis to be performed without even requiring the consideration of the user's simulation output files. This is very important when performing studies on the scale allowed by grid techniques, where the user cannot hope to cope with the deluge of created data using traditional methods alone.

Other job submission and management tools have also been developed that wrap *MCS*, allowing the user to create and manage whole sweeps of simulations at once. These tools have been applied to several large studies including those discussed in this thesis.

The main scientific study discussed in this thesis considers the adsorption of PCDD molecules onto the $(10\bar{1}4)$ of calcite. This study has applied both parameter sweep tools and *MCS* directly, to create and manage the large numbers of simulations required in order to determine how strongly two representative PCDD congeners adsorb onto the calcite surface. In calculating these adsorption energies a location has been found at which the molecules will adsorb most strongly.

This study has shown that these tools can successfully be applied to scientific research. Other project members have now taken the results of this work and are applying it to the study of the other seventy-four PCDD congeners, calculating adsorption energies for each of them. These results will allow for more concrete conclusions to be drawn regarding the strength with which each of the different PCDD congeners adsorb to the calcite surface, which in turn will allow for the determination of factors that cause the different congeners to exhibit different adsorption energies. These conclusions, in combination with other studies of the PCDD molecules within the *e*Minerals project will help to make predictions regarding the adsorption of these pollutants onto other representative materials within the environment.

The current status of the tools developed and discussed within this thesis allows for simple and effective use of grid resources by the user. However, they could be developed further in the future should time permit. The main area that could be developed is the implemented metascheduling algorithm.

The decision as to where to execute a user's code is very complicated, with the algorithm employed within *MCS* making several assumptions about the code execution environment as discussed in section 3.2.3. The development of this algorithm to incorporate information about job queueing and run times would improve the decisions made within *MCS*. This enhanced algorithm would need to make use of statistical methods to predict the amount of time that a user's job will spend queueing on any available resource before executing, choosing the machine that will result in the shortest wait. This will allow *MCS* to function better when used to submit to resources that are very busy and which are also accessed using other submission methods. However, it has not been considered within the current research because the *e*Minerals project does not typically use resources that are busy enough to justify the large amounts of work that will be required to develop and implement these changes to the algorithm.

Further development could also be applied to the data management sections of the submission tools, making them more generic and allowing the use of systems other than the SRB. This would be achieved by extracting the SRB interaction functionality from the core *MCS* code into a separate module. Other modules could then be developed, implementing the same functionality using other data management systems. The user would then be able to specify the data management system to be used as part of their *MCS* input file, with *MCS* making use of the appropriate module.

This support for multiple data management systems has not been considered within the current research because all data management performed within the *e*Minerals project is provided by the SRB. However, for *MCS* and the associated tools to be used more widely in projects outside of *e*Minerals they would need to be more flexible in this area. This flexibility would reduce the number of components that the prospective user would be forced into using should they wish to use *MCS*. It should not be expected that anyone wishing to use the functionality provided by *MCS* would want to be required to use the SRB system for their data management.

In addition to research performed within the *e*Minerals project, *MCS* and the other developed tools are being evaluated and used by other research groups for job submission and management. These other users work in the band theory, quantum chemistry, and application visualisation groups at Daresbury Laboratory, where they investigate various

different research topics. Access is primarily being given to *MCS* through the use of *RMCS*, which means that they are able to gain the power of *MCS* without the hassles associated with installing all of its prerequisites.

In conclusion, the tools discussed in this thesis have been applied successfully to several research topics, allowing for the study of these topics in previously infeasible detail. These tools have allowed users access to the full power afforded by grid technologies without requiring them to learn how to use complicated job submission and management systems. Additionally the user has been able to integrate new metadata collection and storage mechanisms into their day-to-day work, allowing enhanced collaboration and data accountability without interfering with their normal working practices. This integration will be key to the uptake of grid computing methods in mainstream use.

Appendix A

AgentX

This appendix discusses the *AgentX* tool and some of its uses within the *e*Minerals project. Of the uses introduced here, many are discussed in detail elsewhere in this thesis, as listed below.

A.1 An Introduction to *AgentX*

AgentX is a tool developed at Daresbury Laboratory, to enable the processing of data documents, extracting relevant data and their context, as specified by the user. *AgentX* primarily works with XML data formats, although it can also be applied to normal text files and development is in progress to allow the use of data stored in databases.

AgentX implements a simple API that can be used by other applications to find data in a document according to their context. The context is specified through a series of queries, based on terms specified in an ontology. These terms relate to concepts of interest (for example '*Atom*', '*Crystal*' and '*Molecule*') and their attributes (for example '*zCoordinate*').

