Usable grid infrastructures: practical experiences from the *e*Minerals project

MT Dove*, AM Walker, TOH White, RP Bruin, KF Austen, I Frame*, G-T Chiang*

Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ

** affiliated to the National Institute for Environmental eScience at above address*

P Murray-Rust

Unilever Centre for Molecular Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW

RP Tyer, PA Couch, K Kleese van Dam

STFC, Daresbury Laboratory, Warrington, Cheshire WA4 4AD

SC Parker, A Marmier†, C Arrouvel

Department of Chemistry, University of Bath, Bath BA2 7AY

† Now School of Engineering, Computer Science and Mathematics, University of Exeter, North Park Road, Exeter EX4 4QF

Abstract

We report experiences from the *e*Minerals project, in collaboration with the National Institute for Environmental *e*Science, in developing a highly-usable grid infrastructure to support simulation sciences. The focus has been on a close integration between compute, data and collaborative components, with the aim to make each of these highly usable for the scientists. Here we describe both general aspects of our approach and specific details of our toolset. The latter include the overall grid infrastructure, the data grid, the user interface, the job submission process, tools for creating and managing combinatorial or parameter-sweep studies, our XML tools, and our metadata tools, together with their role in running jobs, managing and sharing both data and information, and in enabling informed collaboration between researchers.

1. Introduction

This paper forms part of a workshop on usable grid infrastructures. The immediate question is what is meant by 'usable'? Or more precisely, usable to whom and for what? The question of usability is dependent on the norms of the community for which the infrastructure is being developed. In the case of the *e*Minerals project, we have been developing grid solutions for computational scientists working in several domains within the physical sciences. In general, these scientists are most comfortable working with shell tools rather than web portals or graphical interfaces and having some degree of control over their applications. That is not to say that they do not appreciate web or graphical interfaces to certain tools, but in general they do not want their whole work pattern to be wrapped up in a pre-packaged interface.

This type of scientist has traditionally worked to the following paradigm:

- the scientist will prepare input data files, job scripts and application (which may involve modifying the application for the particular study);
- he/she will then transfer them to the compute resource that will be used for running the job;
- the scientist will then log into the resource;
- he/she will then compile the application if an executable does not exist;
- the job will be submitted to an appropriate queue;
- some time later the scientist will log back onto the compute resource to monitor the status of the job;
- when the job is completed, the scientist will transfer the data back to his/her local computer (typically their desktop computer) to perform subsequent analysis.

We have recently been witnessing a transformation in the power of compute resources. Typically the function of the high capability compute resource is being reproduced by commodity clusters, and except for the most challenging of computational jobs the scientist now aims to handle hundreds or thousands of jobs within detailed parametric sweeps rather than being restricted to the small number of jobs that once was all that was feasible. *The traditional paradigm has come to the end of the road*. The usual approach now is for the computational scientist to write detailed scripts to handle the job submission and data management, although still following the traditional approach in terms of running jobs whilst logging into the compute resource and in terms of transferring data. This is now only "usable" for the fortunate few for whom writing scripts is one of their natural skills.

The traditional paradigm also suffers from one important aspect that was neglected until the advent of escience, namely it offers nothing to facilitate

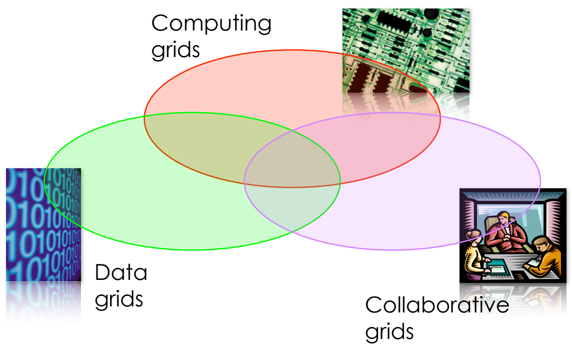


Figure 1. Representation of the three components of the grid infrastructure envisaged by the eMinerals project. The focus is on the areas of overlap between the different components.

collaboration. In fact, with the ability to run many jobs, even local collaborators (e.g. a researcher's supervisor) will have great difficulty in extracting information from the data obtained by one researcher. A common example of this problem is if a PhD student does not publish the results of their work before the end of their studentship; increasingly there will be only a remote chance of the supervisor being able to extract anything of value from the vast quantities of files left behind.

