Experiences in Developing a Node of an International Computational Physics Data Grid

Paul Coddington[1,2], Gerson Galang[2], Waseem Kamleh[1,3], Derek Leinweber[3], Sam Moskwa[1], Julia Patterson[1], Qiang Wang[2], Andrew Wendelborn[2], Shunde Zhang[1], Qunfang Zhang[1]

1. South Australian Partnership for Advanced Computing, University of Adelaide, Adelaide SA 5005, Australia
2. School of Computer Science, University of Adelaide, Adelaide SA 5005, Australia
3. Special Research Centre for the Subatomic Structure of Matter (CSSM) and Department of Physics, University of Adelaide, Adelaide SA 5005, Australia

paul.coddington@adelaide.edu.au

Abstract

The International Lattice Data Grid (ILDG) is an international collaboration that creates standards to enable sharing of data produced by Lattice Quantum ChromoDynamics (QCD) simulations, which are very computationally expensive. In this paper we summarize our experiences in developing an Australian node of the ILDG, including implementing a program to generate metadata conforming to the QCDml XML schema developed by the ILDG, storing the metadata in an XML database, developing metadata catalog and file catalog services using standard web service interfaces, providing data download using an SRM interface, and implementing a web portal to the ILDG. One of the problems we encountered in this work was that existing metadata catalog software was not able to meet the requirements of this project. Although this work is specific to Lattice QCD data and the ILDG specifications, most of the main issues in developing the data grid node are quite general and not specific to the type of data.

Keywords: Data grid, metadata catalog, simulation data.

1 Introduction

Quantum ChromoDynamics (QCD) is the quantum theory of quarks and gluons, which are the fundamental constituents of matter that interact to create sub-atomic particles such as protons and neutrons. Accurate predictions from QCD can typically only be made using very computationally intensive numerical investigations. This is done using an approach known as Lattice QCD, in which space-time is discretized as a 4D mesh, or lattice, of points. A discretized model of the QCD action, which describes the quarks and gluons and their interactions, is solved using a Monte Carlo simulation.

Each Monte Carlo run is done for a particular choice of action, and for specific parameter values such as quark mass and interaction strength, and generates an ensemble of sample configurations of the possible state space. Different analysis programs are then run on these configurations to generate results that can be compared to experiment, or predict the results from new experiments.

The size of a single Lattice QCD configuration data file is on the order of hundreds of Mbytes. An ensemble will have on the order of a hundred configurations from the same Monte Carlo simulation, so may be tens or hundreds or Gbytes. An entire data set from a research group may have on the order of a hundred ensembles (Monte Carlo runs with different input parameters), and thus be tens or hundreds of TBytes in size.

Lattice QCD simulations are notoriously computationally intensive, and physicists studying lattice QCD are heavy users of supercomputing facilities. Each configuration data file produced by Lattice QCD simulations is therefore very computationally expensive, typically requiring thousands of CPU-hours to generate. Because the data sets are so expensive to generate, the Lattice QCD community typically consists of large (usually national) collaborations that share computational resources and data. In some cases these collaborations are developing their own computational grids for this purpose, based on their national (or international) grid infrastructure and services. Currently there are large lattice QCD collaborations in the UK (UKQCD), USA (USQCD), Japan (JLDG), Australia (CSSM) and a European collaboration (LDG) with groups from Germany, France and Italy.

In December 2002 the International Lattice Data Grid (ILDG) project was initiated (ILDG 2006), to coordinate the worldwide sharing of Lattice QCD data between research groups in lattice QCD, in particular the large collaborations listed above. The ILDG formed two working groups to define the required standards to enable the development of a data grid, a Metadata Working Group to define standards for metadata and data formats, and a Middleware Working Group to define standard services for finding and accessing the data. The approach to the ILDG is very similar to international data grid projects that focus primarily on experimental data, in areas such as high-energy physics (LHC Compute Grid) (LCG 2007) and astronomy (International Virtual

Observatory), although currently the ILDG is concerned solely with distributed data access and not (yet) with distributed data processing.

This paper describes our experiences in setting up an Australian node of the ILDG, to serve several terabytes of configuration data from the Centre for the Subatomic Structure of Matter (CSSM), which is the national Lattice QCD collaboration in Australia. Although this work is specific to Lattice QCD data and the ILDG, most of the main issues in developing the data grid node are quite general and not specific to the type of data.

2 Data Grids

Grid computing aims to enable shared access to distributed resources, including computational resources, storage, and data sets. For many grid applications, the management, discovery, and transfer of large amounts of data is at least as important as access to shared computational resources. These are often referred to as data grid applications (Atkinson *et al*. 2003, Venugopal *et al.* 2006). In some cases, such as the ILDG, the focus is solely on enabling easy discovery and access of distributed data sets, and processing of the data is not considered in the functionality of the data grid. In these cases, the systems have much in common with efforts to develop digital libraries, with similar implementation issues. The integration of separate distributed data archives into a federated data grid requires:

- standard data formats for data files;
- standard metadata schema to describe the data;
- a registry for finding the servers that are part of the data grid;
- standard interfaces for querying the metadata (at a particular site or across multiple sites) to find the desired data;
- standard protocols and interfaces for accessing the data, and possibly also for processing the data before download.

Implementing a metadata query across all the sites in the data grid can be done using a distributed query to all sites and collating the results. However to address scalability, fault tolerance and performance issues, federated queries across large numbers of distributed data servers are more commonly done by indexing (or "harvesting") a subset of the metadata from all the sites into a metadata index which is used for the query. There may be more than one such index, to provide fault tolerance.

