Metadata management and grid computing within the *e*Minerals project

RP Tyer, PA Couch, Thomas V Mortimer-Jones, K Kleese van Dam, IT Todorov
STFC, Daresbury Laboratory, Warrington, Cheshire WA4 4AD

RP Bruin, TOH White, AM Walker, KF Austen, **MT Dove**
Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ

Abstract

We report a pragmatic approach for metadata management developed by the *e*Minerals project and which enables non-intrusive automatic metadata harvesting from grid-enabled simulation calculations. The framework, called the *RCommand framework*, gives users a set of unix line commands and a web interface to the metadata database. Harvesting of metadata relies on the use of XML for output data file representation, and new developments of the *RMCS* grid job submission tool incorporating the AgentX library. We report the use of the *Rgem* tool to use metadata as a direct interface to data to enable collation of data obtained from parameter-sweep grid studies.

1. Introduction

This paper describes a set of tools developed by the *e*Minerals project to facilitate metadata management and automatic harvesting of metadata from computer simulations. The key design requirements have been to ensure that the tools should be non-intrusive for the users, pragmatic in their design, and functionality that will tempt scientists to adopt them into their regular work patterns.

Metadata is of value to simulation scientists for a number of reasons. *First*, metadata enables scientists and their collaborators to identify the information content of a set of data files without needing to download them from where they are stored or archived (we discuss the data grid aspects later). It is not necessarily the case that output data files contain sufficient information to enable even the owner to properly identify the information content sufficiently; the exercise of generating metadata may actually force many code developers to think seriously about the issues of the vital supplementary information that should be provided along with the key output results. *Second*, metadata enables researchers and their collaborators to locate data files relatively easily, through directed searches or browsing through the metadata. Like the first point, the usefulness is critically dependent on the quality of the metadata. *Third*, metadata can provide a primary interface to data, which is one of the features that is particularly important for large data sets generated using grid computing methods. This is important both for the originator of the data and also for collaborators.

It is useful at this point to give a practical example. The *e*Minerals project, through which this work is being carried out, has a strong focus on studying environmental processes at a molecular level using a range of atomistic simulation methods (such as quantum mechanical tools and large-scale classical molecular dynamics methods). One study concerns molecular pollutants on mineral surfaces. In the case of the polychlorobiphenyl (PCB) system, chemical formula $C_{12}H_{10-x}Cl_x$, we are aiming to compare the energies of all 210 congeners (molecules with different values of x and different positions of the atoms) in isolation and in contact with a mineral surface, together with repeat calculations using different simulation models. Metadata is used to document the exact conditions of each simulation, effectively replacing the role of the logbook or README file; while these sufficed for a scientist working on a limited number of simulations, they fail when faced with the massive increase in capacity of these studies enabled by access to grid computing. This information enables collaborators to quickly understand the details of the various calculations, and to find the location of the data files. The metadata also enables collaborators to search for data files that match certain conditions, such as type of atomic basis set (i.e. the numerical description of the electronic wave function), or the orientation of a mineral surface, or even to search for the molecules or crystal faces with the lowest binding energies. Finally, when quantities such as the value of the energy for each run are stored as metadata, it is relatively easy to use our tools to collate the data for subsequent analysis.

In this paper we will describe the approach to metadata capture and management developed within the *e*Minerals project. We will give an outline of the broader context first, and then will describe the *RCommand* framework and client tools. We will discuss these are used for metadata collection, and will outline the important role of XML file representation. We will conclude with two case studies.

2. The *e*Minerals computing and data environment

The *e*Minerals project has focussed on developing grid computing infrastructures for simulation sciences [1]. Our approach has been to integrate grid computing (based on clusters and Condor pools) with grid data management methods based on the San Diego Storage Resource Broker (SRB). To make the interactive with the compute grid as easy to use as possible we have developed the *RMCS* tool for grid job submission; this is a web services wrapping of our established *my_condor_submit* tool. Users provide *RMCS* with a Condor-like set of directives within a single file, which control issues such as job metascheduling, locations of the collections on the SRB that contain input files and applications and where output files should be placed, and, most recently, how metadata are to be collected (discussed below).