Grid computing offers the prospect of completely replacing this traditional paradigm, and the *e*Minerals project has sought to develop new approaches to exploit this. We have taken the view that the traditional paradigm contains no "best parts of" that are worth salvaging to create a hybrid with new grid methods. Instead we are now offering a genuinely usable infrastructure for computational scientists of the type we have described in our opening paragraph. Grid solutions solve the problem of management of many jobs and associated data files, but on top of this they offer the prospect of adding value to data through simultaneous automatic capture of metadata, and tools to enable extraction of information from data files. Such tools will not only make the management of data easier for the researcher, they will also make collaborative sharing of both data and information much easier. The *e*Minerals toolset is now able to make the processes of grid computing, data management, information delivery and collaboration easy for the computational scientist.

It is worth noting that one important part of the traditional paradigm has been the role of the log book. If a researcher has taken the log book seriously, it should be possible for a collaborator or supervisor to understand what the researcher did and thus pick up from the researcher at any point. With the capability of running many jobs, it is becoming increasingly untenable to expect fellow researchers to maintain comprehensive lab books. Moreover, the flow of data from simulation to analysis to final graph is increasingly entirely within the electronic realm. And in any case, it is hard to share lab books with collaborators who work in distant institutes. Within the *e*Minerals project we have have not sought to reproduce the lab book for escience. Instead we recognise, as will be discussed later, that many of the functions the lab book was intended to fulfil in a scientist's process can now be reproduced, or even improved upon, by various components of our toolset.

2. The *e*Minerals perspective

2.1 The *e*Minerals project

The *e*Minerals project has a primary focus on developing grid infrastructures to support computational scientists who perform simulations of the atomistic components of environmental processes. One exemplar is the simulation of the energies involved in the adsorption of organic pollutant molecules such as dioxins on mineral surfaces. These are challenging calculations in their own right, but the challenge is magnified through the fact that there are 76 congeners of the dioxin molecule, i.e. by representing the chemical formula as $(C_6H_xCl_{4-x})O_2(C_6H_yCl_{4-y})$, there are 76 different molecules that are formed by different values of x and y in the formula and different locations of the H and Cl atoms within the molecule. We need to be able to perform these calculations for all congeners for each mineral surface being studied, and there are many important mineral surfaces within soils.

We have developed a view of escience that comprises a close integration between compute grids, data grids and collaborative grids. as represented in Figure 1. The compute and collaborative grids overlap because there is a sharing of resources within the project, and because the compute grid tools (described below) have been developed through a strong collaboration between scientists and grid developers. The compute and data grids overlap because we have increasingly understood how grid computing now enables us to run so many jobs that we have required a completely new approach to data management in its broadest sense. The overlap between collaborative and data grids is based on the need for collaborators to share data in a way that is useful. Sharing data is actually only part of the issue; what scientists need to be able to do is share the information content of data, an issue we called "information delivery".

The *e*Minerals project involves scientists, application developers, computer scientists and grid specialists from seven institutes within the UK: the universities of Cambridge, Bath, and Reading, Birkbeck College, University College London, and the Daresbury eScience Centre.

2.2 Development of a usable grid infrastructure; the general approach

Over the duration of the *e*Minerals project, there have been significant changes in tools and opportunities. In order to capture the advantages these changes give, we have taken a 'bottom-up' approach to developing the *e*Minerals infrastructure and toolset. In such an approach, there is a close interaction between the scientist users and the developers. Alternative descriptions for this approach