This approach requires a set of mappings relating these terms to document fragment identifiers. It is these identifiers that are evaluated to locate parts of documents, representing data with a well defined context. In this way, *AgentX* can hide the details of a particular data format from its users. This means that the complexities of dealing with XML file formats, and the necessity to understand the precise details of a particular data model are removed. The user is not required to understand the details of the ontology or mappings, but must have an understanding of the terms used.

The first production level usage of *AgentX* was its integration into *MCS*, where it is used to extract metadata from simulation code output files. The usage of *AgentX* in this manner is discussed in detail in section 3.2.3. This integration showed how useful *AgentX* can be in retrieving information from CML files, which led to the integration of *AgentX* directly into the project simulation codes, so that they are able to read CML files directly in addition to their traditional input file formats. This integration into simulation codes is discussed below in section A.2.

The *AgentX* paradigm can be one that is difficult to understand for users new to the system. In order to simplify the introduction to the *AgentX* API and ontology, the *ontologyViz* tool was developed. This tool is introduced below in section A.3 and discussed in more detail in section 4.3.3.

A.2 *AgentX* usage

AgentX has several uses within the *e*Minerals project. Firstly, it is used to extract metadata from the simulation code output files as a part of the *MCS* job submission process. Secondly, it is used to provide interoperability between the different simulation codes. Finally, it is used to convert between different data formats.

Information extraction - The main use of *AgentX* within the *e*Minerals project is to extract information from the different data files produced by the simulation codes used. Much of the metadata extraction performed by *MCS* is implemented by making calls to *AgentX*. The use of *AgentX* in this manner allows the user to deal with higher level concepts than is allowed by accessing XML files directly. These concepts will correspond to terms used by the scientist in their everyday work and so will mean that the user does not need to invest effort into learning new terms. The information extraction process is discussed in detail in section 3.2.3.

Interoperability - *AgentX* has been incorporated directly into the DL_POLY_3 simulation code, where it is used to replace the section of the code that reads in the input file, with all parsing code replaced with calls to *AgentX*. This allows for the retrieval of appropriate values, based on semantic meaning within the input file, rather than simple document structure. As an example, an initial system pressure may be retrieved by using *AgentX* to select the value of the element with dictionary reference 'initial pressure'.

When using this paradigm, every element marked up within the different file formats is required to have a dictionary reference. This allows the user reading the data to be able to look up both the meaning of any concept and the derivation of its value. The inclusion of the dictionary references in the actual file syntax allows this process to be performed by the user both when reading the file directly and when viewing the results of visualisation, as described in section 4.2.6.

The use of these dictionary references also makes it possible to generate one all-encompassing ontology, mapping logical concepts to the different dictionary references within each of the individual file formats. This approach is something that *AgentX* aims to provide. This will allow any code using *AgentX* to simply request the value of a certain concept from the input file, in whatever format it is written, and whatever method is used to store it. This makes the code independent of any one file format, which in turn allows for interoperability between different codes.

Data format conversion - A tool called '*axTransform*' has been developed within the *e*Minerals project that makes use of *AgentX* to convert between different file formats within the *e*Minerals project. The transformation process is slightly more complicated than the use of pure XML methods, as described in chapter 4, because it is required to perform the necessary *AgentX* calls on the initial data.

A suitable algorithm for converting between data formats using *AgentX* is to read a template file, writing the contents to a new file, until an *AgentX* data specification is found. Once such a specification is encountered, *AgentX* must be invoked, retrieving the required data from the initial data file.

Since the transformation process is standard for all such conversions, *axTransform* was developed to perform the transformation, given a required template file and suitable data. *axTransform* was initially developed separately to *AgentX*, but has now been incorporated into the standard *AgentX* release. The data specification format used by *axTransform* has been directly copied from the metadata collection lines of *MCS*, which are described fully in section 3.2.3.

A.3 The *AgentX* ontology

As has already been mentioned, *AgentX* is based on the use of ontologies to provide semantic meaning. The ontology allows for the mapping from the concepts presented

to the user by *AgentX*, to the actual document structure and on to the relevant data. Ontologies however are a difficult concept to understand and also to navigate.

The ontology used by *AgentX* is implemented using the Resource Description Framework (RDF)[1] and the Web Ontology Language (OWL)[2], marked up using XML. The use of these two languages allows for the specification of each concept, and their relationships to one another. However, these languages and the principles they specify are very complicated, which makes it difficult to understand the meaning of the ontology.

To enable the typical scientist to understand and make use of *AgentX* and its ontology I have developed a tool that allows the user to view the ontology, called '*ontologyViz*'. *ontologyViz* visualises the ontology allowing the user to navigate their way through its structure, only showing concepts that are related to the currently selected concept. *ontologyViz* and its usage is discussed in detail in section 4.3.3.