Although we are currently using the SRB for data management (archiving, staging and sharing), we have chosen to not be specifically tied to the SRB for all data transactions and hence also for metadata management. For example, we are also experimenting with webdav based data grid approaches, and other users may want to have metadata tied to files stored on FTP or HTTP servers. Thus our metadata tools need to be decoupled completely from the specific data grid solutions we are using. This mitigated against using the SRB's metadata tools as our primary metadata infrastructure.

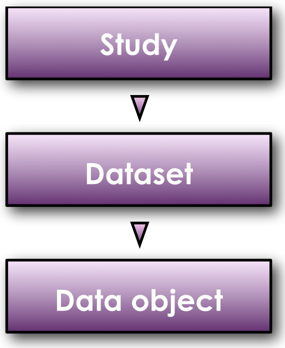


Figure 1. Graphical representation of the three metadata levels.

3. Metadata in the *e*Minerals project

3.1 Metadata organisation model

The model we use has three levels within which metadata are organised, Figure 1. The top level is the *study level*, which is self-explanatory. It is possible to associate named collaborators with this level, enabling collaborators to view and edit the metadata within a study. The next level is the *dataset level*. This is the most abstract level, and users are free to interpret this level in a variety of ways. Examples are given in Table 1. The third level is the *data object level*. This is the level that is associated with a specific URI, such as the address of a file or collection of files within the SRB, or an HTTP or FTP URL. The data object may be the files generated by a single simulation run, and/or the outputs from subsequent data analysis. In combinatorial or ensemble studies, it is anticipated that there will be many data objects associated with a single dataset, and it is the dataset level at which different types of calculations within a single study are organised.

This hierarchy of levels provides a high degree of flexibility for the individual scientists, each of whom will tailor it with for different work patterns. Some scientists may generate a large number of study levels, whereas other scientists might put all their work into one study. We remark that our tools can attach metadata to each of the three levels.

Table 1. Examples of how the study / dataset / data object levels have been used to organise data.

Study	Molecular dynamics simulation of silica under pressure	*Ab initio* study of dioxin molecules on clay surface
Data set	Identified by the number of atoms in the sample and the interatomic potential model used	Identified by number of chlorine atoms in the molecule and the specific surface site
Data object	Collection on the SRB containing all input and output files	Collection on the SRB containing all input and output files

3.2 Metadata to capture

Typically it is expected that metadata associated with the study and dataset levels will be added by hand, with the automated metadata capture to be provided at the data object level (although we provide tools for metadata to be automatically captured for datasets). We define five types of metadata to capture:

Simulation metadata: information such as the user who performed the run, the date, the computer run on etc.

Parameter metadata: values for the parameters that control the simulation run, such as temperature, pressure, potential energy functions, cut-offs, number of time steps etc.

Property metadata: values of various properties calculations in the simulation that would be useful for subsequent metadata searches, such as final or average energy or volume.

Code metadata: information that the code developers deemed useful when creating the code to enable users to determine what produced the simulation data, such as code name, version and compilation options.

Arbitrary metadata: strings that the user deems important when running the study to enable later indexing into the data, such as whether a calculation is a test or production run.

4. The *RCommand* metadata framework

4.1 Architecture

The metadata framework we have developed is based on a three-tied architecture, as shown in Figure 2. The three tiers are:

Client: The client layer seen by most users is a set of binary tools written in C using the gSOAP library (the RCommands shell tools described below). The motivation for using C was the requirement that the tools be as self-contained as possible so they can easily be executed server side via Globus. Other client tools can be built onto this layer.

Application Server: The application server is written in Java using Axis as the SOAP engine and JDBC for database access. The code essentially maps from procedural web service calls to SQL appropriate for the underlying database [2].

RDBMS: The backend database is served by a Oracle 10g RAC cluster. Although Oracle is used, there is no requirement for any Oracle specific functionality.