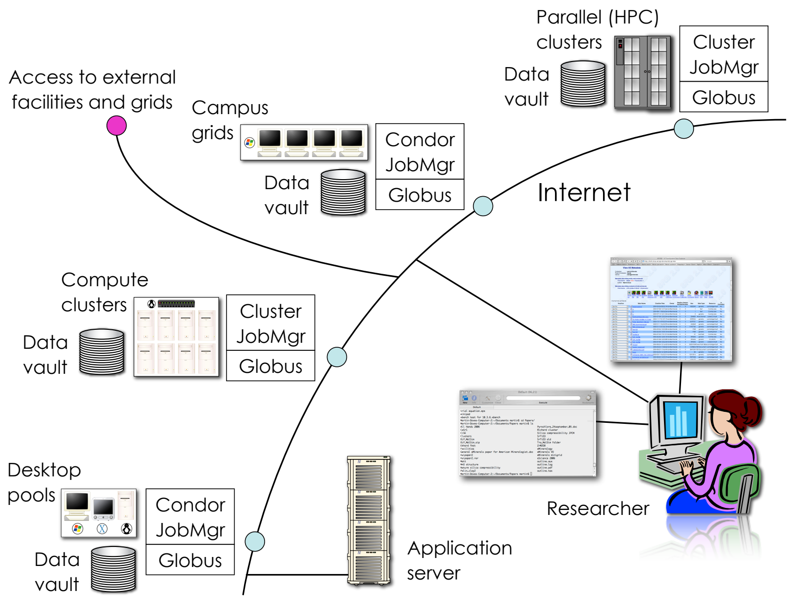


Figure 2. Representation of the eMinerals grid infrastructure. Compute resources include clusters and Condor pools. Globus is used to manage access to the grid resources (including user authentication). With the use of standard middleware tools (such as Globus and Condor), access to other grid resources and facilities is straightforward. The infrastructure contains an integrated data component with data vaults attached to the Globus gatekeepers. The application server provides access to metadata associated with the files in the data vaults and other associated databases, and gives support for various interface tools. The researcher is shown with representations of terminal command line and web-based interfaces to the grid infrastructure; the key to usability is in the interfaces.

might be 'democratic', 'anarchic' or 'chaotic'. Such an approach requires confident and informed leadership, and a committed teamwork approach, because people have to work to a common goal that is necessarily dynamic. Anticipating later discussion, we highlight one example of this sort of approach. Soon after the *e*Minerals project started we became aware of the work on the development of the Chemical Markup Language (CML). Using CML was not part of our original project plan, but over time our vision of the opportunities afforded by making CML a major part of our work became much wider, and we devoted many resources to incorporating CML into our work, including developing several tools that exploited CML, creating new CML-specific tools, adding CML capability to our simulation tools, and contributing to the development of CML. This aspect of our work has significantly enhanced all aspects of the vision represented in Figure 1; the point is that it was easy to adapt to this opportunity through the use of a bottom-up approach.

One aspect of the bottom-up approach is that users effectively control the development of tools. For example, if a tool doesn't meet the needs, it can be altered at an early stage, and in some extremes tools are abandoned if they do not attract interest. The big advantage of this is that the tools that are developed have usability built into them from the outset.

2.3 *e*Minerals user profile

Briefly, it is important to comment on the profile of the users we are developing against, because this provides the context for this paper. Our users are, to a person, very comfortable with command lines and shell tools. They are not necessarily capable of developing complex scripts, although they are able to adapt scripts to meet their needs. They are happy to use portals and graphical interfaces for specific tools, but they are very uncomfortable in having such interfaces for most of their working environment. Moreover, our users want to have complete control over their simulation applications, and are very uncomfortable working with applications that are wrapped as a service if it means that they lose direct control over the application. Even for applications they have not developed, they still want to retain the ability to modify them to a lesser or greater extent. A usable grid infrastructure for these scientists must ensure that users retain such a level of control over what they do. This is consistent with the idea of a bottom-up or democratic infrastructure, as opposed to a *provider/consumer* model.

The *e*Minerals scientists often need to perform large-scale combinatorial or parametric studies, which typically involve several hundreds of jobs in one go. In the traditional paradigm way of working discussed in the introduction, the scientist would work through the set of jobs one by one. Our scientists now expect submission of many jobs and the subsequent job and data management to be as easy as submitting a single job.

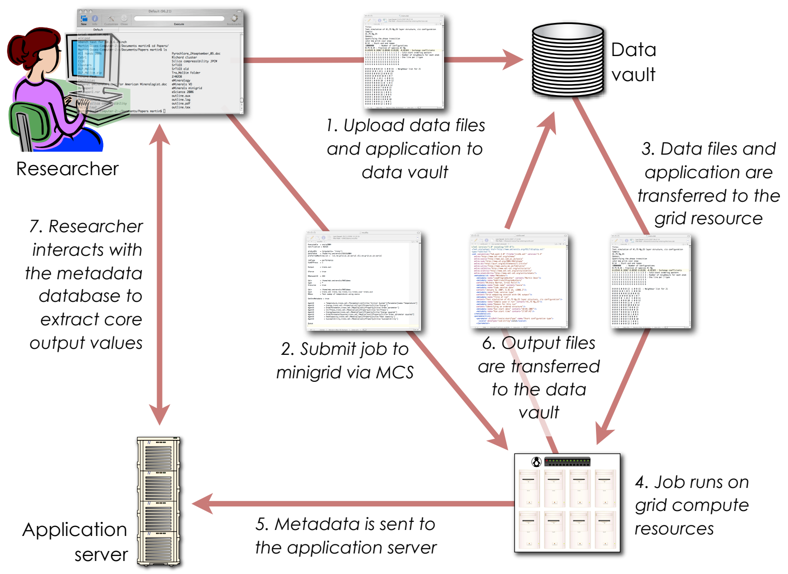
3. *e*Minerals grid infrastructure

3.1 The basics of the infrastructure

The *e*Minerals grid infrastructure is represented in Figure 2. This shows an integration of heterogeneous compute resources (the virtue of 'heterogeneous' being that users have very heterogeneous requirements) and data resources. The data resources are currently based on the Storage Resource Broker, (SRB) but work is in progress to develop a new webdav-based data grid resource. One key to the usability is in the use of standard middleware components, such as Globus and Condor. Another key is in the design of the interface seen by the user, and this will be discussed extensively below. One point to make here is that these have enabled the users to exploit additional facilities, such as the NW-Grid, as they become available.