[1] http://www.w3.org/RDF/
[2] http://www.w3.org/TR/owl-features/

Appendix B

The Chemical Markup Language (CML)

This appendix discusses the Chemical Markup Language (CML)[1], which is an XML language designed to represent chemical data. CML is used heavily within the *e*Minerals project and is relied upon for much of the metadata collection performed by *MCS*. It is also required for the visualisation performed by tools such as *ccViz* and my graphing tools as discussed in chapter 3.

B.1 An introduction to CML

CML was one of the first XML languages and has been refined over many years, developing into a very powerful language, able to represent many different aspects of chemical data. Representation can range from simple specification of atoms and their movement, right through to specification of whole molecular simulation runs.

The use of XML based languages for simulation code input and output files means that the process of reading and writing the files can be simplified. With traditionally formatted (non-XML) data files, each simulation code must implement its own method for reading from, and writing to, these files, taking care of understanding and maintaining data structure and content.

[1] http://cml.sourceforge.net/

The use of XML means that the codes may simply call standard library functions to read in the relevant information. These libraries are implemented in all major languages, even the computational scientist's favourite, Fortran. The Fortran library for writing XML, called 'FoX'[2] (formerly 'XMLF90') [78] was developed by *e*Minerals project members, to allow Fortran codes to make use of XML file formats.

B.2 *e*Minerals CML standard, '*CMLComp*'

The use of a common language across different simulation codes means that these codes are able to interoperate with one another, exchanging relevant data. This sort of interoperability ensures that the user need not waste time cutting and pasting the output data from one simulation code into the input format of another. Rather, this can all be performed behind the scenes, by the simulation codes themselves.

This method of allowing interoperability is effectively based around the idea of every code using the same file format. In this case, interoperability is obviously trivial, as each code's native format is identical. However, this method is not as simple to achieve and use as might be hoped, because consensus is required between each of the different code authors regarding the different meanings associated with each term, and therefore, how these terms can be used within the input and output file formats.

The sociological problems associated with common formats are not the only potential problems; rather there are technical issues too. CML itself is general purpose and allows for simulation codes to construct file formats from as much, or as little of the language as required. In addition, components can be combined in any order desired by the file author. The combination of these two features of CML means that it is not always trivial for two output formats written using CML to be mapped to one another, or to retrieve the data from one format, when expecting another.

Traditionally, codes may not output enough information to be able to start another simulation, due to additional information being required within the input file. This problem can, of course, also occur in the opposite direction, where the output from a simulation code contains more information than is required for an input. The former of these two problems is, of course, more serious, because extra information can always be ignored, whereas the shortfall in information cannot always be replaced without additional knowledge of the system involved.

[2] http://www.uszla.me.uk/software/FoX.html

Within the *e*Minerals project the native interoperability problem has been tackled by the production of a standard, specifying a subset of the CML language to be used. In addition, the manner in which elements within the subset can be combined is specified, such that the subset is more rigid in defining how it can be used than the full CML language. This subset, called '*CMLComp*'[3], allows for the use of all elements that are required within the output from each of the different codes used within the *e*Minerals project. In fact, each of these codes currently creates CML that is valid in terms of *CMLComp*.

The use of *CMLComp* means that any code creating output in this format can use the *ccViz* tool described in [77]. This means that visualisation tools need not be developed specifically for each code, reducing the effort required, whilst instantly enhancing the benefits, associated with the transition from bespoke output formats to CML based formats.

[3]http://www.uszla.me.uk/CML/eminerals.html

Appendix C

Metadata storage

This appendix describes the concept of metadata, and the different metadata tools that have been integrated into my submission tools and the *e*Minerals *minigrid*, explaining both how they work, and why they are needed.

Metadata is commonly referred to as '*data about data*', and can be used to provide context about other data. When used in this way, metadata is often constrained to use certain, well-defined terms that have semantic meaning. These terms can then be searched by project members, allowing enhanced data discovery.

Within the *e*Minerals project, metadata is handled by the '*RCommands*' [69], consisting of a central application server and metadata database, which stores all of the actual metadata. In addition to this, command line, client tools are provided, which interact with the application server using web services.

The *RCommands* aim to address the problems associated with keeping track of the context of the scientist's data. This is done through the storage of metadata, related to the whole process of a scientist's investigation of a research topic.

The metadata is stored in a hierarchical model, allowing the representation of metadata related to the complete spectrum of work, from individual files, right through to complete studies of a scientist's research topic. This is achieved by the application of a subset of the CCLRC metadata model [61], as depicted in figure C.1.