One of the main reasons for using a three tier model is that the backend database is behind a

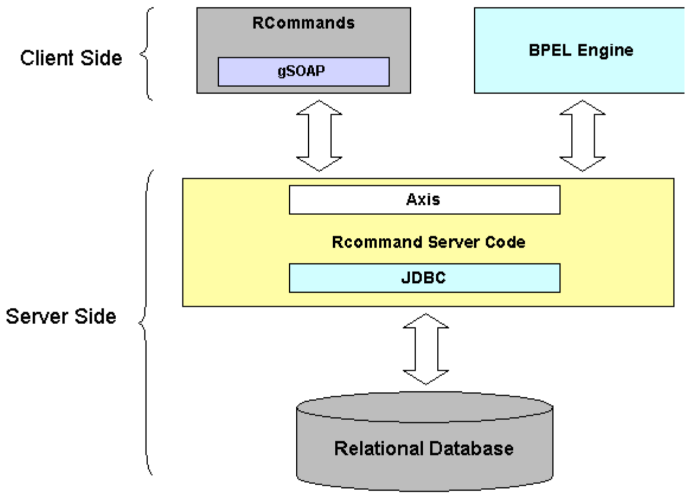


Figure 2. The three-tier architecture of the RCommand metadata framework.

firewall and thus cannot be accessed directly from client machines. The application server provides the interface between the user's tools and the metadata database. SOAP messages are sent to the application server from the cient tools via SSL-encrypted HTTP. The application server is authenticated using its certificate while the client requests are authenticated using usernames and passwords.

The use of web service technology is advantageous as it allows all the network related code to be auto-generated. On the server, the Axis SOAP engine is configured to expose specified methods of certain classes as RPC web services. The corresponding client code is generated on the fly by gSOAP from the WSDL file produced by Axis. In addition to the rapid development afforded by the use of web services, there is an additional requirement to allow the server side functionality to be exploited by members of the project interested in web services workflows [3].

4.2 The RCommand shell tools

We have developed a set of scriptable unix line commands as the primary client interface to the metadata database. These enable users to that perform functions such as uploading metadata to the metadata database, and listing metadata items. The various comments, which we colloquially call the *RCommands*, are defined in Table 2. To use these commands, the user needs a directory called `.rcommands`, which contains a configuration file with information about the user's metadata account and the location of digital certificates.

4.3 Metadata Manager: the web interface to the metadata database

The RCommand shell commands were initially written to provide tools that can be used in

Table 2. The ten RCommand unix line commands.

RCommand	Action
`Rinit`	Starts an RCommand session by setting up session files
`Rpasswd`	Changes the password for access to the metadata database
`Rcreate`	Creates study, dataset and data object levels, associating the lower levels with the level immediate above, adding a name to each level, adding a metadata description and topic association in the case of creating a study, and associating a URI in the case of creating a data object.
`Rannotate`	Adds metadata. In the case of studies or datasets, this enables a metadata description, and in the case of datasets and data objects it also enables metadata name/value pairs. It also enables more topics to be associated with a study.
`Rls`	Lists entities within the metadata database. With no parameters, it lists all studies, and with parameters it will list the entries within a study or dataset level. It can also be used to list all possible collaborators or science topics.
`Rget`	Gives the metadata associated with a given study, dataset or data object. In the case of a study, it can also list associated collaborators and science topics.
`Rrm`	Removes entities or parameters from the metadata database.
`Rchmod`	Add or remove co-investigators from a study.
`Rsearch`	Search the metadata database, tuned to search within different levels and against descriptions, name/value pairs and parameters.
`Rexit`	Ends an RCommand session, cleaning up session files.

scripts, but nevertheless they give scientists a useful interface to the metadata database. However, there are cases when a web interface is better, particularly when requiring a graphical overview that cannot be provided by a unix shell interface. We have developed a web interface to the metadata data based called the *Metadata Manager* (*MDM*, Figure 3). This provides an initial overview of the study level, from which the user can drill down into the various layers. The user can perform a number of the functions that are provided by the RCommand shell tools.

Although some users have clear preferences for either portal interfaces or command line tools, it is more natural to use a web interface for the high level (study) metadata and the command line tools via scripts for the fine grained (parameter) metadata. However, the same functionality is exposed via the portal, web service API and command line tools. This allows the users the flexibility to fit the metadata functionality into their preferred working paradigm, rather than requiring the user to adopt their working methods to fit into the technology.