Copies of the same data file may exist on multiple sites in the data grid, and a common approach is to assign a globally unique logical file name (LFN) to each data file, and then use a file catalog, or replica catalog (Chervanek *et al* 2004), to map a logical file name into a set of URIs specifying the location of each of the data file replicas. The application (or the grid middleware) can then select the most appropriate replica (perhaps based on available network bandwidth) and instigate a file transfer to the desired location.

A number of software packages are available to support the development of data grid applications. Many of them are available in grid computing middleware systems such as the Globus Toolkit, the Virtual Data Toolkit (VDT) (VDT 2007), LCG and gLite (gLite 2007). The Storage Resource Broker (SRB) (SRB 2007) is another software package that is commonly used for building data grid applications.

One of the problems faced in this project was that implementation started in early 2004, during a period when grid middleware was undergoing a transition from fairly low-level systems with a procedural execution model (such as Globus Toolkit 2 and associated versions of the Virtual Data Toolkit) towards a high-level, object-oriented, service-oriented architecture based on web services (such as Globus Toolkit 4). At that time, most existing data grid software was based around GT2 and VDT, while some services had been implemented using the OGSI grid services (OGSI 2003) used by Globus Toolkit 3, which was being deprecated and soon to be replaced by Globus Toolkit 4, based on Web Services Resource Framework (WSRF). This made it difficult to build an architecture based on the data grid software available at the time. The ILDG Middleware Working Group decided to develop a service-oriented architecture based on web services, which we believe was a good decision given the uncertainties around grid technologies at that time. However, it meant that much of the software needed to be implemented from scratch.

3 International Lattice Data Grid

As described in the Introduction, Lattice QCD simulations require a lot of supercomputer time and produce large amounts of data. The International Lattice Data Grid (ILDG) project was initiated to coordinate the worldwide sharing of Lattice QCD data.

In order to create an interoperable distributed scientific data repository, the ILDG Metadata Working Group defined standards for configuration data file formats and metadata schema for specifying information about the data (Coddington, Joo *et al.* 2007), and the ILDG Middleware Working Group defined protocols and interfaces to services for finding and accessing the data (Coddington, Zhang *et al.* 2007).

The underlying design decisions for the data grid architecture and specifications were made in 2003-4 and some modifications and additions have been made since then. In order to enable interoperability between the different data generation and data processing programs, languages and platforms and the different data grid middleware and software used by the different participants in the ILDG, the Metadata Working Group based the data and metadata standards on XML and the Middleware Working Group used a Web Services framework for the architecture of the ILDG, with interfaces to the Metadata Catalog and File Catalog services defined using the Web Services Description Language (WSDL) (WSDL 2001). Members of the Australian team were on both of these working groups and provided input and feedback on the design of these

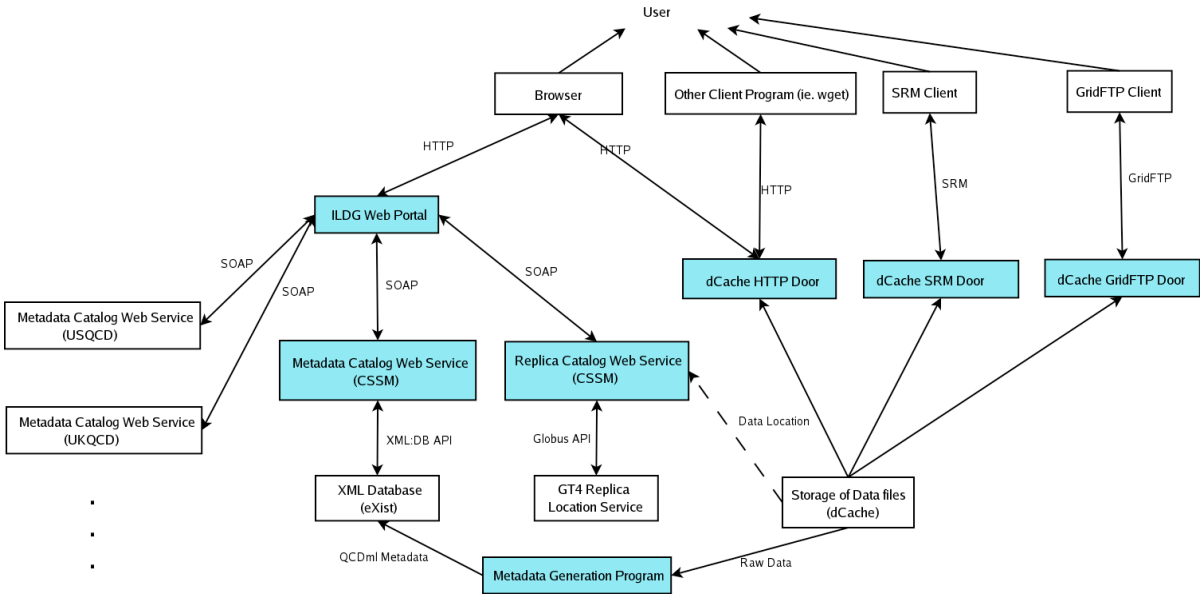

Figure 1. ILDG Architecture

specifications. A brief overview of these specifications is given below. More information about the data and metadata standards, and also about the services architecture, is available from the ILDG web site.

The overall architecture of the system is outlined in Figure 1. The components developed by us are represented by coloured boxes. The function and implementation of each of the components, and how the overall system operates, is explained in the following subsections.

