

GEDDM: Comparisons of OGSA-DAI and GridFTP for access to and conversion of remote unstructured data in legal data mining.

Karen Loughran
Mark Prentice
Paul Donachy
Terry Harmer
Ron H Perrott
Belfast e-Science Centre
www.qub.ac.uk/escience

Sarah Bearder
Jens Rasch
Datactics Ltd
www.datactics.co.uk

Abstract

Managing unstructured data is a problem that has been around for as long as people have been using computers to electronically process information. As demands for data collection increases so does the number of formats and structures in which it is stored, presenting inherent problems for data mining applications.

Additionally, the sheer volume of data presents challenges for access and conversion in a timely manner. To further compound this problem the size of datasets will increase exponentially in future. There is therefore a need to access and convert large quantities of data from a variety of formats in a common, parallel and structured manner.

GEDDM is a collaborative industrial e-Science project in conjunction with BESC and industrial partners Datactics Ltd. A Common Semantic Model (CSM) is defined to assist with the representation and conversion of data from various sources. This model facilitates the conversion of data residing in a range of formats into a common format for subsequent data mining. The project exposes CSM conversion capabilities via a suite of Grid Services called Data Conversion Services (DCS).

This paper presents two implementations of the DCS. One under OGSA-DAI and another under GridFTP. Implementation and results are discussed, evaluated and conclusions presented.

1 Introduction

As information increasingly plays a crucial role in today's economy, the quality of such data is of paramount importance. Demand is constantly increasing for availability of quality data from an ever increasing range of sources. Data is collected about unlimited subject matters and stored in vastly different formats. The extents to which these sources are structured vary immensely. At one end of the scale we have structured data such as databases where columns and fields make individual units of data clearly identifiable. At the other end of the scale we have unstructured data typically in the form of plain text documents containing descriptive text, where little is known about the semantic content or how to identify an independent self-contained piece of information. The level of structure will dictate how and to what extent a source can be queried to extract useful information. At the

unstructured end of the scale queries may be simple string searches. At the structured end, where small indivisible units of data are identifiable, more complex queries will isolate or combine multiple entities to perform more intelligent data mining searches.

This paper presents the background and architecture of the CSM. Core to the design is the necessity to concisely and accurately describe the format and structure of Flat File Format (FFF) data sources. A Data Description Language (DDL) in the form of an XSD schema is presented which defines a set of rules to help a data analyst concisely describe a flat file data source. A parser will use this Data Description File (DDF) to convert the original FFF to a Common Output Format File (COFF). A high level overview of the design of an object oriented parser to perform this conversion is presented.

Two methods of implementing the DCS as Grid services are discussed and evaluated. Firstly, OGSA-DAI is middleware designed to assist with access and integration of data from separate data sources via the grid. Secondly, GridFTP is a high-performance, secure, reliable data transfer protocol optimised for high bandwidth wide-area networks.

Extensions are made to OGSA-DAI's activity framework to represent, convert and access data held in remote Flat File Formats (FFF). This is achieved by providing additional functionality to four key extensibility points within DAI's architecture. In this framework the DCS backend parser processing is wrapped within OGSA-DAI services.

GridFTP is already the popular choice for high-performance file transfer. Its functionality is extended to exploit additional command level Extended Retrieve/Extended Store (ERET/ESTO) processing abilities to tightly integrate conversions of unstructured data at the server prior to transmission. ERET/ESTO processing refers to extended functionality FTP retrieve/store commands which typically reduce the data in size prior to transmission. In this framework the DCS backend parser processing becomes an integral part of the GridFTP server.

Issues and experiences of developing both the OGSA-DAI and GridFTP frameworks including relevant benefits and drawbacks of each are presented. In particular a commercial use case from the GEDDM project is implemented to provide performance comparisons of the OGSA-DAI and GridFTP solutions. Benchmarking results are presented for access to and conversion of a large (1GByte) remote file using both frameworks. The data to be converted from the industrial use case includes text dumps in report format of legal court case transcripts. In addition, an assessment is made of installation requirements, pre-requisites, setup and usability from the user's perspective.

Finally, based on experiences gained and lessons learned through implementation of the DCS, conclusions are drawn and a roadmap is provided for enhancing the model by suggesting possible future optimisations and improvements.