The three levels of metadata are '*Studies*', '*Datasets*' and '*Data objects*', relating broadly to different levels within a typical scientific study. These levels are designed to be flexible in their usage, allowing the user to adapt the model to fit with their own mental model of

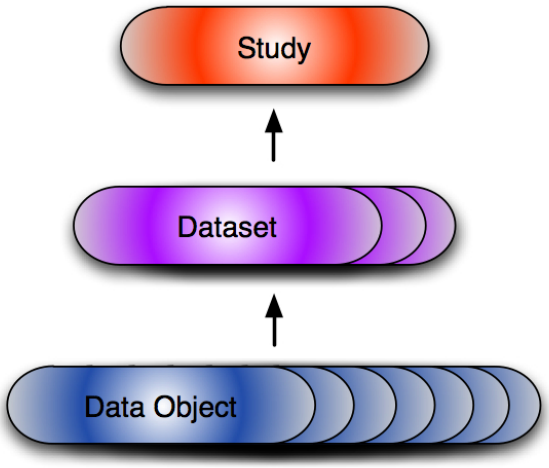

Figure C.1: Representation of the subset of the CCLRC metadata model used within the *RCommands*. *Studies* can contain multiple *Datasets*, which can in turn contain multiple *Data objects*.

their work. Examples of the use of the different metadata elements are given in table 3.1.

The division of metadata into the three separate, although linked sections, means that the scientist's work can be grouped similarly to the manner naturally used within their day-to-day work. This in turn means that the work performed and indexed with metadata, is much easier to find than without metadata. This, coupled with the ability to search both the scientist's own, and their collaborators' metadata, leads to the retrieval of data becoming greatly simplified.

Appendix D

Remote *MCS* (*RMCS*)

'Remote *MCS*' (*RMCS*) is a job submission tool developed within the *e*Minerals project to bring the *minigrid* to the user's desktop. *RMCS* was developed at Daresbury Laboratory and provides a thin wrapping of *MCS*, whilst allowing submission from machines without Globus and Condor installed. It required a number of changes to the workings of the standard *MCS*, in order to function correctly. These changes are described in section 3.3.3.

RMCS implements a standard three tier web services application:

(i) Binary, client command line tools (written in C);

(ii) Web services and database server processes (written using Java);

(iii) An *integrated grid* back end;

Tier (i) represents the tools seen by the user. They are currently a set of command line tools, although work is in progress to create a GUI interface to *RMCS*. Commands have been provided to submit jobs, monitor the current status of all jobs owned by the user and to remove jobs if desired. There is also ongoing development to allow the user to retrieve the output of running jobs, so that the user can monitor their job's actual progress, once it starts executing.

The client commands work by sending a set of well defined web service requests to tier (ii) of the system. The tier (ii) tools then process the received requests, and obtain delegated user credentials, allowing the system to pretend to be the user, invoking *MCS* with the user specified input file. Once the job has been successfully submitted by *MCS*, *RMCS* parses the *MCS* output and stores relevant information in its database.

Other tier (i) commands can then be used to help the user to keep track of their submitted jobs. The command instructs the tier (ii) section of the application to retrieve the required information from the database, and to then perform the necessary querying of the underlying Condor system, used by *MCS* to submit the jobs. Jobs can also be deleted using the same tier (i) to tier (ii) command submission process.

Tier (iii) of the *RMCS* system is an *integrated grid*, such as the *e*Minerals *minigrid*, as has been discussed in detail elsewhere in this thesis. *RMCS* is able to submit jobs to any resource to which *MCS* is able to submit, since the actual grid submission process is handled by the standard *MCS* code.

RMCS is currently being used by some of the *e*Minerals project members as part of further system development. The aim for *RMCS* is to take into account the feedback from the current testing period, before being released to mass use within the project.

Appendix E

Enclosed CD-ROM contents

This appendix outlines the contents of the CD-ROM attached to this thesis. This CD-ROM contains the source code for many of the tools discussed in this thesis. The length of the code involved in these tools prohibits their reproduction in the text of this thesis. A brief introduction to each tool will be given here, with the electronic versions provided on the CD-ROM.

E.1 *my_condor_submit* (*MCS*)

The first tool included on the CD-ROM is the job submission tool *my_condor_submit* (*MCS*), which is discussed in detail in chapter 3. *MCS* is available in both source form and in each of the installation packages that are provided to users within the *e*Minerals project. These packages can be found in the `my_condor_submit` directory.

`my-condor-submit_1.2.0_i386.deb` and `my_condor_submit-1.2.0-1.i386.rpm` are the packages provided for users of the Debian and Redhat Linux operating systems respectively. The `my-condor-submit_1.2.0.tar.gz` file is the source code distribution provided to users who do not have administrative access to the submission machine, or who cannot use the other packages. The `my_condor_submit-1.2.0` subdirectory contains the actual source code, test scripts, user manuals, etc. from which the other packages are created.

E.2 Parameter sweep tools

The parameter sweep tools discussed in chapter 3, which allow the creation and submission of large numbers of jobs are also included. The tools can be found in the `parameter_sweep_tools` directory. The provided commands and their related directory structure are discussed in detail in section 3.4 and instructions for their use can be found in the `README.txt` file provided in the `parameter_sweep_tools` directory.