3.1 Metadata standard (QCDml)

Since XML has become the standard approach for exchanging structured data over the Internet, the ILDG Metadata WG decided to adopt XML as the format of the metadata for describing the QCD simulation data files, and has defined the standard metadata fields using an XML Schema called QCDml (Maynard and Pleiter 2004, Coddington, Joo *et al.* 2007). QCDml allows information to be provided about each Lattice QCD configuration file and ensemble, including the program used to generate the data, the input parameters to the program, and the different types of Lattice QCD models and algorithms that the program implements. Version 1.0 of QCDml was released in June 2004. The current version is QCDml 1.4.1, released in June 2007.

Most of the metadata (and essentially all of the metadata that a physicist would be interested in) refers to parameters of the QCD model (e.g. the quark action and quark masses, the gluon action and interaction strengths, and the lattice size) and the Monte Carlo simulation. These are the same for each configuration data file in the same ensemble. So to avoid repetition and to enable faster queries, the QCDml metadata schema is split into two parts: the Ensemble schema, which describes metadata information common to all the configuration data files in one ensemble; and the Config schema, which describes metadata information that can differ between configuration data files (e.g. time the file was generated). Earlier versions of QCDml did not support some of the models used by CSSM so we proposed additions that were incorporated into later versions of the schema.

3.2 Data format standard (LIME)

The ILDG specified a standard data file format called Lattice QCD Interchange Message Encapsulation (LIME) (ILDG Metadata Working Group 2005) for Lattice QCD configuration files. LIME is an XML-based format that was loosely based on the mechanism used by Direct Internet Message Encapsulation (DIME) to incorporate binary data in an XML message. LIME has a header with some minimal standard metadata fields describing the data (e.g. the size of the lattice), but allows any other fields to be added if desired. The configuration data is stored using a standard binary representation of the configuration data, which specifies the ordering of the multi-dimensional data, the data type and endianness.

3.3 Metadata Catalog (MDC)

The Metadata Catalog provides the services for managing and querying the XML metadata that describes the Lattice QCD simulation data files. Xpath (XPath Guide 2007) is used as the query language. The MDC provides services for querying the Ensemble or Config metadata schemas, and returning the metadata (in XML form) for matching ensembles or data files.

Each ensemble has a globally unique identifier, which identifies all matches to the query and is used to associate

the Ensemble XML file with the set of corresponding Config XML files. It takes the form of a Uniform Resource Identifier (URI), which is referred to in QCDml as the markovChainURI. Similarly, each data file is assigned a URI, which acts as a globally unique logical file name (LFN) for the data file, and is referred to as dataLFN in the QCDml Config metadata schema. Each research group has their own internal mechanism for naming the ensembles and configuration files. The globally unique identifiers and logical file names are created by combining these internal names with a string identifying each research group. The markovChainURI in QCDml is specified to have the form

mc://<ResearchGroupName>/<InternalEnsembleName>

while the dataLFN has the form

lfn://<ResearchGroupName>/<InternalConfigFileName>

for example

lfn://cssm/su3b400k1318s12t24IMPFLICc001

is the logical file name for one of the CSSM research group's configuration files.

3.4 File Catalog (FC)

Querying the Metadata Catalog will provide the logical file names of the data files of interest to the user. The purpose of the File Catalog (or Replica Catalog) service is to convert these logical identifiers into physical file locations (URLs) to enable the files to be downloaded.

There may be multiple URLs for the same logical file name, since copies (or replicas) of the file could be kept on different servers in different locations in order to provide faster access to researchers at different locations, and also the same file could be accessed by different protocols, such as HTTP, FTP, GridFTP and SRM, and therefore be accessed via different URLs.

3.5 Data download service (SRM)

The Middleware Working Group recommended that each ILDG repository provide data access using a standard interface to data files known as the Storage Resource Manager (SRM) interface standard, which is designed to manage storage systems within a collaborative data grid and allows data to be downloaded using high-speed data transfer protocols such as gridFTP. SRM is supported by a number of data management software packages including dCache and DPM, some of which are included in standard grid middleware distributions such as VDT, LCG and gLite. SRM was chosen since it was an existing standard that was already used for high-energy physics experimental data, provides the standard interface to Storage Elements in gLite, and had proven interoperability between the different national or international grid sites used by the ILDG collaborations.

Research groups typically restrict access to their recent data sets to their own research collaborators until they have analysed them and published papers on their results, at which point they are made accessible to other researchers as part of the ILDG. In order to manage access to the data and provide information on who is accessing the data, the ILDG uses the Virtual Organization Management System (VOMS) (VOMS 2007), and has set up an ILDG Virtual Organization (VO) on a VOMS server. Under the ILDG VO, a group has been created for each of the major collaborations that are members of the ILDG (i.e. USQCD, UKQCD, JLDG, LDG, CSSM). VOMS allows researchers to request to join one of these groups in the ILDG VO, and their request is accepted or rejected by administrators who are appointed from each of the collaborations. If accepted, the researcher's details (including their X.509 certificate) are stored by VOMS. The SRM data download service can then use VOMS to authenticate user access to the data at each of the ILDG nodes, based on the user's certificate and their group membership specified in VOMS (see Section 8 for more details). Access permissions on the data servers at each ILDG node can be set up to allow access to configuration files in a particular ensemble from anyone in the ILDG VO, or only from researchers in specified collaborations.

3.6 Services Registry

The ILDG maintains a registry of the locations of each repository's implementation of these services. The registry is currently just an XML file available at a known URL. An alternative approach would be to set up a registry service where ILDG nodes could register their services, and applications could query to find the locations of all the ILDG services. However due to the small number of sites in the ILDG, and infrequent changes to the service information, a registry service was deemed to be unnecessary.