The *MDM* design uses the JSP Model 2 architecture, which is based on the Model-View-Controller (MVC) pattern. The majority of the code in the Model layer is common to both the RCommand shell tools and the *MDM*. Hence, as with the shell tools, the database connectivity is provided using the JDBC libraries.

5. Collecting metadata

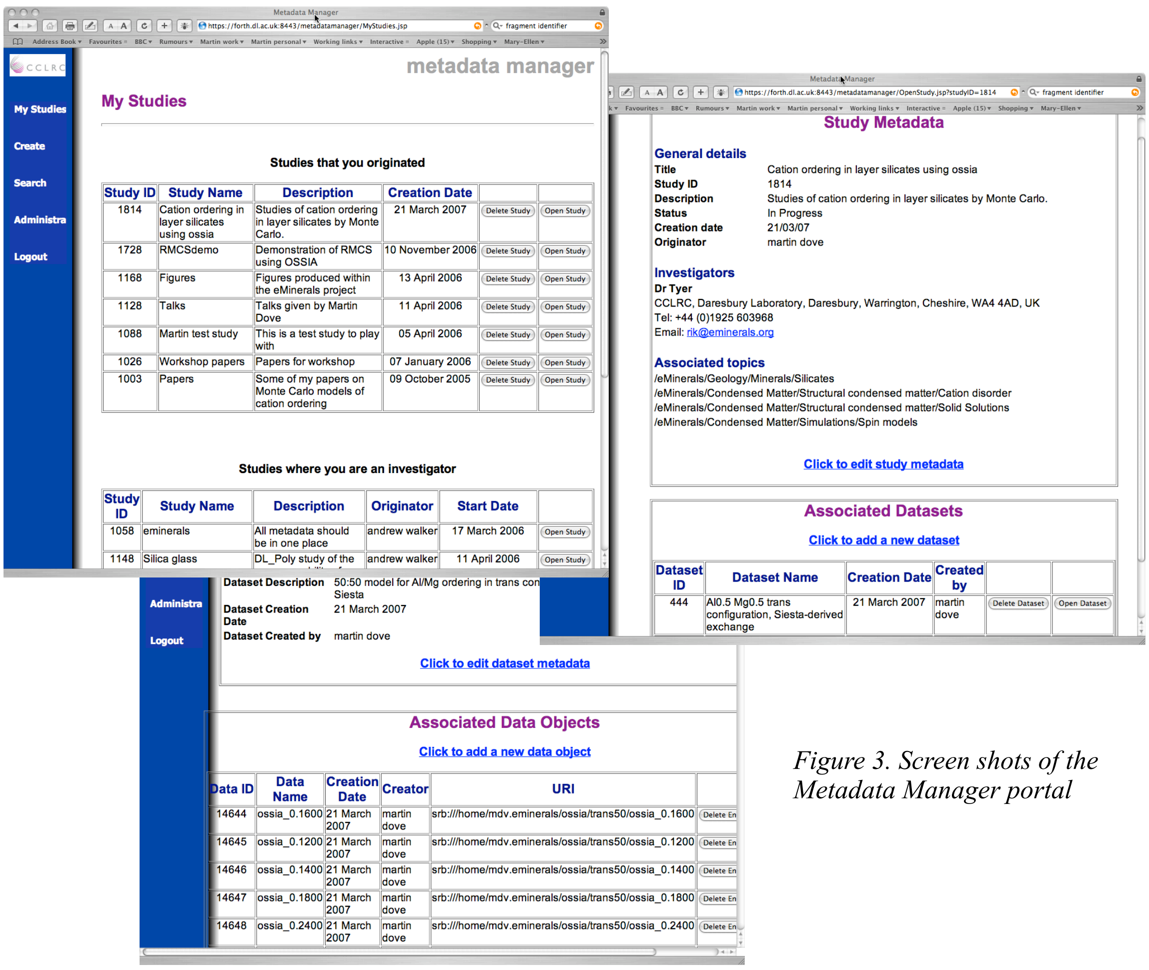
5.1 The role of XML in output files

Much of the metadata we collect is harvested from output data files. To facilitate this, we have enabled our key simulation programs to write the main output files in XML. Specifically we use the Chemical Markup Language [4]. The vast majority of our simulation codes have been written in Fortran, and we have developed a Fortran 95 library called FoX to provide routines for writing well-formed XML [4].