E.3 *ontologyViz*

The source code for *ontologyViz*, the ontology visualisation tool discussed in chapter 4 is included in the `ontologyViz` directory on the CD-ROM. This source code represents the web page presented to the user. The `ontologyViz.sql` file contains the contents of the *ontologyViz* database at the time of writing.

Bibliography

[1] A L Allred. Electronegativity values from thermochemical data, 1961.

[2] David P. Anderson. Public computing: Reconnecting people to science. In *Conference on Shared Knowledge and the Web. Residencia de Estudiantes, Madrid, Spain*, November 2003.

[3] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: An experiment in public-resource computing. *Communications of the ACM*, 45/11:56–61, November 2002.

[4] M. Antonioletti, M.P. Atkinson, R. Baxter, A. Borley, N.P. Chue Hong, B. Collins, N. Hardman, A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N.W. Paton, D. Pearson, T. Sugden, P. Watson, , and M. Westhead. The design and implementation of grid database services in ogsa-dai. *Concurrency and Computation: Practice and Experience*, 17:357–376, February 2005.

[5] KF Austen. *Computer Modelling of the Structure and Reactivity of Carbonate Minerals*. PhD thesis, University of London, 2006.

[6] KF Austen, TOH White, RP Bruin, MT Dove, E Artacho, and RP Tyer. Using escience to calibrate our tools: parameterisation of quantum mechanical calculations with grid technologies. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 645–652, September 2006.

[7] Chaitanya Baru, Reagan Moore, Arcot Rajasekar, and Michael Wan. The sdsc storage resource broker. In *Proceedings of CASCON'98 Conference*, November 1998.

[8] J Basney, M Livny, and T Tannenbaum. High throughput computing with condor. *HPCU news*, 1(2), 1997.

[9] Rüdiger Berlich, Marcel Kunze, and Kilian Schwarz. Grid computing in europe: from research to deployment. In *ACSW Frontiers '05: Proceedings of the 2005*

Australasian workshop on Grid computing and e-research, pages 21–27, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[10] Jon Blower and Keith Haines. Building simple, easy-to-use grids with styx grid services and ssh. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 225–232, September 2006.

[11] A Bosenick, MT Dove, ER Myers, EJ Palin, CI Sainz-Diaz, BS Guiton, MC Warren, MS Craig, and SAT Redfern. Computational methods for the study of energies of cation distributions: applications to cation-ordering phase transitions and solid solutions. *Mineralogical Magazine*, 65(2):193–219, April 2001.

[12] Richard P Bruin, Martin T Dove, Mark Calleja, and Matthew G Tucker. Building and managing the *e*Minerals clusters: A case study in grid-enabled cluster operation. *Computers in Science and Engineering*, November 2005.

[13] RP Bruin, TOH White, AM Walker, KF Austen, MT Dove, RP Tyer, PA Couch, IT Todorov, and MO Blanchard. Job submission to grid computing environments. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 754–761, September 2006.

[14] R. Buyya, D. Abramson, and J. Giddy. Nimrod/g: an architecture for a resource management and schedulingsystem in a global computational grid. *High Performance Computing in the Asia-Pacific Region*, 1:283–289, 2000.

[15] M Calleja, B Beckles, M Keegan, MA Hayes, A Parker, and MT Dove. Camgrid: Experiences in constructing a university-wide, condor-based grid at the university of cambridge. In *Proceedings of UK e-Science All Hands Meeting 2004*, pages 173–178, September 2004.

[16] M Calleja, L Blanshard, R Bruin, C Chapman, A Thandavan, R Tyer, P Wilson, V Alexandrov, RJ Allen, J Brodholt, MT Dove, W Emmerich, and K Kleese van Dam. Grid tool integration within the eminerals project. In *Proceedings of UK e-Science All Hands Meeting 2004*, pages 812–817, September 2004.

[17] M Calleja, R Bruin, MG Tucker, MT Dove, RP Tyer, LJ Blanshard, K Kleese van Dam, RJ Allan, C Chapman, W Emmerich, PB Wilson, JP Brodholt, A Thandavan, and VN Alexandrov. Collaborative grid infrastructure for molecular simulations: The eminerals minigrid as a prototype integrated compute and data grid. *Molecular Simulations*, 2005.

[18] PA Couch, P Sherwood, S Sufi, IT Todorov, RJ Allan, PJ Knowles, RP Bruin, MT Dove, and P Murray-Rust. Towards data integration for computational chemistry. In *Proceedings of UK e-Science All Hands Meeting 2005*, pages 426–432, September 2005.

[19] Karl Czajkowski, Donald F Ferguson, Ian Foster, Jeffrey Frey, Steve Graham, Igor Sedukhin, David Snelling, Steve Tuecke, and William Vambenepe. The ws-resource framework. Globus alliance whitepaper, 2004.