3.7 Use of ILDG

A typical use case for the ILDG is as follows. A researcher uses a portal to find some data of interest. This could be a web portal, or a standalone application (e.g. a Java GUI). First, the portal calls the Metadata Catalog web service to query the Ensemble metadata to find ensembles that match the parameters specified by the user. The MDC returns the XML for matching ensembles, which includes a globally unique identifier for each ensemble. If the user decides to download all the files in a particular ensemble, the portal again calls the MDC to query the Config metadata to find all the data files that are in the specified ensemble. The XML for the matching ensembles is returned, which includes the logical file name (LFN) for each data file. Next, the portal calls the File Catalog (or Replica Catalog) web service, providing the LFNs as input and getting a list of URLs as output, for example the SRM URLs of the data files. The portal then has all the information required to enable download of the data files from the SRM server to a location specified by the user. The user can then run their analysis programs on the data using the computing facilities or grid infrastructure they normally use.

Each of the Lattice QCD research collaborations contributing to the ILDG is expected to provide a repository of the Lattice QCD configuration data that they are willing to share, with:

- data files in LIME format;
- metadata in QCDml format for each ensemble and configuration file;
- a metadata catalog and a file catalog implementing the standard ILDG interfaces;
- a file transfer service that supports an SRM interface for download of data files by any member of the ILDG VO with a valid, internationally recognized grid certificate;
- URIs of the services stored in the services registry file.

Most groups also provide a web site that provides information about their data sets and a portal that uses these ILDG standards to allow users to search or browse the data in the group's ILDG data repository. In the following sections we discuss how the Australian node of ILDG met these requirements.

4 Metadata Generation

We developed a C program to convert between the custom CSSM configuration data file format and the standard ILDG LIME format. This was straightforward but a number of small problems meant that it took longer than expected. For example, CSSM's programs are written in Fortran, which uses extra characters to delimit array boundaries in binary files; different CSSM programs had different fields for metadata; and there were some technical problems in linking to the existing libraries for manipulating LIME files that we used in our implementation. It was a much more complex task to develop a program to generate the metadata describing all of CSSM's Lattice QCD configuration data as XML files conforming to the standard QCDml schema.

Unfortunately, many scientists have a minimalist approach to storing metadata, and this is often restricted to encoding metadata information into directory names or filenames, storing some minimal metadata (such as input parameters) in data files or program output files, or having information written down in (paper) notebooks. Often this is because there was no expectation that the data would be made available to anyone except the researcher or research group who generated it. However the development of high-speed wide-area networks and the Web have enabled easier sharing of large-scale data sets. This has led to many scientific disciplines developing standard metadata schemas and data formats to enable sharing of data, although in many fields (including Lattice QCD) this has been a recent development.

Once a standard metadata schema has been developed, it is relatively easy to set up a process to generate standard metadata for any new data files that are created. However it can be very difficult to generate standard metadata for existing data files where the available metadata may be very limited, particularly for older data sets where some of the data may not be available. This is also an issue for the Lattice QCD data that has been generated by the CSSM. Some of the required metadata is available from various sources, but the other metadata fields had to be obtained from the physicists who had developed the programs and run the simulations to generate the data. The main challenge of this aspect of the work was to capture this information in such a way that minimal effort was required by the physicists to specify the information, and the process of generating the metadata was automated as much as possible

4.1 Metadata Information Services

In order to automatically generate the metadata XML files, we need to find and collect the information required by the QCDml schema from various sources. Due to the complexity of the schema, we will omit unnecessary details and just describe the general approach.

The available information sources for automatically extracting metadata are the configuration data files, some of which may be accompanied by a text report file. The configuration data files are stored in binary format, and contain a small amount of metadata including the lattice size and the variable parameters to the action. The report file may contain some additional information such as the time that the program was run, the number of Monte Carlo steps between each configuration and some parameters to the Monte Carlo algorithm that usually have default values but may be varied. Some metadata is represented in the file name, e.g. what program was used to generate the data, the main parameters to that program, and the configuration number. The timestamp of the data file can be used to supply the time that the data file was generated (this may not always be accurate, e.g. if the files have been transferred from a remote site, but in some cases it is the only time information available). However some required information is not available from any of these sources:

- Management data: including project information and details of who generated the data files;
- Algorithm information: which Monte Carlo algorithm was used for the simulations, and with what parameters;
- Implementation information: what computer, which version of the program, and which parameters of the quark and gluon actions were used to generate the data.

Some of this information is available indirectly, for example there are only a few different configuration generation programs used by the CSSM, and each uses a single Monte Carlo algorithm and a small number of possible actions, and the choice of action is reflected in the data file name. Many of the required action and algorithm parameters are hard coded in the programs.

We therefore set up separate information files to capture some of this static information, based on the relevant elements of the QCDml schema. Some of the files are in plain text format with the information specified as attribute-value pairs, others are in the XML format of the QCDml elements and can just be dropped into the appropriate place when the QCDml files are generated. There are several kinds of information files:

1. Machine: contains the required information such as the name, institution and type of the computer used to generate the data file.

2. Program/Code: a text file containing information about each program, including its name, version, and data, and the names of the gluon action, quark action and Monte Carlo algorithm that are used in the program. The details of the quark and gluon actions and the algorithm are stored in separate XML files. For example, this file may specify that the program uses the DBW2 gluon action, the FLIC quark action, and the GHMC algorithm, and details of these are found in the files DBW2.xml, FLIC.xml and GHMC.xml.