CML specifies a number of XML element types for representing lists of data. We make use of three of these:

`metadataList`: This list contains certain general properties of the simulation, such as code version;

`parameterList`: This list contains parameters associated with the simulation;



My Studies

Studies that you originated

Study ID	Study Name	Description	Creation Date		
1814	Cation ordering in layer silicates using ossia	Studies of cation ordering in layer silicates by Monte Carlo.	21 March 2007	Delete Study	Open Study
1728	RMCSdemo	Demonstration of RMCS using OSSIA	10 November 2006	Delete Study	Open Study
1168	Figures	Figures produced within the eMinerals project	13 April 2006	Delete Study	Open Study
1128	Talks	Talks given by Martin Dove	11 April 2006	Delete Study	Open Study
1088	Martin test study	This is a test study to play with	05 April 2006	Delete Study	Open Study
1026	Workshop papers	Papers for workshop	07 January 2006	Delete Study	Open Study
1003	Papers	Some of my papers on Monte Carlo models of cation ordering	09 October 2005	Delete Study	Open Study

Studies where you are an investigator

Study ID	Study Name	Description	Originator	Start Date	
1058	eminerals	All metadata should be in one place	andrew walker	17 March 2006	Open Study
1148	Silica glass	DL_Poly study of the	andrew walker	11 April 2006	Open Study

Metadata Manager
https://forth.dl.ac.uk:8443/metadatamanager/OpenStudy.jsp?studyID=1814
fragment identifier

Study Metadata

General details

Title	Cation ordering in layer silicates using ossia
Study ID	1814
Description	Studies of cation ordering in layer silicates by Monte Carlo.
Status	In Progress
Creation date	21/03/07
Originator	martin dove

Investigators

Dr Tyer
CCLRC, Daresbury Laboratory, Daresbury, Warrington, Cheshire, WA4 4AD, UK
Tel: +44 (0)1925 603968
Email: rik@eminerals.org

Associated topics

/eMinerals/Geology/Minerals/Silicates
/eMinerals/Condensed Matter/Structural condensed matter/Cation disorder
/eMinerals/Condensed Matter/Structural condensed matter/Solid Solutions
/eMinerals/Condensed Matter/Simulations/Spin models

Click to edit study metadata

Associated Datasets

Click to add a new dataset

Dataset ID	Dataset Name	Creation Date	Created by		
444	Al0.5 Mg0.5 trans configuration, Siesta-derived exchange	21 March 2007	martin dove	Delete Dataset	Open Dataset

Dataset Description	50:50 model for Al/Mg ordering in trans con Siesta
Dataset Creation Date	21 March 2007
Dataset Created by	martin dove

Click to edit dataset metadata

Associated Data Objects

Click to add a new data object

Data ID	Data Name	Creation Date	Creator	URI	
14644	ossia_0.1600	21 March 2007	martin dove	srb:///home/mdv.eminerals/ossia/trans50/ossia_0.1600	Delete En
14645	ossia_0.1200	21 March 2007	martin dove	srb:///home/mdv.eminerals/ossia/trans50/ossia_0.1200	Delete En
14646	ossia_0.1400	21 March 2007	martin dove	srb:///home/mdv.eminerals/ossia/trans50/ossia_0.1400	Delete En
14647	ossia_0.1800	21 March 2007	martin dove	srb:///home/mdv.eminerals/ossia/trans50/ossia_0.1800	Delete En
14648	ossia_0.2400	21 March 2007	martin dove	srb:///home/mdv.eminerals/ossia/trans50/ossia_0.2400	Delete En

Figure 3. Screen shots of the Metadata Manager portal

`propertyList`: This list contains properties computed by the simulation.

