



Storage and Infrastructure Work Packages 5 & 6

SI5: Develop an abstraction layer that supports a range of data replication systems, such as SDSC SRB, Globus RLS and GFarm

SI6: Allow simulation data to be retrieved from repositories or regenerated dynamically using computational services

Final Report

Version 1.0 June 2007

Chief Investigator: David Abramson

Prepared by:

David Abramson

David.abramson@infotech.monash.edu.au

Executive Summary

SI5 concerns the development of a higher level abstraction for replication services, called the Grid Replication Framework (GRF). The GRF allows the user to interact with a single API, but use several different replication services. It has a flexible modular design for supporting new replica clients. With this plug-in design, users can easily develop clients for GRF that add data access supports for other systems.

In SI6 we propose a life cycle, which starts when the data is first generated, and tracks its progress through replication, distribution, deletion and possible re-computation. We have designed and implemented an infrastructure, called Active Data, which combines existing Grid middleware to support the scientific data lifecycle in a platform-neutral environment.

Table of Contents

1	Introduction.....	4
2	Project Milestones	5
3	Project Outcomes.....	6
3.1	SI5	6
3.2	SI6	8
4	Archival Storage of Project Deliverables	10
5	Recommendations	11
6	Publications	12
7	Report Signoff.....	13

1 Introduction

This document describes the goals and achievements of the SI5 and SI6 work packages, and discusses how they contribute to DART. Since these two goals are closely related, it makes sense to report them in the one document.

The Grid provides infrastructure that allows an arbitrary application to be executed on a range of different computational resources, providing significant flexibility for users. Scheduling decisions are often made at run time, and are based on metrics such as the availability of the computational resources, network bandwidth and any necessary data files as well as trying to match application characteristics to the most suitable platforms.

When the input files are very large, or when fault tolerance is important, users may distribute complete replicas of the data to multiple Grid nodes. Such data replication is often facilitated by special middleware such as the Storage Resource Broker from SDSC and the Replication Services from Globus. Whilst powerful, existing middleware suffers from two shortcomings. First, it typically requires modification to existing applications, which is problematic when large and complex legacy codes are executed. Many applications are too fragile to be readily adapted. Second, if the user wishes to optimize application performance, they must choose the replica manually as there is no automatic selection. This is complex and error prone, and also does not allow for performance changes that might occur during program execution.

SI5 concerns the development of a higher level abstraction for replication services. It allows the user to interact with a single API, but use several different replication services.

Scientific applications often involve computation intensive workflows and may generate large amount of derived data. In SI6 we propose a life cycle, which starts when the data is first generated, and tracks its progress through replication, distribution, deletion and possible re-computation. We have designed and implemented an infrastructure, called Active Data, which combines existing Grid middleware to support the scientific data lifecycle in a platform-neutral environment.

2 Project Milestones

Project milestones were met. In SI5 we developed a working system that abstracts the SRB, GFarm and Globus RLS replication schemes. This has been demonstrated on a number of applications. In SI6 we have defined a data life-cycle, and have implemented a proof of concept demonstrator using Kepler.

SI5 milestones

1	System Design	Jun 2006	Met
2	Pilot Service	Dec 2006	Met

SI6 milestones

1	System Design	Jun 2006	Met
2	Prototype demonstrator	Dec 2006	Met

3 Project Outcomes

3.1 SI5

The Storage Resource Broker (SRB) is a data management system that employs a client-server architecture. It provides mechanisms to access data based on some attributes other than filenames. SRB uses a MCAT (Meta data Catalog) service to store meta data information for the stored datasets. Such information is used to identify and describe the associated data. Users of SRB can access the data using the command line clients called Scommands. Other means including a web portal called MySRB, and a Windows-based client called inQ. There are also APIs in C, Java and Python. SRB also provides two libraries, SrbIO and UnixIO, to facilitate code modification for legacy programs.

Initiated in Japan, Gfarm is an implementation of the Grid Datafarm architecture. This architecture is designed to handle hundreds of terabytes to petabytes of data using a global distributed file system. Different from SRB, Gfarm focuses on a Grid file system that provides scalable IO bandwidth and scalable parallel processing by integrating many local file systems and clusters. It uses a meta data management system to manage the file distribution, file system meta data and parallel process information. A file system daemon runs in each node to support remote file operations. The daemon also handles user authentication, file replication, node resource control and status monitoring. Files stored in the Gfarm file system can be accessed using a C library, or the Gfarm command line clients. Also, a syscall-hook library is provided for legacy programs to access Gfarm files.