2 Current Practice

The biggest challenge for representation and conversion of unstructured data is from data sources which do not fall within the category of traditional databases. For example, email, web log, PDF and Word report documents. These formats traditionally store little or no information about the structure or semantic content of data within them. They are essentially text-like Flat File Formats (FFF) with no means of identifying independent self-contained pieces of information. Additional to these formats, custom database formats often do not have ODBC type connectors, making direct access by external data mining applications impossible. In these circumstances database contents are often dumped to plain text files (FFF) using arbitrary print formats.

FFFs present many challenges for data conversion. These arise from the sheer irregularity of how data in FFF files can appear. In current practices, typically a new program must be designed and implemented for conversion of *each* new FFF data source to a structured format which can be subsequently mined. This is a costly and time consuming operation which must be performed for each business request to data mine a FFF. Of late, the Industrial partner has had requests from major customers in the US to interrogate data sources in numerous structures and formats. These business opportunities all present a similar technical problem, in that interfacing to such information in a common structured approach across such disparate structure and sources was a bottle neck and problematic. (E.g. one legal customer held over 45Tb of data in various formats including email, PDF, web logs, various RDBMS).

3 Architecture

In the overall architecture of the Common Semantic Model a client will first retrieve a remote sample of a FFF data source via a **getSample** service. Based on this sample the client will make a first attempt at describing the format of the FFF in a DDF file. The client will then undertake an iterative process of performing remote conversions on the FFF via a **conversion** service with each newly revised version of the DDF file. The process is repeated until the client is happy that the output from conversion accurately represents a

formatted version of the original FFF data source. Results of each sample conversion will be displayed locally on the client in response to a **getResult** service call. Figure 1 demonstrates an architecture whereby a client performs such services.

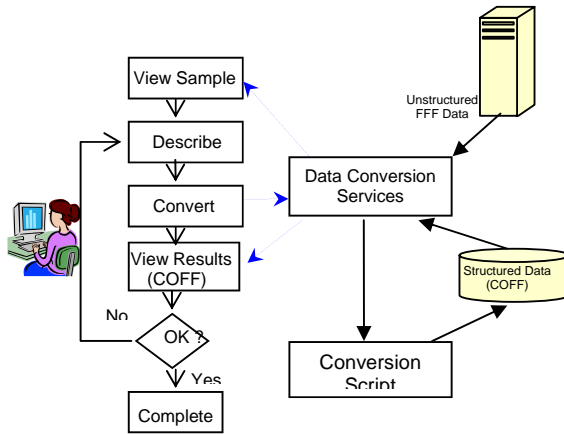


Figure 1 – CSM Architecture

4 Data Conversion Services

Core to the design of the Common Semantic Model is the necessity to concisely and accurately describe the format and structure of FFF data sources. The Data Conversion Service will rely heavily on this description to convert these data sources into a coherent common text format.

In GEDDM a description of a FFF data source is contained within a Data Description File (DDF). The DDF describes what is known about the format and structure of the data source. The DDF is expressed in XML to make it easy to build supporting tools and libraries. A Data Description Language (DDL) defines a set of rules by which a description of a FFF data source in a DDF must comply.

There are many similarities in principle between the design of the DDL within this Common Semantic Model and the design of Data Format Description Language (DFDL) [4] by an OGSA-DAI work group [1]. In evaluating suitability of DDL and DFDL for use within the CSM it is important to highlight the main difference between them. An FFF description in DDL will be considerably more restrictive and less extensible than one in DFDL. DDL focuses primarily on the description of the 4 broad categories of ascii text FFF formats in the industrial client's

business domain (email, web log, report and database dump). DDL is sufficient for our needs and relatively straightforward to implement. DFDL processing has not yet been implemented under OGSA-DAI and it awaits GGF approval before progress is made in this area. But one compelling argument for the simpler approach with DDL is that for our purposes we treat the files as character and string data as we are simply representing and converting a *text* format to a common *text* format for *subsequent* mining. We do not need to represent and manipulate pieces of data as dates, times, integers, floats etc. We are only concerned with the preparation of data resources before data mining by Datactics core data mining engines.

Figure 2 below shows an example of a FFF data source generated from a text file dump of a database. Figure 3 shows how it is described in DDL and figure 4 gives the resulting Common Output Format File (COFF) after conversion which clearly records format and structure within the data. As an example of just one type of irregularity which may exist within an FFF, the text of one column (the address field) is continued on the following line *after* the ending new line of the record. This is recorded in the description of the data source with a “multiline” attribute within the “pfield” element.