It is usual to have more than one of each list, particularly the `propertyList`. Clearly these lists correspond closely to the metadata types described earlier in this paper. Examples are shown in Figure 4.

5.2 Automatic metadata capture within a grid computing environment

As described in the introduction, the *e*Minerals project scientists run their simulations within the *e*Minerals minigrid using the *RMCS* tool [1]. This is a generic tool that will work on any compute grid infrastructure that has Globus and the SRB and metadata client tools installed (eg the National Grid Service).

As noted above, *RMCS* has a Condor-like interface for the user. It has the standard Condor commands for running jobs, with additional commands for managing data within the SRB. It now has additional commands for accessing the metadata database.

Here we give a brief introduction to the metadata capture processes. *RMCS* deals with four different types of metadata capture:

1. An arbitrary text string specified by the user.
2. Environment metadata automatically captured from the submission and execution environment including details such as machine names, submission and completion dates, executable name, etc.
3. Metadata captured is extracted from the first `metadataList` and `parameterList` elements described in the previous section.
4. Additional metadata extracted from the XML documents. These specifications take the form of expressions with a syntax similar to XPath expressions. These are parsed by MCS and broken down into a single term used to provide the context of the metadata (such as '`FinalEnergy`') and a series of calls to be made to the AgentX library [5].

The AgentX library calls, made from *RMCS* as a result of the user specified metadata expressions, query documents for data with a specific context. For example, AgentX could be used to find the total energy of a system calculated during a simulation. In this case (Figure 5), the user specified expression might have the form:

```
AgentX = FinalEnergy,
  output.xml:PropertyList
```

```
<?xml version="1.0" encoding="UTF-8"?>
<cml xmlns="http://www.xml-cml.org/schema"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<metadataList>
 <metadata name="identifier" content="DL_POLY version 3.06 / March 2006"/>
</metadataList>

<parameterList title="control parameters">
 <parameter title="simulation temperature" name="simulation temperature"
 dictRef="dl_poly:temperature">
 <scalar dataType="xsd:double" units="dl_polyUnits:K">50.0</scalar>
 </parameter>
</parameterList>

<propertyList title="rolling averages">
 <property title="total energy" dictRef="dl_poly:eng_tot">
 <scalar dataType="xsd:double" units="dl_polyUnits:eV_mol.-1">-2.7360E+04
   </scalar>
 </property>
</propertyList>

</cml>
```

Figure 4. Extracts of a CML output file, showing examples of the `metadataList`, `parameterList` and `propertyList` containers.

```
[title='rolling
averages'].Property
[dictRef='dl_poly:eng_tot'].value
```

The term providing the name of the metadata item is '`FinalEnergy`' and the document to be queried is `output.xml`. The string following '`output.xml:`' is parsed by *RMCS* and converted to a series of AgentX library calls. In this example, AgentX is asked to locate all the data sets in `output.xml` that relate to the concept '`property`' and which have the reference '`dl_poly:eng_tot`' (the average energy of a system in the context of the DL_POLY simulation code). The value of this property is extracted and associated with the term `FinalEnergy`. The RCommand shell tools are then used to store this name-value pair in the metadata database.

AgentX works with a specification of ways to locate data in documents (such as a CML document) that have a well defined content model. There are two components to the AgentX framework:

1. An ontology that specifies terms relating to concepts of interest in the context of this work. These terms relate to classes of real world entities of interest and to their properties. The ontology is specified using the Web Ontology Language (OWL) and serialised using RDF/XML. The Protégé Ontology Editor and Knowledge Acquisition System has been used for its development.
2. The second component is the mappings, which are used to relate terms in the ontology to document fragment identifiers. For XML documents, these fragment identifiers are XPointer expressions that may be evaluated to locate data sets and data elements in the documents. Each format is associated with its own set of mappings, and a set has been developed for *e*Minerals use of the Chemical Markup Language. The mappings are also serialised using RDF/XML.

AgentX is able to retrieve information from arbitrary XML documents, as long as mappings are provided. Such mappings exist for a number of codes (e.g. VASP, SIESTA and DL_POLY), and the number of concepts involved is being increased. In addition, mappings are under development for other codes.