3. Algorithm: an XML file containing the element in the QCDml Ensemble schema that specifies the details of the algorithm used to generate the data files in the ensemble.

4. Gluon Action: the gluonAction component of the Ensemble QCDml schema.

5. Quark Action: The quarkAction element of the Ensemble QCDml schema.

Configuration files are stored in a separate directory for each ensemble. Our strategy was to provide all the metadata that has to be manually generated in a small, simple plain text file, called "metadata.txt", in the directory for each ensemble. This file must be manually created, and contains just a few lines of information such as the project name, the name and institution of the researcher who ran the programs to generate the data, the name and version of the program used (which specifies the text file containing information about the program), and the machine used to run the program (which specifies the file with information about the machine). The same metadata can often be applied to multiple ensembles, so the metadata generation program can take the location of a global metadata.txt file as an optional input parameter.

4.2 Metadata Generation Program

The program to generate QCDml format metadata for CSSM data was developed in Java. It accepts the name of a directory containing an ensemble of configuration files as the runtime argument, a global metadata.txt file as input, and creates a Config XML metadata file for each LQCD configuration data in the directory, as well as a single Ensemble XML metadata file. There are two functional components in the program: information reader and metadata XML generator. The program employs the top-down parsing method to build a Document Object Model (DOM) tree or Document object, then outputs the DOM tree into an XML file.

Firstly, the program reads in all the information in the global metadata file, and (if it exists) the local metadata file in the given directory. The information in the local metadata file can override the global defaults. From this information, the QCDml Generator program can find the appropriate machine and program information files. From the program information file, we get the program metadata as well as information about the Monte Carlo algorithm and the quark and gluon actions. All the information read in by the QCDml generator program is stored in a hash table for later retrieval.

The QCDml generator program also reads in some information from the binary data file and (if it exists) the report text file. We investigated the output subroutines of each CSSM Lattice QCD program, and then composed specific handler programs to parse the binary files associated with each one, which used slightly different output formats. The report text files were also classified according to their formats and a parser written for each type. There is a handler mapping configuration file storing the matches among the program, the binary data file, and the report text file. Via the program information file and the handler mapping file, we locate the appropriate binary file handler and report file handler, and use them to parse these files to extract the required information.

The Java SAX parser is used to handle the predefined XML files containing metadata about the machine, algorithm, gluon and quark actions. The program constructs a DOM tree for each XML document and then uses a formatted XMLOutputter object to write out the DOM tree into an XML file.

5 Metadata Catalog

We spent some time looking at existing solutions for implementing a Metadata Catalog. At the time, there were several possibilities available. The European Data Grid project had developed a metadata catalog, however it was tightly integrated with the LCG grid middleware and could not easily be used as a stand-alone package. A Metadata Catalog Service (MCS) (Deelman *et al* 2004) has been developed that is built on top of OGSA-DAI (Antonioletti *et al* 2005), an implementation of the Open Grid Service Architecture (OGSA) Data Access and Integration standards. Unfortunately, the MCS and the original version of OGSA-DAI that was available in 2004 were developed using OGSI grid services (OGSI 2003) rather than standard web services. A web services version of OGSA-DAI has recently become available, however there is still no standard web services version of MCS. Also, MCS did not support arbitrary schemas and XPath queries. Storage Resource Broker provides a metadata catalog called MCAT, but it is not designed to be a general metadata catalog and only supports limited types of information. It would be very difficult to use it with the complex schemas used in QCDml. Also, it is tightly integrated with SRB, and ILDG chose SRM for data access, which SRB does not support.

We needed to provide a metadata catalog that conformed to the web services interface standards specified by the ILDG middleware working group, which accepts XPath format queries based on the QCDML schema. None of the existing software that we investigated offered what we wanted, so we developed our own metadata catalog implementation, using a web services interface to an XML database. This proved to be reasonably straightforward, and simpler than we anticipated. In this case, developing a custom solution was simpler and faster than modifying existing systems.

5.1 XML Database

Since we were generating the metadata in XML format, and aiming to support queries using XPath, it was easier to use an XML database as the backend to the Metadata Catalog. There is a limited choice of free, open source XML databases. The most commonly used are Xindice (Xindice 2007) and eXist (eXist 2007). We initially chose Xindice, since it is an Apache project and it is one of the databases supported by OGSA-DAI. It was also very simple to install, by deploying a web archive (war) file onto a Tomcat server.

The Xindice Core server is a seamless XML database server, which means all data going into and out of the server are XML. The Xindice server can be accessed through APIs or command line tools. We used the XML Database (XML:DB) API (XML:DB 2003), which is a standard API for XML databases, developed by the XML:DB Initiative, so that we had the option to change to another XML database in the future.