The Globus Replica Location Service (RLS) is an implementation of the RLS framework. RLS is designed to replace the centralized Globus Replica Catalog in previous version of the Globus Toolkit. The RLS maintains and provides the mapping information from unique logical file names to physical file names; each of the physical file names usually contains the actual location of a file. It provides API and client commands to register, query and remove the file mappings but does not provide any data access mechanism. Other tools are needed in order to access the data. Files registered in the RLS can be copied using other tools such as GridFTP and, as a technical preview, the Data Replication Service (DRS). Despite this, there is no support provided by RLS to perform simple file IO.

For most applications data access usually involves conventional file operations, such as open, read, write, seek, stat and close. Each of the above three systems are built using different standards and protocols. Therefore, data access to one system is different from the others.

SRB and Gfarm both provide APIs for accessing replicated data and the APIs are in Unix IO style. Thus, performing file IO on an SRB object or a Gfarm file is similar to performing local file IO. However, this can create problems in research collaboration when shared data are stored in different systems, because an application developed to read SRB data cannot read files stored in Gfarm. Clearly, a more flexible data access middleware on top of the systems is needed.

Such middleware should provide transparent data access to the application. This means that the application is not required to know the location of the file or how to access it. An example of such middleware that has similar goals is GFAL, which is a library that presents Unix IO style interface for IO operations to provide file access using logical file names rather than fixed file names. However, one significant disadvantage in GFAL is that it does not support multiple replica systems, including SRB and Gfarm, which means that the application cannot access data stored in different replica systems.

GFAL has another limitation, namely, no optimisation is performed during file access. Optimised data access can significantly increase the execution performance. Optimisation methods varies, but the goal is to locate a better data source using information such as network condition, CPU load, distance to the data, etc.

Our GRF, shown in Figure 1 unifies the existing replica systems and hides the underlying complexity of the different data access mechanisms. Our current implementation supports , SRB, Globus RLS and Gfarm. Building on top of GRF, a Grid application is provided with a set of Unix style IO functions that support transparent data access. The file IO operations are separated from the application itself, so that there is no need to modify the application code when change of location occurs.

The GRF has a flexible modular design for supporting new replica clients. With this plug-in design, users can easily develop clients for GRF that add data access supports for other systems.

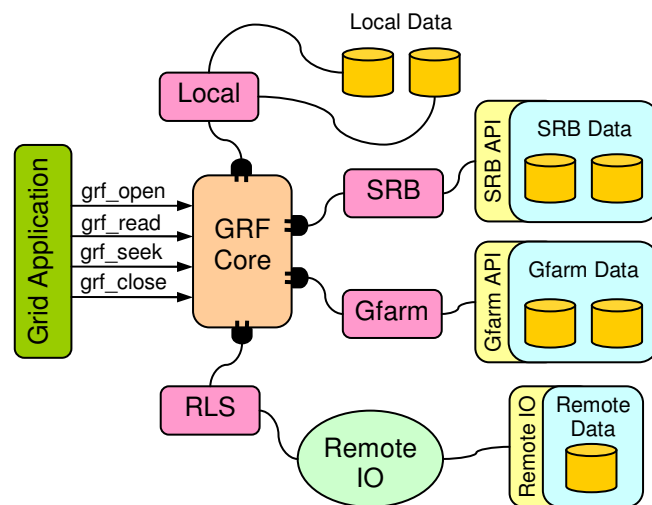


Figure 1 Architecture of GRF

3.2 SI6

Typically, scientific data is generated in two main ways; either from real laboratory based experiments or in-silico experiments. Real experiments in areas such as high energy physics, astronomy, biology and medicine, and geophysics, capture data from scientific instruments like telescopes, micro-arrays, seismic sensors and synchrotrons. The data is usually stored in primary repositories, where additional metadata can be attached and made available for interrogation. Primary data can be replicated and distributed to a variety of geographically distributed sites, where they are processed, mined, analyzed and refined to produce secondary data sets. Primary data also comes from in-silico experiments; computer simulations that model some real world phenomenon (possibly driven by real world data) and produce output data which is then stored and analyzed in much the same way as real data. Regardless of the data source, there is a well recognized need for data curation and provenance information management. For example, metadata such as the date and environmental conditions must be saved for a real experiment. Likewise, input parameters and date information must also be saved for an in-silico experiment. Because data from these two sources is handled in very similar ways, data management middleware typically does not distinguish between them. Thus, common solutions for handling metadata can be applied regardless of source. Tool kits such as the Storage Resource Broker (SRB) from the San Diego Supercomputing Centre, and the Globus RLS epitomize this approach.