Because the content of the data, for AgentX's purposes, resides entirely in the mapping between the concepts and the document structure (rather than solely in the structure itself), we have been able to design a more efficient representation for large datasets (such as DL_POLY configurations containing more than 10^6 atoms) at the expense of losing the contextual information from the document itself compared to standard XML.

5.3 Post-processing using the RParse tool

Although by design of its implementation, XML output will capture all information associated with the inputs and outputs of a simulation, it is inevitable that the automatic tools may miss some of the metadata. The nature of research work dictates that it is not always obvious at the start of a piece of work what properties are of most interest. Thus we

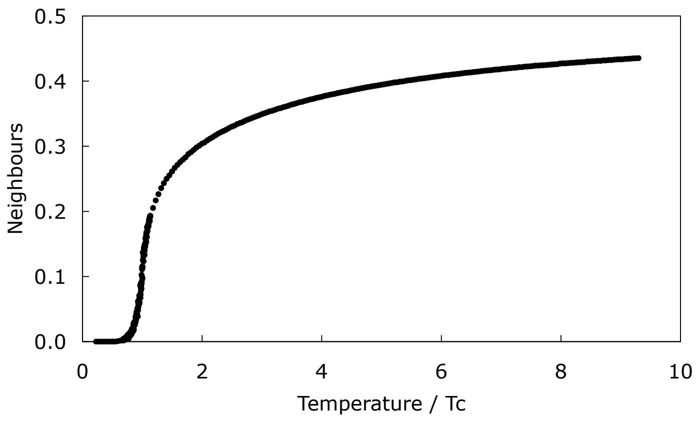


Figure 5. Example of one data set obtained using the Rgem command; in this case we plot the short-range order for cation ordering in a layer silicate as a function of temperature.

have developed a tool to scan over the XML files contained within the data objects in a single dataset to extract metadata based on CML parameters.

The *Rparse* tool allows automatic metadata ingestion after the simulation has finished and the data have been stored within the datagrid. The tool will trawl specified collections within SRB space, download key XML output files and harvest metadata for the user. This metadata is then inserted into a specified study/dataset within the metadata database along with the URI pointing to the relevant datafile within the SRB. This functionality is achieved by using the *Scommands* (SRB unix shell client tools), the RCommand shell tools, and the AgentX library. The user needs to specify a root collection in SRB, output files of interest, AgentX query expressions, and the dataset into which the metadata is to be inserted. Using this tool, it has been possible to automatically insert metadata for hundreds of XML files that where generated prior to the MCS metadata functionality being available.

6. Metadata as a primary interface to data

In the introduction we cited three uses of metadata. The first two of these, namely to provide information about data and to locate data, are common reasons for collecting good metadata, and the tools we have described above work well for this. The third reason is to provide a primary interface to data, and we now describe what this means together with a tool we have developed for this.

Let us consider the example of a study in which we compute properties of a material, such as energy or volume, for many different temperatures. The computations for each different temperature are performed as separate jobs on different grid resources, and each has its own set of output files (including the CML output file) stored in different collections within the SRB or another data archive. The task facing the researcher is to collate all the data to produce a table of energy and volume as a function of temperature. The traditional approach is to scan over all the output files to extract the input temperature and the final or average values of energy and volume. The main overhead for the researcher is to download all the files from the data grid and manage the data locally. However, the task of scanning the output files for the core information was already performed with the collecting of metadata, either at the time of creation within the *RMCS* process or subsequently using the *Rparse* tool, and the metadata database is playing the role of a cache for these data items.

We have developed a tool, called *Rgem*, that uses the codebase of the *RCommands*, to scan over the metadata associated with a set of data objects contained within a dataset and extract output parameter values from the metadata database. In the above example, the user would tell *Rgem* to collate the values of temperature, energy and volume from the metadata associated with all data objects within a given dataset. *Rgem* then returns a table of data in space-separated format, which the user can either copy into a file or import into an analysis/ visualisation application.