3.2 The use of a data grid component

The *e*Minerals scientists have discovered that the use



Data vault
Researcher
1. Upload data files and application to data vault
3. Data files and application are transferred to the grid resource
7. Researcher interacts with the metadata database to extract core output values
2. Submit job to minigrid via MCS
6. Output files are transferred to the data vault
4. Job runs on grid compute resources
Application server
5. Metadata is sent to the application server

Figure 3. Representation of the processes associated with running jobs on the eMinerals minigrid. The researcher has one process for the submission of the job and the management of the data (steps 1,2 and 7). The job itself has another process associated with data transfer and metadata extraction (steps 3–6).

of a datagrid component, in our case the SRB, has made a significant qualitative change to the way that they work. The datagrid is closely integrated with the compute grid in one simple way, giving the user a new working paradigm (which contrasts with the paradigm described in the introduction):

- the scientist will prepare input data files, job scripts and application for many jobs;
- he/she will then transfer them to the data grid;
- the set of jobs will be submitted to the grid infrastructure from the user's desktop, with the jobs handling metascheduling, data transfer between the compute resource and data grid, and automatic collection of metadata;
- the user monitors the progress of jobs from their desktop;
- when the jobs are completed, the scientist can view the output files using our XML tools, and collate the core results from the metadata database.

This process is shown in Figure 3. The data grid is core to this process, providing a data staging tool that integrates easily into the user's desktop (the SRB provides something like a file system view of their data) and provides the user with complete control over their files. The qualitative change to our work patterns has several forms:

- we now expect to be able to share data with our collaborators, and we expect this to be easy to the point of being automatic;
- we now routinely and automatically produce complete archives of all files associated with a given study;
- the data grid provides a single place to deposit data, and this process is easy.

Figure 3 shows an additional aspect to the data component, namely the automatic capture of metadata. We discuss this in detail in a separate paper [3], but some aspects are discussed below.

3.3 Job submission interface: MCS

The primary user interface to job submission is the *my_condor_submit* (MCS) tool. In brief, MCS is a perl program that will enable a user to submit a job to a grid infrastructure. The user provides a simple file containing directives, as illustrated in Figure 4, and the job submission process is controlled by the information given within the file and then handled by the Condor-G wrapping of Globus. MCS has a close integration between the compute and data grids, in that it is designed specifically to handle the activity workflow described above. An earlier version of MCS has been described elsewhere [1], but it now has a number of new features which we will discuss here with reference to Figure 4. The user has control over a number of actions within the job:

- ability to nominate a number of grid resources to which MCS can submit the job (metascheduling);
- provide information on the type of resource (cluster or Condor pool, and number of processors if a cluster);
- select data locations (e.g. within the SRB) for the application, input and output files;
- instruct the job to automatically collect metadata from the output XML files.

With regard to the latter point, automatic metadata capture is an important aspect of usability, for discovery of data location and information on data, and as a primary interface to the core outputs contained within the data, again as discussed below.

3.4 Job submission process: RMCS

MCS requires users to have Globus and Condor installed on their desktops, which certainly mitigates against usability to some extent. Thus we have developed a web services wrapping of MCS called RMCS. In effect, the use of RMCS converts the two-tier client/grid model of MCS to a three-tier model of client, server and grid, as represented in Figure 5. The interaction between the client and RMCS server can be performed from the user's desktop without the need to install additional middleware tools, and the demands on the institute's firewall are light. These considerations are critical for making grids usable.