App	Account	Address	Balance
IMP	343818	Dede H Smith 181 Glen Rd Earls Court, London	8600.76
IMP	565777	Annie Saunders 60 Newhaven St Edinburgh, Scotland	9905.50

Figure 2 – FFF data source

As there is no clear delimitation between fields in this data source it must be described with positional information within the SCF using “pfield” elements:

```
<datasource>
<database>
<header><headertext>App Account Address
Balance</headertext></header>
<rectype eorecord='\n'>
<pfield name="App" pos=1 length=3/>
<pfield name="Account" pos=10 length=6/>
<pfield name="Address" pos=24 length=23
multiline="yes"/>
<pfield name="Balance" pos=49 length=8/>
</rectype>
```

```
</database>
</datasource>
```

Figure 3 – DDF description of FFF

```
<STARTSOURCE>
<TEXT> App Account Address
  Balance </TEXT>
<STARTRECORD>
IMP@343818@Dede H Smith, 181 Glen Rd, Earls
Court, London@8600.76
<ENDRECORD>
<STARTRECORD>
IMP@565777@Annie Saunders, 60 Newhaven St,
Edinburgh, Scotland@ 9905.50
<ENDRECORD>
<ENDSOURCE>
```

Figure 4 – COFF output from conversion

Each separate indivisible unit of data as described in the DDF is clearly delimited from other data in a common manner.

For the actual conversion of a data source (described by a DDF) an object oriented hierarchy of components is defined which together collectively represents a complete data source. From these, a direct mapping can be made between XML components within the DDF and Object Oriented (OO) class components which internally represent data units of the type described by the XML element. The implementation performs a SAX parse of the XML description held in the DDF, and creates and inserts objects of appropriate types into the correct location within the new overall OO hierarchical representation of a data source.

Each object in this hierarchy encapsulates information relating to the data unit it describes in its attributes. It will also encapsulate an all important “parse” method which will contain code for parsing a component of its type from a FFF based on its attributes. Once the SAX parse phase has generated an internal representation of the source in the form of objects, a call is made to the “parse” method of the highest parent object, i.e. the “data source” object, which will result in a “parse” operation on each sub-component in turn. As each object is parsed, the resulting output is produced to COFF with clear delimiters inserted.

5 Implementation under OGSA-DAI

The conversion service capabilities described in the previous section are exposed via a suite of Grid Services called Data Conversion Services (DCS). This is based on the Open Grid Services Architecture (OGSA) model [2].

The DCS is implemented under an OGSA-DAI [1] Grid based framework which provides an extension to the OGSA specifications [2,3] to allow data resources such as databases to be incorporated within an OGSA framework. This supports access to and integration of a wide range of data sources. OGSA-DAI is extended to represent, convert and access data held in FFF such as those identified in this paper. This is achieved by providing additional functionality to four key extensibility points within DAI’s architecture. The provision of:

1. A Connectivity Driver class which invokes conversion operations on FFF data sources.
2. Activity definitions and class mappings to allow the client to interact with a FFF data source.
3. Design of perform documents to express data conversion operations.
4. A configuration file to specify the data source exposed and point to details of the connectivity driver, activity & class mappings and perform document designs active for the FFF data source.

Familiarisation of OGSA-DAI architectural framework was required before defining a new conversion activity and incorporating it into the existing framework. The existence of clear documentation helped in the process of doing this. The process of actually transferring files between client and server in this implementation uses GridFTP directly as opposed to using the deliverToGridFTP activity within OGSA-DAI. At the time when this was being implemented under an earlier version of OGSA-DAI deliverToGridFTP would not work.

In test scenario A, a large 1Gigabyte unstructured FFF is remotely accessed and converted using the DCS OGSA-DAI services from a 1.5Ghz Sunfire box with 1 Gigabit network connection to a 2.2 GHz optron also with a 1 Gigabit network connection. Both

client and server exist geographically within the campus at Queens University Belfast on different sub-nets.

In test scenario B, a large 1Gigabyte unstructured FFF is remotely access and converted across organizational boundaries using DCS OGSA-DAI services from a 3.5Ghz Pentium 4 box located in London to a machine with the same specification in Belfast. Both of these machines have a 1Gigabit network connection.