Rgem has a significant impact in the usability of grid resources. The task of accurately collating all the core output values from parameter-sweep studies is not trivial to set up and run, but *Rgem* is sufficiently general that using the metadata as the interface to the data reduces this task to running a single shell command, and the results are returned to the user nearly instantaneously.

7. Case study

7.1. Cation ordering in silicates

The process of cation ordering in layer silicates (such as clays) is performed using the Monte Carlo method, with one simulation of each temperature performed on a separate grid resource. We collected as metadata input parameters such as temperature, system size, number of steps, and interatomic interaction parameters, and output average values such as energy, heat capacity, short-range and long-range order parameters. The simulations were performed as a collaboration between two of the authors, who are based in different institutes. The metadata were used to enable the two collaborators to understand in detail the different runs each other had performed (e.g. when changing the system size). This is an example where *Rgem* was used to collate tables of output data from the metadata. An example result is shown in Figure 5; the number of points on the graph shows the extent of the data collation problem.

7.2. Simulation studies of organic molecules on mineral surfaces

As discussed in the introduction, we are running detailed studies of the adsorption of organic pollutant molecules on mineral surfaces. We run a script to create all the input files for the optimisation of a given set of molecules (using a quantum mechanics code), the docking of these molecules onto a chosen surface and the calculation of the basis set superposition error (BSSE). At each step the input files are uploaded onto the *e*Minerals data grid before the calculation is scheduled to run using *RMCS*. Throughout this process metadata are added to the metadata database.

In this example, we deposit all data objects in a single data set and a single study. As well as simplifying scripting by reducing the number of logical locations that must be stored and used to set up calculations, this provides a useful test of the database search capability. Data objects are created in two ways. For each set of input data created a data object is created by calling the RCommand shell tools from the desktop resource. This object points to the SRB collection that contains the input files, and includes the time and date of creation, the reason for the run (molecule optimisation in vacuum, relaxation on a surface, or one of several calculations needed to correct for the BSSE) and the fact that the job has been submitted. The second data object is created by *RMCS* on successful completion of the calculation. This data object is related to the results of the calculation and points at a collection in the SRB where these results reside (in principle this could be the same collection as the input files, but to avoid pollution of the input we create a sub collection called “results” for each calculation). Metadata stored at this stage includes the environment where the job ran and key parameters of the calculation collected from the XML output of the run.

8. Summary points

- We have developed a production-level infrastructure that integrates data, compute and metadata functionality. At the heart of this metadata functionality is the RCommand scriptable shell tools for metadata manipulation. These tools, coupled with the *RMCS* framework and AgentX, have enabled the automatic harvesting of metadata with no additional overhead for the scientists.
- We have provided portal and client tools to allow the users to search and manipulate their metadata. The use of a web service API for the server side components has further enabled the metadata functionality to be integrated into existing client tools with minimal effort.
- We have shown that the metadata database can be automatically populated with little effort and in a non-intrusive way.
- While there is a clear semantic difference between metadata and data, some of our experience suggests that for practical purposes it is sensible to blur the distinction. For example, using a final energy of –1.15 eV as metadata enables the user to search for datasets with energies lower than –1 eV.
- The *Rgem* tool enables researchers to use the metadata as a primary interface to data, enabling them to easily collate data from many separate jobs within a parameter-sweep study performed on a compute grid infrastructure.

Acknowledgement

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

References

1. MT Dove *et al*. “Usable grid infrastructures: practical experiences from the eMinerals project”. *Proceedings of UK eScience All Hands Meeting 2007*
2. M Doherty, K Kleese, S Sufi. “Database Cluster for e-Science”. *Proceedings of UK eScience All Hands Meeting 2003*, pp 268–271
3. C Chapman *et al*. “Simple Grid Access using the Business Process Execution Language”. *Proceedings of UK eScience All Hands Meeting 2006*, pp 377–380
4. TOH White *et al*. “Application and uses of CML within the eMinerals project”. *Proceedings of UK eScience All Hands Meeting 2006*, pp 606–613
5. PA Couch *et al*. “Towards Data Integration for Computational Chemistry”. *Proceedings of UK eScience All Hands Meeting 2005*, pp 426–432