For people using shell tools, the RMCS package consists of a set of six shell command tools that interact with a central server via gSOAP. These

```
Executable   = ossia2004
notification = NEVER

globusRSL = (arguments= 'trans')
pathToExe = /home/bob.eminerals/OSSIA2004

preferredMachineList =  pbs1.uni.ac.uk cluster.college.ac.uk
jobType     = performance
numOfProcs = 1

Output      = trans.out

Sforce      = true
Sdirect     = true

Sdir        = /home/bob.eminerals/ossia/trans50
Sget        = *
Srecurse    = true

Sdir        = /home/bob.eminerals/ossia/trans50
Sput        = *

RDatasetID = 444
RDesc       = Sweep of temperature using ossia with 50:50 case
MetadataString = Model,"Fitted to SIESTA calculations"
GetEnvMetadata = true
AgentXDefault  = trans.xml

AgentX = Energy,trans.xml:/PropertyList[last]/Property[title='Energy']
AgentX = OrderParameter,trans.xml:/Module[last]/Property[title='Order parameter']
AgentX = PDF1,trans.xml:/Module[last]/Property[dictRef='ossia:PDF1']
AgentX = Stiffness,trans.xml:/Module[last]/Property[title='Stiffness']
AgentX = EnergySquared,trans.xml:/Module[last]/Property[title='Energy squared']
AgentX = HeatCapacity,trans.xml:/Module[last]/Property[title='Heat capacity']
AgentX = Susceptibility,trans.xml:/Module[last]/Property[title='Susceptibility']

Queue
```

Figure 4. Representative MCS input script.

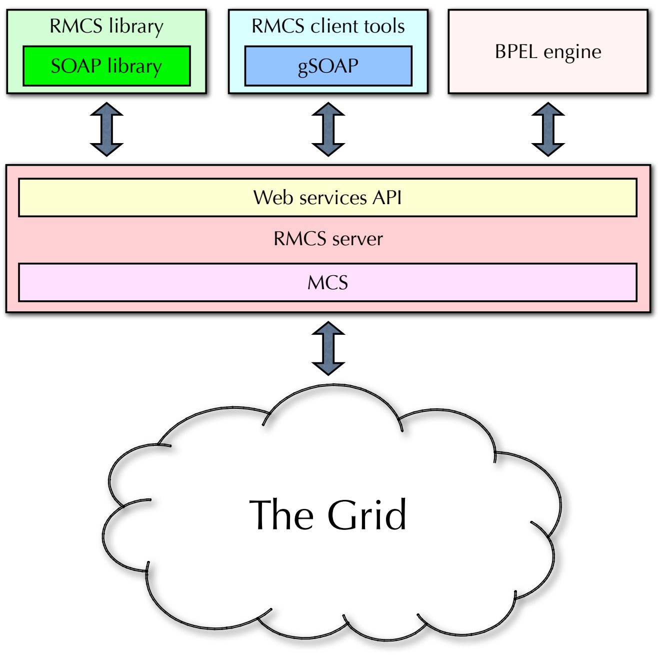


Figure 5: Representation of the three-tier infrastructure of the RMCS web services wrapping of MCS.

commands perform the following tasks:

- Submit an MCS job (*rmcs_submit*);
- Cancel an RMCS job (*rmcs_cancel*);
- Inspect the status of an RMCS job (*rmcs_status*);
- Remove an RMCS job from the status list (*rmcs_remove*) after it has run;
- Update the user's proxy credentials to the local Myproxy server in the event that the initial proxy has expired (*rmcs_update*);
- Change the RMCS password (*rmcs_passwd*).

The first three of these are self-explanatory. The only point to note is that the MCS file contains all the information that a job requires, so the *rmcs_submit* command requires no information other than the name of the MCS file. Once a job has been submitted, the RMCS infrastructure maintains a record of the status of a job that is updated regularly. To use the RMCS commands, the user first needs to upload his/her grid certificate credentials to a myproxy server. This task might be accomplished using the Globus *myproxy_init* command, but RMCS is designed to be run from computers on which Globus has not been installed. Instead, users can use the CCLRC Myproxy Upload tool for example, a java client tool with a graphical interface. The updated proxy will have a finite lifetime; the *rmcs_update* tool enables new a myproxy to be associated with a job.

3.5 Combinatorial job submission

As noted in the introduction, the real gain of grid computing (the "killer application"?) is the ability to perform large-scale combinatorial or parameter sweep studies. Usability requirements means that we need a tool that will accurately set up and submit many jobs at once. In fact there are two stages to this. The first is unavoidably bespoke to some degree or other. What is required is a tool to set up a set of input files, e.g. for a parameter sweep over temperature, we need a set of input files each containing a different value for the temperature input parameter. Our tools for this task also generate a simple configuration file that matches the input file names against job names.