6 Implementation under GridFTP

GridFTP is a high performance, secure, reliable data transfer protocol for high bandwidth wide-area networks. Version 4.0.0 has a new server architecture which, among other improvements, allows a user to manipulate data prior to transmission by exploitation of extra ERET/ESTO processing handled via a Data Storage Interface (DSI). A DSI presents a modular abstraction layer to a storage system. It consists of several function signatures and a set of semantics. When a new DSI is created a programmer implements the functions to provide the semantics associated with them. For our purposes we incorporate conversion of unstructured FFFs to structured FFFs within the DSI.

Another useful feature in the new server architecture is support for striping - using multiple hosts to move a single file [6]. With striping, multiple nodes work together and act as a single GridFTP server. An underlying parallel file system allows all nodes to see the same file system. Each node then moves (reads or writes) only the pieces of the file that it is responsible for, allowing for multiple levels of parallelism, CPU, bus, NIC, disk, etc. [7]. In striped mode, transfers are divided over all available back end data nodes, thus allowing the combined bandwidth of all data nodes to be used. The architecture of a typical striped GridFTP server is shown in Figure 5 below.

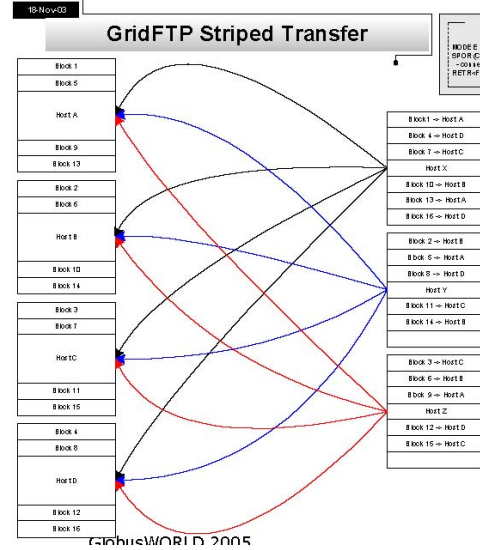


Figure 5 – GridFTP Striped Transfer [7]

Firstly, a standard DSI is written. This is based on GridFTP's default DSI for normal transfer of files between client and server (without extra processing). Once this DSI is tested and works as expected for normal GridFTP transfers, code extensions are made to the 'send' routine to perform extra processing on the data prior to transmission back to the client. This extra processing currently takes the form of a system command call to python scripts to perform the conversion of the unstructured input FFF to a structured output COFF file. The COFF file is then transferred back to the client (not the original unstructured FFF). The conversion operation is therefore incorporated seamlessly within GridFTP's server processing.

The process of writing and debugging the DSI was not altogether straight forward. This was partly due to the fact that work began on its creation prior to the availability of any GT4.0.0 release documentation on creating a DSI. Little documentation existed in any one location to help in writing a Data Storage Interface but a useful source of information was a grid-ftp mailing list [8].

In GridFTP test scenario A, transfer and conversion of a 1Gigabyte unstructured flat file is made from a 1.5Ghz Sunfire box with a 1Gigabit network connection to a 2.2 Ghz opteron also with 1Gigabit network connection. Both client and server exist geographically

with the campus of Queens University Belfast on different subnets.

In GridFTP test scenario B, a large 1Gigabyte unstructured FFF is remotely access and converted across organizational boundaries from a 3.5Ghz Pentium 4 box located in London to a machine with the same specification in Belfast. Both machines have a 1Gigabit network connection.

In test scenario C a remote striped GridFTP architecture is setup to transfer and convert a large 1Gigabyte unstructured FFF from London to Belfast. Three servers are used at both ends to facilitate the striped transfer. Each of these is a 3.5Ghz Pentium 4 box with 2 1Gbps network cards. One card provides access to a Blue Arc storage device (NFS mounted) and the other card provides access onto Janet. Blue Arcs are designed for multiple concurrent access so hosts at the sending side can simultaneously access different blocks of the file and move these simultaneously over the network to the target site. Likewise at the receiving end the Blue Arc storage device can simultaneously write the blocks received from the various remote striped servers to the correct location on disk. Figure 6 illustrates the architecture of this test scenario.

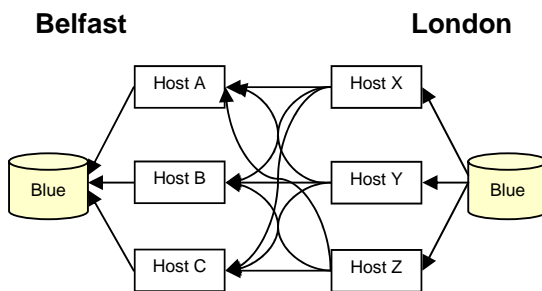


Figure 6 – GridFTP Striped Transfer [7]

7 Conclusions and Results

For DCS services under OGSA-DAI, the client experiences a significant time delay on each initial connection to the services. Subsequent connections did not suffer this delay, presumably due to caching.