[20] Nora H de Leeuw. Surface structures, stabilities, and growth of magnesian calcites: A computational investigation from the perspective of dolomite formation. *American mineralogist*, 87:679–689, 2002.

[21] WA Deer, RA Howie, and J Zussman. *An introduction to the rock forming minerals.* Longman, 2nd edition, 1966.

[22] Martin T Dove. *Structure and Dynamics; An atomic view of materials.* Oxford University Press, 2003.

[23] MT Dove, M Calleja, R Bruin, J Wakelin, M Keegan, S Ballard, G Lewis, SM Hasan, V Alexandrov, RP Tyer, I Todorov, P Wilson, M Alfredsson, GD Price, C Chapman, W Emmerich, S Wells, A Marmier, and Z Du S Parker. Collaborative tools in support of the eminerals virtual organisation. In *Proceedings of UK e-Science All Hands Meeting 2004*, pages 127–134, September 2004.

[24] MT Dove, M Calleja, R Bruin, J Wakelin, MG Tucker, GJ Lewis, SM Hasan, VN Alexandrov, M Keegan, S Ballard, RP Tyer, I Todorov, PB Wilson, M Alfredsson, GD Price, C Chapman, W Emmerich, SA Wells, A Marmier, SC Parker, and Z Du. The eminerals collaboratory: tools and experience. *Molecular Simulations*, 2005.

[25] MT Dove, M Calleja, J Wakelin, K Trachenko, G Ferlat, P Murray-Rust, NH de Leeuw, Z Du, GD Price, PB Wilson, JP Brodholt, M Alfredsson, A Marmier, RP Tyer, LJ Blanshard, RJ Allan, K Kleese van Dam, IT Todorov, W Smith, VN Alexandrov, GJ Lewis, A Thandavan, and SM Hasan. Environment from the molecular level: an escience testbed project. In *Proceedings of UK e-Science All Hands Meeting 2003*, pages 302–305, September 2003.

[26] MT Dove, LA Sullivan, AM Walker, K Trachenko, RP Bruin, TOH White, P Murray-Rust, RP Tyer, PA Couch, IT Todorov, W Smith, and K Kleese van Dam. Anatomy of a grid-enabled molecular simulation study: the compressibility of amorphous silica.

In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 653–660, September 2006.

[27] MT Dove, TO White, RP Bruin, MG Tucker, M Calleja, E Artacho, P Murray-Rust, RP Tyer, I Todorov, RJ Allan, K Kleese van Dam, W Smith, C Chapman, W Emmerich, A Marmier, SC Parker, GJ Lewis, SM Hasan, A Thandavan, V Alexandrov, M Blanchard, K Wright, CRA Catlow, Z Du, NH de Leeuw, M Alfredsson, GD Price, and J Brodholt. escience usability: the eminerals experience. In *Proceedings of UK e-Science All Hands Meeting 2005*, pages 30–37, September 2005.

[28] Z Du, VN Alexandrov, M Alfredsson, KF Austen, ND Bennett, MO Blanchard, JP Brodholt, RP Bruin, CRA Catlow, C Chapman, DJ Cooke, TG Cooper, MT Dove, W Emmerich, SM Hasan, S Kerisit, NH de Leeuw, GJ Lewis, A Marmier, SC Parker, GD Price, W Smith, IT Todorov, R Tyer, AM Walker, TOH White, and K Wright. A virtual research organization enabled by the eminerals minigrid: An integrated study of the transport and immobilisation of arsenic species in the environment. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 481–488, September 2006.

[29] I Foster and C Kesselman. Globus: A metacomputing infrastructure toolkit. *International journal of Supercomputer Applications*, 11(2):115–128, 1997.

[30] I Foster, C Kesselman, and S Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[31] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and HPJ. Von Reich. The open grid services architecture. Globus alliance whitepaper, 2005.

[32] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufmann, July 1998.

[33] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steve Tuecke. Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5:237–246, 2002.

[34] Fabrizio Gagliardi, Bob Jones, François Grey, Marc-Elian Bégin, and Matti Heikkurinen. Building an infrastructure for scientific grid computing: status and goals of the egee project. In *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences.*, volume 363, August 2005.

[35] Julian D. Gale and Andrew L. Rohl. The general utility lattice program (gulp). *Molecular Simulations*, 29:291–341, May 2003.

[36] Jeffrey Grethe, Chaitan Baru, Amarnath Gupta, Mark James, Bertram Ludaescher, Maryann E. Martone, Philip M. Papadopoulos, Steven T. Peltier, Arcot Rajasekar, Simone Santini, Ilya N. Zaslavsky, and Mark H. Ellisman. Biomedical informatics research network: Building a national collaboratory to hasten the derivation of new understanding and treatment of disease. *Studies in Health Technology and Informatics*, 112:100–109, May 2005.