The more general stage is to take the configuration file and turn that into jobs. For this we have developed a tool, called *Monty*, that is closely integrated with RMCS. *Monty* is a perl program, and requires two input files: a configuration file and an MCS template file. *Monty* also requires a directory containing any common files that are required to run the jobs. This might be, for example, a file containing an initial atomic configuration. When *Monty* is executed, it performs a number of tasks:

- It first generates a complete set of MCS files based on the name mapping provided in the configuration file;
- It then creates a complete set of collections within the SRB using the names within the configuration file, and then uploads the set of input files, including the common files, to these collections;
- Monty then submits all the MCS files using the *rmcs_submit* command.

Monty effectively turns combinatorial job submission into a single-command operation.

3.6 XML output files

In principle most users will not need to know much about the use of XML, but XML is critical for many aspects of the usability of our tools. The *e*Minerals project makes considerable use of CML. Most of our simulation codes are written in Fortran, and we have have written a set of libraries to enable Fortran codes to write output in XML format in general, and CML (and KML) format specifically, called FoX [3]. To use CML in our toolset, we have developed a way to structure CML output files. In particular, our CML output files are structured in blocks that contain metadata, input parameters, step-wise output, and summary output. Although most users will not worry about XML per se, the point for this paper is that as part of our toolkit we have made the use of XML relatively easy, including, through the use of FoX, creating XML output files.

3.7 Metadata management

The final component in *e*Minerals toolkit is metadata extraction and management. We have developed an infrastructure called the RCommands [3]. The RCommands framework consists of a back-end metadata database, a set of client tools, and an application server that maps the web service calls from the client tools onto a set of SQL calls. The client tools for the user consist of 10 shell commands for performing tasks such as listing metadata items, annotating datasets and data objects with metadata, and searching for datasets and data objects based on metadata. The metadata items include the URIs of the associated data objects.

The MCS example file (Figure 4) shows how users interact with the job and the XML output file to extract metadata. The directive *GetEnvMetadata = true* instructs the job to extract metadata from the job environment and from the metadata and parameter blocks of the CML output files. The remaining directives, those beginning with "AgentX", instruct the MCS job to extract specific metadata from the CML. In this example (which is extracted from a somewhat larger file) MCS is being instructed to collect items such as the average values of energy, order parameter and heat capacity to be stored as metadata. It is not common to collect output values as metadata, but there are at least two reasons why this is useful. First, by way of example, if a scientist has run studies of the adsorption of all congeners of dioxin on a set of minerals surfaces, the *Rsearch* command can be used to locate the calculation with the lowest adsorption energy. The second reason to collect output data values as metadata is to enable metadata to be used as a primary interface to the data, as we will now discuss.

3.8 Metadata as an interface to data: the *Rgem* client tool

It is useful to cite a user case: consider a parametric study in which we have calculated values of quantities such as energy for a wide range of input parameter values, such as temperature. Each calculation has been performed as a separate job, so that the core output is stored with many other quantities within a myriad of output files stored within the data grid. Rather than go to the output files, we can use the metadata directly to construct a table of output values, such as energy *vs.* temperature. We have developed the *Rgem* command to perform this task automatically, making the collation of data from many grid computations remarkably easy for the researcher. He/she specifies the data set that contains metadata for a set of job runs and the names of parameters to be extracted. *Rgem* then generates a table of data in space-separated format, which the researcher can then import into an analysis or visualisation program. This process is much easier than downloading all the files from the data archive and then parsing them all to extract the required information.

4. The *e*Minerals toolset and the collaborative grid

The work described so far in this paper concerns the focus that the *e*Minerals project has had on making compute and data grids usable. Now we consider how these tools also make collaboration much more