OGSA-DAI's Distributed Query Processing (DQP) uses SOAP to transfer result sets from

server to client. This is fine for small result sets, but impractical for the transfer of whole flat files which typically may be anything up to and above 1Gbyte in size. In our case anyway we simply wish to use OGSA-DAI without DQP as we wish only to perform remote access and conversion of unstructured files. In this case the best options available to us through OGSA-DAI for transferring large files are the deliverToURL or deliverToGridFTP activities. But without the requirement to process individual queries on the data, the extra overheads of installation, startup time etc. involved with running DCS services under OGSA-DAI is difficult to justify.

GridFTP alone proves to offer high performance, secure and reliable large file data transfer. Its new striping feature allows for multiple levels of parallelism, CPU, bus, NIC, disk, etc. giving optimum efficiency for transferal of very large files. But most importantly for our purposes, the ability to create and load our own dynamic DSI allows us to tightly encompass *conversions* of remote unstructured flat files within GridFTP's server modules.

A full set of results timings and performance metrics will be made available at the AHM2005 conference.

8 Summary

Grid technology presents a framework that aims to provide access to heterogeneous resources in a secure, reliable and scalable manner across various administrative boundaries. Conversion of unstructured data sources is an ideal candidate to exploit the benefits of such a framework.

The design of the SCL language to describe a FFF in this application is suitably restrictive and simplistic for our purposes; we were only concerned with the conversion of text *prior* to subsequent data mining. Other data mining applications may benefit from the full power and complexity of DFDL which provides the ability to manipulate data from FFFs as dates, times, integers etc. It would therefore provide an application with the ability to perform proper data mining operations on FFF.

This paper described implementation of a DCS under both an OGSA-DAI architecture and under a GridFTP extended server architecture.

Implementation under OGSA-DAI was quite straightforward with clear documentation to help install services and extend activities. For DCS under GridFTP, lack of readily available documentation on writing a DSI to extend the GridFTP server presented some problems for its implementation, although a grid-ftp mailing list proved very helpful.

At run-time OGSA-DAI suffered from very slow initial connection times. This combined with the fact that we do not need to perform query processing on individual data records meant the extra installation and runtime overheads of OGSA-DAI were difficult to justify for our application.

GridFTP in contrast is light in terms of installation and proves to be a very fast, secure and reliable transfer tool which can neatly encompass the conversion operation within its server architecture component. The ability to add extra back end data nodes in a striped architecture allows potential for excellent performance improvement in the transfer of very large files.

In the future ongoing performance evaluation of results will be made and the feedback will be used to help identify enhancements to benefit future implementations of the DCS. Investigations will be made into creating a tighter integration of the conversion services within the GridFTP server module. Currently the conversion scripts which are integrated into the DSI are written in Python and called from the DSI 'C' module through a system command. A 'C' implementation of the services would avoid the necessity to use the system command and would result in a tighter, more efficient integration.

DCS services under GridFTP could also be extended to cope with distributed query processing. The server DSI could be extended to produce the COFF output in XML format. The client could then query information from this XML document using XPath. In this way the client could seamlessly query a distributed unstructured FFF as if it were held in a structured format in a local XML database.

References

[1] OGSA-DAI
<http://www.ogsadai.org>

- [2] OGSI
<http://www.gridforum.org/ogsi-wg/>
- [3] OGSA
<http://www.globus.org/ogsa/>
- [4] DFDL
<http://forge.gridforum.org/projects/dfd1-wg/>
- [5] Grid Enabled Distributed Data Mining and Conversion of Unstructured Data
<http://www.qub.ac.uk/escience/projects/geddm/>
- [6] Globus Toolkit 4.0 Release Manuals
<http://www.globus.org/toolkit/docs/4.0/data/gridftp/>
- [7] GT4-GridFTP for developers, the New GridFTP Server.
[http://www.nesc.ac.uk/talks/516/](http://www.nesc.ac.uk/talks/516/GridFTP4Developers.ppt)
GridFTP4Developers.ppt
- [8] GridFTP mailing list
http://www-unix.globus.org/mail_archive/gridftp-mpd/threads.html