Later in the project we did in fact change the XML database to use eXist, since we found some problems with Xindice. eXist has several advantages that makes it a better option than Xindice for our MDC. Firstly, Xindice does not fully support the XPath 1.0 specification (Clark and DeRose, 1999); for example, Xindice can not handle an XPath with square brackets, such as, /person/name[1], where it will return an empty data set. However, eXist not only supports XPath completely, but also supports XQuery and XUpdate. Secondly, the performance of eXist is better than Xindice. According to tests done by Kolar et al. (Kolar 2006), eXist is up to 10 times faster than Xindice. In some cases, Xindice took a long time to process a query to the MDC. Lastly, system management of eXist is easy since it has a user-friendly web GUI and a Java GUI. Backup and restore can be achieved by clicking several buttons. On the other hand, Xindice only has a Java command line tool for data management and indexing, and cannot support backup and recovery. The only drawback of eXist is that it takes a longer time to store data than Xindice, since it generates extensive structural indexes while storing data. In this project, ILDG metadata stored in database will usually not be modified and addition of new data sets is infrequent, so that the XML database is running in read-only mode most of the time. Thus, the most important feature for our purposes is the performance of query not insertion. Hence, eXist meets the needs of this project. Migration from Xindice to eXist was not much work since they both conform to the XML:DB API specification, so there was no need to change the existing program structure. The only thing changed was the XPath query sentences, where many XPath 1.0 features were utilised to make these sentences clearer and more efficient.

5.2 MDC Implementation and Web Service

The MDC was developed as a stand-alone Java application. We made use of XML:DB XML Database API to construct the following services that we wanted our backend MDC to have:

- Managing the MDC, such as adding and removing metadata XML files within the MDC;
- Querying metadata;
- Updating metadata.

We used Apache Axis to provide the web service version conforming to the WSDL interface specifed by the ILDG Middleware Working Group. This was straightforward, although we found some interoperability problems between various MDC clients for some older versions of Axis.

6 File Catalog

The Globus Toolkit Replica Location Service (RLS) provides the basic functionality required by the ILDG File Catalog service. The most important function is to retrieve a list of physical URLs corresponding to a specified logical file name. Additional functions are defined to add and remove mappings between logical file names and physical URLs.

RLS is a standalone server that provides a simple registry storing physical location information of multiple replicas of a file. It provides a command line interface and APIs for different languages, such as Java and C, however it does not provide a web services interface, which is required for ILDG.

In our implementation of the ILDG File Catalog, we developed a simple web service wrapper RLS, which conformed to the WSDL definition from the ILDG Middleware Working Group. The wrapper was written in Java and based on Apache Axis 1.4. Thus, the implementation is quite straightforward. Axis is first used to generate skeleton classes according to the WSDL, then code is inserted into these classes to call the RSL Java API to do the work.

Our initial implementation of the File Catalog used the GT3 version of RLS, which was the version available at that time. Later we converted to using the GT4 version, which was essentially the same, so the migration was trivial. In the next version of the Globus Toolkit, version 4.1.2, a WSRF interface for RLS, called WS-RLS, will be provided.

7 Web Portal

A researcher would typically access the ILDG using a web interface or an application program that provided a graphical user interface. We have developed a web portal that enables users to search for and download data sets from any ILDG sites. It can also be configured to provide a customized interface to a particular ILDG site.

The researcher would first use the web portal to find data sets of interest by specifying some basic simulation parameters of interest, e.g. find all the ensembles generated using a particular Lattice QCD model, for a specified range of input parameters and lattice sizes.

The user can specify that the query is for a particular repository (e.g. the Australian ILDG repository containing CSSM data), in which case only one metadata

catalog would be queried. Alternatively, the search could be done for selected ILDG repositories, in which case the portal would do a distributed query across all the specified ILDG metadata catalogs, using the contact points for the web services obtained from the ILDG service registry, and the results would be combined and returned to the user. The search query is converted to an XPath query and sent to the specified Metadata Catalog(s).

The metadata query returns a list of globally unique identifiers (GUIDs) of ensembles that match the query, with a small amount of information about each ensemble. More detail about an ensemble can be found by simply clicking on the ensemble identifier. This generates a web page using an XSLT program to convert the XML metadata into HTML. The full XML document can also be displayed. The user could view this information and possibly change or refine their query. They could then select which of the matching ensembles or configuration files they wanted to download.

Based on this information, the user can select which ensembles they wish to download. The portal uses the Metadata Catalog to find the logical file names (LFNs) of all the configuration data files associated with the desired ensembles. It then queries the File Catalog to find the locations (URLs) of the physical files.

The portal will then generate a script that will use a relevant client program to download them. If the LFNs are a group of HTTP URLs, the script will use wget to download the file; if the LFNs are some SRM URLs, the script will use srmcp as the download program. Thus, users can easily run this script to download the files.

The portal makes use of a web service client that uses Dynamic Method Invocation (DMI) as a way to access the Metadata Catalog web service implementation. DMI has been chosen over using static stubs because the metadata catalog returns the result 30 times faster if DMI is used.

Results of the ensemble search have been divided into batches of ten, to provide easier viewing of results to the user and to avoid the server from slowing down if it gets a lot of hits. The web service client also makes use of web service sessions so when results are divided into batches of ten, succeeding queries (queries that ask for the next ten results) can return the result quicker rather than instantiating a brand new query. The speedup gained by enabling web service sessions is about a factor of ten.

8 Data Access using SRM

File transfer can be done using standard utilities such as HTTP, FTP or SFTP, or more sophisticated utilities developed for grid computing, such as GridFTP, or the Storage Resource Manager (SRM), which aims to provide more control, flexibility and robustness for data transfer.

The ILDG collaboration has specified Storage Resource Manager (SRM) as a standard data access interface, and ILDG sites are currently using dCache (dCache, 2007) as the platform to store and download ensemble data, since it is a commonly used storage system that supports SRM. SRM (Storage Resource Manager) (SRM, 2007) is a protocol that provides data allocation and file management for Grid users, and is used to interface to Storage Elements in gLite. Generally speaking, SRM is not a protocol to transfer files, such as FTP. Instead, it holds information about distributed, shared storage systems and advises users on the best location to retrieve data by weighing different figures, such as network topology, workload of each system, etc. In dCache, the whole process is transparent to end users. A user just enters an SRM URL to the dCache SRM client, then the client will get a GridFTP URL from the SRM server and thereafter start data transfer automatically from the nominated destination.