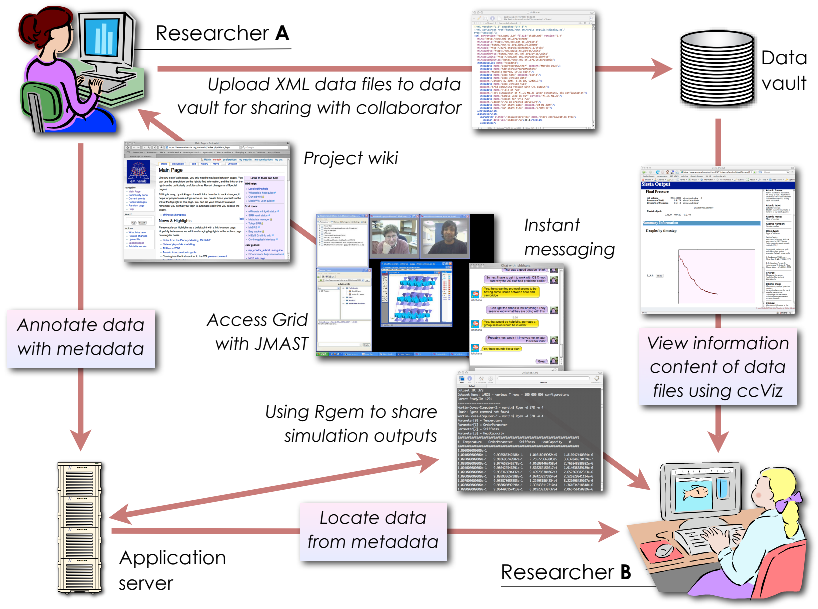

Researcher A
Upload XML data files to data vault for sharing with collaborator
Data vault
Project wiki
Instant messaging
Access Grid with JMAST
Annotate data with metadata
View information content of data files using ccViz
Using Rgem to share simulation outputs
Locate data from metadata
Application server
Researcher B

Figure 6. Representation of the the collaborative grid. Communication tools include the Access Grid with the JMAST application sharing tool, and instant messaging. Tools such as wikis allow for asynchronous and permanent communication. The collaborative data tools are based on the use of data grids and metadata tools. If the data files are in XML format, Researcher B can use information delivery tools such as ccViz [4] to read the information content of the shared data files without sharing all of the domain knowledge of Researcher A. Both researchers can use the metadata to construct tables of results without either researcher needing to parse output files. Researcher B can also use the metadata to locate the data files generated by Researcher A.

accessible. A key point is the central role of our use of CML for data representation.

The way that our tools aid collaboration is shown in Figure 6. This shows how one researcher (A) is collaborating with a fellow researcher (B) who wants to understand the results generated by the first researcher. The traditional approach, which frankly is completely inadequate in the era of grid computing, is for one researcher to send the collaborator a set of data files by email (or by posting them to an FTP or HTTP server) and then conducting a conversation by email to explain how to interpret the files. Using the toolset described above, we can now do a lot better.

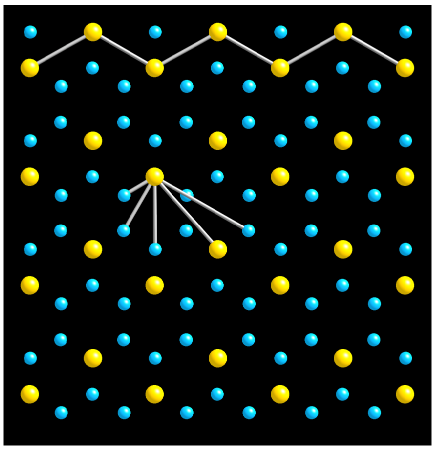
First, direct interaction between researchers is, in our experience, much better carried out using tools such as instant messaging, desktop Access Grid (with application sharing tools), and wikis. In fact we envisage that social networking sites will evolve to further enhance this collaborative interaction, and we are running a set of experiments in this area. For sharing data, our use of the SRB (or a comparable data grid infrastructure) automatically enables sharing of output files with collaborators. We have developed a web interface to the SRB, called *TobysSRB*, which not only allows for easier access to the data than afforded by other tools, but also provides a one-click transformation of the XML files to XHML pages using our *ccViz* tool [4]. This enables the output file to be viewed by the collaborator in a form that makes sense without the need for prior expert knowledge, by using a combined information- and user-centric representation of the information content with tables of data represented both in tabular form or plotted as SVG graphs on demand. Thus the second researcher is much better placed to understand the information generated by the compute jobs of the first researcher. The experience of the second researcher is further enhanced by the fact that she/he can find the data using the metadata contained within the RCommands framework, a process that required little action from the first researcher. Using the *Rgem* tool, both researchers can construct tables of data easily.

5. The role of the system and network managers

A lot of work to make grids usable relies heavily on the contributions of system and network managers. An example is in the area of security, specifically with regard to firewalls. The patterns of network communication required to support grid computing are significantly different from those of most researchers. The use of the Access Grid is one example, which we want to have available on all researchers' desktops. The underlying multicast technology is not needed on many networks and is therefore unfamiliar to many network administrators. Furthermore, specific firewall configurations are required for many of our other software and middleware tools, including Globus, Condor and the SRB. Active positive collaboration between network administrators and scientists is essential for escience to be usable. Within the lead author's institute we have demonstrated that such a collaboration is both feasible and consistent with running an institute network that simultaneously meets the needs of the researchers and the security requirements of the network administrators.