[37] Alastair Hampshire and Gordon S. Blair. *JGrid: Exploiting Jini for the Development of Grid Applications*, volume 2604, pages 132–142. Springer, 2002.

[38] S Mehmood Hasan, Gareth J Lewis, Vassil N Alexandrov, Martin T Dove, and Matt G Tucker. Multicast application sharing tool for the access grid toolkit. In *Proceedings of UK e-Science All Hands Meeting 2005*, pages 433–440, September 2005.

[39] K. Karasavvas, M. Antonioletti, M.P. Atkinson, N.P. Chue Hong, T. Sugden, A.C. Hume, M. Jackson, A. Krause, and C. Palansuriya. Introduction to ogsa-dai services. *Lecture Notes in Computer Science*, 3458:1–12, May 2005.

[40] Massimo Lamanna. The lhc computing grid project at cern. In *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment; Proceedings of the IXth International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, volume 534:1-2, pages 1–6, November 2004.

[41] B. Lesyng, P. Bala, and D. Erwin. Eurogrid - european computational grid testbed. *Journal of Parallel and Distributed computing*, 63:590–596, May 2003.

[42] P Martin, D Spagnoli, A Marmier, S C Parker, D C Sayle, and G Watson. Application of molecular dynamics dl_poly codes to interfaces of inorganic materials. *Molecular Simulations*, 8:1079–1093, November 2006.

[43] H Meuer, E Strohmaier, and H Simon J Dongarra. Top 500 supercomputing sites, June 2006.

[44] Peter Morville, editor. *Ambient findability.* O'Reilly, 2005.

[45] P Murray-Rust. Chemical markup language. *XML principles, Tools and Techniques*, pages 135–147, 1997.

[46] R. J. Needs, M. D. Towler, N. D. Drummond, and P. R. C. Kent. Casino version 1.7 user manual, 2004.

[47] Erika J Palin and Martin T Dove. Investigation of al/si ordering in tetrahedral phyllosilicate sheets by monte carlo simulations. *American Mineralogist*, 89:176–184, 2004.

[48] H G Rice. Classes of recursively enumerable sets and their decision problems. *Trans. Amer. Math. Soc.*, 74:358–366, 1953.

[49] C Romming, HM Seip, and IM Aaneseno. Structure of gaseous and crystalline 2,2'-dichlorobiphenyl. *Acta Chemica Scanadanavia Series A - Physical and Inorganic Chemistry A*, 28 (5):507–514, 1974.

[50] RE Rundle, NC Baenziger, AS Wilson, and RA McDonald. The structures of the carbides, nitrides and oxides of uranium. *J. Am. Chem. Soc.*, 70(1):99–105, 1948.

[51] S S. Fleming and A. Rohl. Gdis: a visualization program for molecular and periodic systems. *Zeitschrift Fur Kristallographie*, 220:580–584, 2005.

[52] CI Sainz-Diaz, EJ Palin, MT Dove, and A Hernández-Laguna. Ordering of al, fe and mg cations in the octahedral sheet of smectites and illites by means of monte carlo simulations. *American Mineralogist*, 88:1033–1045, 2003.

[53] CI Sainz-Diaz, EJ Palin, A Hernández-Laguna, and MT Dove. Octahedral cation ordering of illite and smectite. theoretical exchange potential determination and monte carlo simulations. *Physics and chemistry of minerals*, 30:382–392, 2003.

[54] Ronald L Sass, Rosemary Vidale, and Jerry Donohue. Interatomic distances and thermal anisotropy in sodium nitrate and calcite. *Acta Crystallographica*, 10:567–570, 1957.

[55] V Schomaker and L Pauling. The electron diffraction investigation of the structure of benzene, pyridine, pyrazine, butadiene-1,3, cyclopentadiene, furan, pyrrole, and thiophene. *Journal of the American Chemical Society*, 61(7):1769–1780, 1939.

[56] M.D. Segall, P.J.D. Lindan, M.J. Probert, C.J. Pickard, P.J. Hasnip, S.J. Clark, and M.C. Payne. First-principles simulation: ideas, illustrations and the castep code. *Journal of Physics: Condensed Matter*, 14:2117, 2002.

[57] Catherine Soanes and A Stevenson. *Oxford Dictionary of English*. Oxford University Press, 2005.

[58] José M Soler, Emilio Artacho, Julian D Gale, Alberto Garcia, Javier Junquera, Pablo Ordejón, and Daniel Sánchez-Portal. The siesta method for ab initio order-n materials simulation. *Journal of Physics: Condensed Matter*, 14:2745–2779, 2002.

[59] David Stainforth, Jamie Kettleborough, Myles Allen, Matthew Collins, and Andy Heaps. Climateprediction.com: Distributed computing for public interest modelling research. *Computing in Science and Engineering*, 4/3, 2002.