In this project, data is stored in a dCache server, which provides various interfaces for users to download, such as GridFTP, SRM, dccp (a dCache native protocol), and HTTP. Since dCache supports GSI authentication, several GSI-related modules are involved, e.g. a VOMS server (VOMS, 2007) that is used to manage the relation between users and groups (or virtual organizations), a GUMS server (GUMS, 2006) to manage the mapping between groups and local users. In order to collaborate with other groups that are members of ILDG, a central VOMS server in Germany is used to hold the information of the ILDG virtual organization and other groups that are part of ILDG, e.g. European collaborations and national collaborations in the US, UK, Japan and Australia. On the other hand, a local GUMS server configures the mapping of ILDG group and local accounts, which makes it easy to give file access permissions to the ILDG members. Then dCache uses its gPlazma module to get information from the local GUMS server to determine if the user can access the file they are requesting. The system structure is shown in Figure 2.

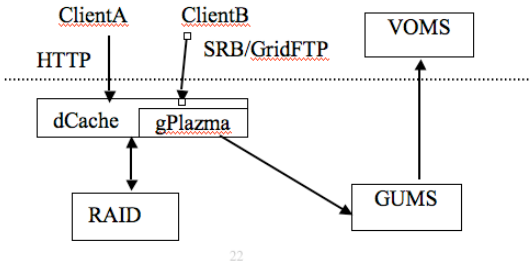

Figure 2. Data storage, access and authorization components of an ILDG site data server.

In our test system, dCache only has one pool to store data, which is in a RAID. dCache provides two protocols to users accessing from the Internet. Since the dCache is behind a university firewall, dedicated ports need to be opened to allow incoming requests. For example, port 80 is opened for http transfer, port 8443 is for SRM, while port 2811 is for GridFTP. As some of our data is not confidential and can be shared by the public, HTTP can be used to download this data because it does not need authentication, and so users do not require a recognized grid certificate. GSI security mechanism is enabled for

SRM and GridFTP requests, where a local GUMS server and a central VOMS server are involved. The central ILDG VOMS server stores certificate information of all ILDG users so that the owners of these certificates can access all ILDG data servers. The local GUMS synchronizes with the central VOMS server every hour and keeps a copy of certificate information. It also maintains a mapping of which local account each certificate is mapped to, so it can provide dCache with the information of a particular user and their associated local account.

dCache has a configurable set of permissions that each local account is granted on data files, such as read-only, or read-write. Users need to have a certificate that is authorised by an ILDG-recognised Certificate Authority (CA), which means that the CA root certificate is in place so that GUMS and gPlazma can use it to validate the certificate of an incoming request.

Before users download the data, a proxy must be created. When dCache gets the request, it will contact the local GUMS to ask if this user is mapped to a local account and what the account is. Then this local account is acting on behalf of the remote user to process data and returns the result. If the user is requesting via SRM, a pool that is close to the physical location of that user will be used to transfer the file. Since only one pool is set up for the current dCache system, SRM always returns a URL pointing to that pool, and then GridFTP is used to download the data file.

Getting file transfers working from all the ILDG sites was one of the biggest challenges in the ILDG project. There are many components that all need to be working properly for the SRM data download to work, including the network and firewall, the authentication system (gPlazma, GUMS and VOMS), certificate management and the mechanism for trusting different certificate authorities, dCache (which is non-trivial to install and configure), and the underlying data storage hardware. It took a lot of effort and a lot of testing by users at all sites to finally get file downloads from all ILDG sites working effectively.

During the time of installation, it was found that dCache versions 1.7.0-33 and later are stable and perform well. An old version was installed beforehand, which caused some problems, such as losing connection to pools and not getting authenticated by gPlazma. After reinstalling version 1.7.35, all these problems disappeared. Another big issue is the HTTP download module. This module is not documented in the official dCache manual, and still under testing so it is a challenge to use it. It did cause a problem to gPlazma. When this module is started, gPlazma always encounters a time-out error when retrieving information from GUMS. With the help of dCache engineers, the configuration was changed to start gPlazma in module mode instead of cell mode, and this fixed the problem.

In the next phase, a test with a tape silo will be conducted, with data stored on tape and using dCache as a ‘real’ cache for tape silo. Also, a data federation test between Canberra and Adelaide is in preparation

9 Conclusions

We have developed an Australian node of the International Lattice Data Grid, which supports standard interfaces to data grid services, and provides data and metadata in standard formats.

This work illustrated many of the challenges involved in making data available in a standard way through an international federated data repository.

Standards and interoperability are good things, but it can take a long time for many different research groups to agree on a set of standards for their discipline, since this is usually a complex, detailed task with a relatively low level of resource applied to it – scientists are mostly interested in doing science, not learning about XML schema and web services and debating the details of metadata standards and data types.

Standards are often revised to deal with new types of applications and data, new use cases, and new technologies. This means programs that use these standards need to be actively maintained and updated. For ILDG, the interfaces to the metadata catalog and file catalog have both been revised, and the QCDml metadata schema has been revised several times.

Even where standards are defined, developing robust, interoperable implementations of the standards is a challenging problem. Issues such as different sites using different grid middleware (e.g. different versions of gLite, VDT and Globus), different software, different versions of the same software, different certificate authorities, firewalls, revision of standards, have all caused problems in the design, implementation and use of the ILDG.