6. Conclusions

The primary message from this paper is that we have created a grid infrastructure for simulation scientists that is a close integration between compute, data and collaborative requirements, is complete in the sense that it covers the whole job life cycle from submitting jobs to a grid environment through to extracting the core information for analysis and sharing with collaborators, and is highly usable. Usability is an important feature, and has been designed into the development of the toolkit from the outset by our bottom-up policy of enabling scientists to work closely with the developers to set the overall



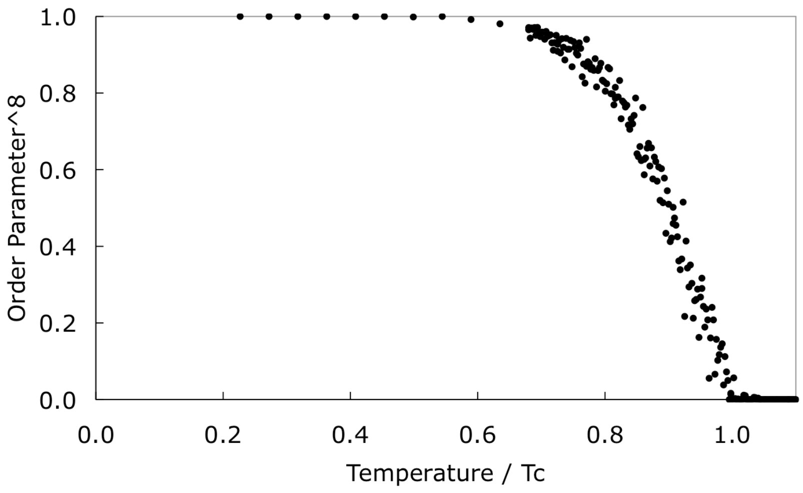


Figure 7. Example of a grid computation, namely a sweep through temperatures to study the ordering of cations in layer silicates using the Monte Carlo method. In this example the physics problem is represented by a set of interactions between neighbouring sites, with energies associated with different types of neighbour pairs (left). Each temperature is run on a different grid resource, as described in the text. Collation of data from the outputs of all runs is required in order to generate graphs showing the behaviour as a function of temperature. The graph on the right shows the degree of long-range order as a function of temperature raised to the power of 8; theory proposes that in the vicinity of the phase transition the degree of order should scale as $(1–T/T_c)^{1/8}$.

escience research agenda.

We show an example in Figure 7. This comes from a study of the ordering of cations in silicate minerals. In this example, the work was performed through a collaboration between two of the authors. Either of the collaborators would use the tools described in §3.5 to create and submit several hundred jobs at one time (a two-command process). The other collaborator could then find the location of the output files from the metadata, together within information about the jobs (such as values of input parameters, system size, number of steps). Both collaborators could check that different jobs were well-converged using the *ccViz* tool within *TobysSRB*, and both could produce tables of data for plotting, as in Figure 7, using the *Rgem* tool and the approach of treating metadata as the primary interface to data. The complete study involved the integration of compute, data and collaborative grids (Figure 1), the grid infrastructure represented in Figure 2, the process represented in Figure 3, and the collaboration cycle represented in Figure 6. The data shown in Figure 7 were obtained using many separate processors on a number of individual grid resources, with job creation and submission handled by one command each, with the data located through a simple browse through the metadata, and with one command required to collate the data for construction of the graphs.

Finally we note that the toolset described in this paper are quite generic, and can be used by other communities of researchers. For users of the NGS, some of the components are already available for people to use. For communities wanting to work with independent grid resources, we believe that the tasks for the system manager are no harder than setting up installations of Globus (which is needed on the various grid resources). The demands on the user in terms of installing tools on his/her own desktop are very light, consisting merely of copy a few executables and setting up the appropriate paths. The bigger challenge is actually learning a new way or working, rather than learning how to use the tools.

Acknowledgments

We are grateful to NERC for financial support (grant reference numbers NER/T/S/2001/00855, NE/C515698/1 and NE/C515704/1).

References

1. RP Bruin *et al*, “Job submission to grid computing environments”, *Proceedings of the Fifth UK eScience All Hands Meeting, 2006*, pp 754–761
2. TOH White et al. “Application and uses of CML within the eMinerals project”. *Proceedings of the Fifth UK eScience All Hands Meeting, 2006*, pp 606–613
3. RP Tyer *et al*, “Metadata management and grid computing within the *e*Minerals project”, *Proceedings of the Sixth eScience All Hands Meeting, 2007*
4. TOH White *et al*, “ A lightweight, scriptable, web-based frontend to the SRB”, *Proceedings of the Fifth UK eScience All Hands Meeting, 2006*, pp 209-216