[60] Robert D Stevens, Alan J Robinson, and Carole A Goble. mygrid: personalised bioinformatics on the information grid. *Bioinformatics*, 19(suppl_1):302–304, 2003.

[61] Shoaib Sufi and Brian Mathews. Cclrc scientific metadata model: Version 2, 2004.

[62] Z. Szotek, W. M. Temmerman, and H. Winter. Application of the self-interaction correction to transition-metal oxides. *Phys. Rev. B*, 47(7):4029–4032, Feb 1993.

[63] John Taylor. *e*Science definition; http://www.rcuk.ac.uk/escience/archive/phase1.asp, November 2000.

[64] I T Todorov and W Smith. Dl_poly_3: the ccp5 national uk code for molecular-dynamics simulations. *Philosphical Transactions of the Royal Society of London: MPES, Theme DEM issue*, 362:1835–1852, 2004.

[65] K Trachenko and MT Dove. Densification of silica glass under pressure. *Journal of Physics: Condensed Matter*, 14:7449–7459, 2002.

[66] OB Tsiok, VV Brazhkin, AG Lyapin, and LG Khvostantsev. Logarithmic kinetics of the amorphous-amorphous transformations in sio_2 and geo_2 glasses under high-pressure. *Physics Review Letters*, 80:999–1002, 1998.

[67] S Tsuneyuki, M Tsukada, H Aoki, and Y Matsui. 1st principles interatomic potential of silica applied to molecular dynamics. *Physics Review Letters*, 61:869–872, 1988.

[68] R Tyer, M Calleja, R Bruin, C Chapman, MT Dove, and RJ Allan. Portal framework for computation within the eminerals project. In *Proceedings of UK e-Science All Hands Meeting 2004*, pages 660–665, September 2004.

[69] RP Tyer, PA Couch, K Kleese van Dam, IT Todorov, RP Bruin, TOH White, AM Walker, KF Austen, MT Dove, and MO Blanchard. Automatic metadata capture and grid computing. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 381–384, September 2006.

[70] BWH van Beest, GJ Kramer, and RW van Santen. Force fields for silicas and aluminophosphates based on ab initio calculations. *Physics Review Letters*, 64:1955–1958, 1990.

[71] MC Warren, MT Dove, ER Myers, A Bosenick, EJ Palin, CI Sainz-Diaz, BS Guiton, and SAT Redfern. Monte carlo methods for the study of cation ordering in minerals. *Mineralogical Magazine*, 65(2):221–248, April 2001.

[72] Robert C Weast, editor. *Chemical rubber company handbook of chemistry and physics*. Chemical Rubber Company, 52nd edition, 1971-1972.

[73] Stephen Wells, Martin Dove, and Matthew Tucker. Reverse monte carlo with geometric analysis – rmc+ga. *Journal of applied crystallography*, 37:536–544, 2004.

[74] Stephen A Wells, Martin T Dove, and Matthew G Tucker. Finding best-fit polyhedral rotations with geometric algebra. *Journal of physics: condensed matter*, 14:4567–4584, April 2002.

[75] Stephen A Wells, Martin T Dove, Matthew G Tucker, and Kostya Trachenko. Real-space rigid-unit-mode analysis of dynamic disorder in quartz, cristobalite and amorphous silica. *Journal of physics: condensed matter*, 14:4645–4657, April 2002.

[76] Toby O White, Richard P Bruin, Jon Wakelin, Clovis Chapman, Daniel Osborn, Peter Murray-Rust, Emilio Artacho, Martin T Dove, and Mark Calleja. escience methods for the combinatorial chemistry problem of adsorption of pollutant organic molecules on mineral surfaces. In *Proceedings of UK e-Science All Hands Meeting 2005*, pages 773–780, September 2005.

[77] TOH White, P Murray-Rust, PA Couch, RP Tyer, RP Bruin, MT Dove, IT Todorov, and SC Parker. Application and uses of cml in the eminerals project. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 606–613, September 2006.

[78] TOH White, P Murray-Rust, PA Couch, RP Tyer, RP Bruin, IT Todorov, DJ Wilson, MT Dove, and KF Austen. Application and uses of cml within the eminerals project. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 606–613, September 2006.

[79] TOH White, RP Tyer, and RP Bruin. A lightweight scriptable web-based frontend to the srb. In *Proceedings of UK e-Science All Hands Meeting 2006*, pages 209–216, September 2006.

[80] R Zimmerman, C Weickhardt, U Boesl, and EW Schlag. Influence of chlorine substituent positions on the molecular structure and the torsional potentials of dichlorinated bephenyls: R2p1 spectra of the first singlet transition and am1 calculations. *Journal of Molecular Structure*, 327:981–997, 1994.