Lack of metadata or provenance information is a common problem with much scientific data. In many cases the data was generated or collected at a time when the researcher would not have anticipated that the data would be shared with researchers outside of their own group, so little effort was made to collect metadata. Even where metadata was collected, there may have been no agreed metadata standards at the time.

Existing public domain software to support scientific data repositories is very limited and typically restricted to supporting certain types of data and metadata schemas. For the most part, different scientific disciplines have designed and developed their own custom solutions. It is hoped that in future standard approaches and tools will become available for implementing federated scientific data repositories.

Acknowledgements

This work was funded in part by the Australian Partnership for Advanced Computing (APAC) Grid Applications Program and the South Australian Partnership for Advanced Computing (SAPAC) Summer Internship scheme. Meng Chen and Simon Pietsch provided additional programming effort and Daniel Cox and Grant Ward helped with systems support. Thanks to the members of the ILDG working groups for their invaluable input and assistance.

References

[All URLs accessed September 2007; dates listed are of last known update]

Antonioletti, M. *et al.*, The Design and Implementation of Grid Database Services in OGSA-DAI. *Concurrency and Computation: Practice and Experience,* Volume 17, Issue 2-4, Pages 357-376, February 2005.

Atkinson, M. *et al.*, Data Access, Integration and Management, in *The Grid 2: Blueprint for a New Computing Infrastructure*, Ian Foster and Cark Kesselman eds., Morgan Kaufmann, 2003.

Chervenak, Ann L., Palavalli, N., Bharathi, S., Kesselman, C., Schwartzkopf, R., Performance and Scalability of a Replica Location Service, in *Proc. of the International Symposium on High Performance Distributed Computing (HPDC-13),* Honolulu, June 2004.

Clark. J. and DeRose., S. XML Path Language (XPath) 1.0, November 1999. http://www.w3.org/TR/xpath.

Coddington, P., Joo, B., Maynard, C.M., Pleiter, D., Yoshie, T. (ILDG Metadata Working Group), Marking up lattice QCD configurations and ensembles, *Proc. XXV Int. Symposium on Lattice Field Theory, Regensburg, 2007, Proceedings of Science (LATTICE 2007) 048.*

Coddington, P., Zhang, S., Ishii, N., Sato, M., Melkumyan, D., Pleiter, D., Beckett, G., Ostrowski, R., Simone, J., Joo, B., Watson, C. (ILDG Middleware Working Group), Towards an interoperable International Lattice Datagrid, *Proc. XXV Int. Symposium on Lattice Field Theory, Regensburg, 2007, Proceedings of Science (LATTICE 2007) 044.*

dCache. http://www.dCache.org, 2007.

Deelman, E. *et al.*, Grid-Based Metadata Service, in *Proc. 16th International Conference on Scientific and Statistical Database Management (SSDBM04)*, Santorini Island, Greece, 21-23 June 2004.

eXist, http://exist.sourceforge.net/, 2007.

gLite, http://glite.web.cern.ch/glite/, 2007.

GUMS, http://grid.racf.bnl.gov/GUMS/, 2006.

RLS, Replica Location Service, http://www-unix.globus.org/grid_software/data/rls.php, 2007.

Globus Toolkit, http://www-unix.globus.org/toolkit/, 2007.

ILDG, http://www.lqcd.org/ildg/, 2006.

ILDG Metadata Working Group, ILDG Binary File Format (Rev. 1.1), 15 December 2005, http://www-zeuthen.desy.de/~pleiter/ildg/ildg-file-format-1.1.pdf.

Kolar P., Loupal P. Comparison of Native XML Databases and Experimenting with INEX, in *Proc. of the Dateso 2006 Annual International Workshop on Databases, Texts, Specifications and Objects*. Desna, Czech Republic, April 26-28, 2006.

LCG, LHC Computing Grid, http://lcg.web.cern.ch/LCG/, 2007.

Maynard, C.M., and Pleiter, D., QCDml: First milestones for building an International Lattice Data Grid, *Proc. of the XXIInd International Symposium on Lattice Field Theory (Lattice 2004)*, *Nuclear Physics B Proc. Suppl.* Volume 140, March 2005, Pages 213-221.

OGSI, Open Grid Services Infrastructure (OGSI) Version 1.0, 2003; http://www.ogf.org/documents/GFD.15.pdf

QCDml 1.4.1 Ensemble Schema, http://www.lqcd.org/ildg/QCDml/ensemble1.4/QCDml Ensemble1.4.1.xsd

SAPAC ILDG Portal, http://cssm.sasr.edu.au/ildg/

SRB, http://www.sdsc.edu/srb/index.php/, 2007.

SRM, Storage Resource Manager, http://sdm.lbl.gov/srm-wg/, 2007.

VDT, http://vdt.cs.wisc.edu/, 2007.

Venugopal, S., Buyya R., and Kotagiri, R., A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing, *ACM Computing Surveys*, Volume 38, No. 1, 1-53pp, ACM Press, New York, USA, March 2006.

VOMS, http://vdt.cs.wisc.edu/components/voms.html, 2007.

WSDL, http://www.w3.org/TR/wsdl, 2001.

WSRF, http://www.globus.org/wsrf/, 2007.

Xindice, http://xml.apache.org/xindice/, 2007.

XML:DB, XML:DB Initiative for XML Databases, http://xmldb-org.sourceforge.net/xapi/api/index.html, 2003.

Xpath Guide, http://xml.apache.org/xindice/guide-xpath.html, 2007.