The problem with an exponential growth in data is that it requires an exponential growth in storage capacity. Conventionally, communities usually store and replicate everything they capture (or compute). However, this is clearly wasteful, and it is unlikely that we will be able to continue doing this for very long. There are a number of potential solutions to the storage problem. For example, compression techniques might reduce the amount of storage required. However, in some instances it may be simpler to recapture or re-compute the data, rather than to store it. This leads to the development of a virtual data grid, where data are identified by logical names and derived data from previous experiments can be re-generated when needed. Research on virtual data has attracted focused attention from a small number of groups.

One view of the re-computation problem is that we want to produce a copy of a file transparently where one does not exist. Thus, if an application opens a file that was produced by a computational model, but the data has been deleted, then the file could be re-computed without the application being aware. Clearly, implementing such a strategy requires sufficient metadata on how a data set is produced, and a set of mechanisms that can reproduce it on demand.

Further, building on the idea of a virtual data grid, we view data regeneration as a special case of replication. When a computational model executes, it generates some output data that may later become input data of some other models. The data may be replicated to many different locations for various reasons. Later on when the data are not needed, they may be removed. In the end, this would result in complete data removal. When a reader application reads from this data, data regeneration is needed since all copies of the data are gone. We call this the Grid Data Life Cycle (GDLC), shown in Figure 2, from non-existence of data, to the original copy of new data (computed), to many copies of the data (replicated), to non-existence of data (deleted), and possibly back to one copy of the data (re-computation).

We have developed a prototype system that manages the GDLC of the computational models and workflows in a platform neutral environment. This system, called Active Data, provides mechanisms to existing applications and workflows to run on the Grid. It also allows them to access data stored in different replica management systems, to associate metadata that describes how the data is computed across multiple replica systems, to ensure that data cannot be removed unless sufficient metadata for regeneration is associated, and to regenerate the required data transparently when needed during execution. Importantly, because Active Data is built under our GriddLeS system, no source code modification is needed.

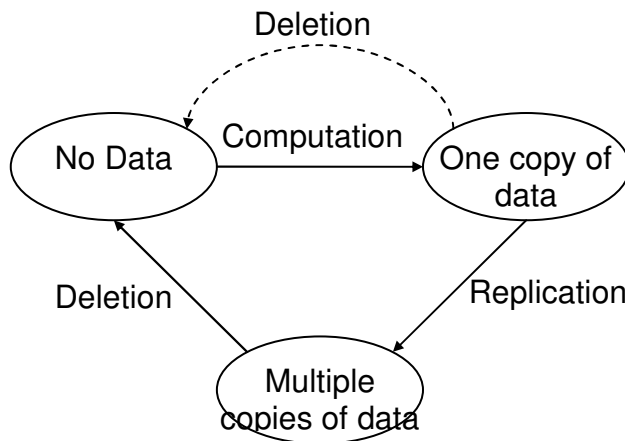


Figure 2 – Grid Data Life Cycle (GDLC)

4 Archival Storage of Project Deliverables

See attached CD/

5 Recommendations

From a research basis both projects were successful, and have demonstrated advances in middleware for scientific data management. Significant effort would be required if they are to be provided as robust commercial quality code.

The Grid Replication Framework (GRF) we have developed in SI5 unifies different replica systems by providing an interface that abstracts the different data access mechanisms. The same set of IO routines is used to access files available in different systems. This provides applications transparent data access to the supported systems and enables easy change of data source without the need to modify the application. Importantly, building such application requires little replica system knowledge because GRF hides the complexity of the underlying IO operations. The case studies demonstrated that GRF can greatly help applications running in multiple communities where files are distributed and stored in different replica systems.

Performance improvement is performed by a data selection service. Experiments in previous studies show that optimized data access can significantly increase execution performance. Results show that performing large amount of reads on small blocks of data is inefficient on the Grid because of the extra overheads. In particular, overheads within the IO layers of the different replica systems vary and can result in significant performance impact. Techniques such as pre-reading the data and buffering are needed. We plan to develop a module for GRF that provides a local data cache in order to overcome the overhead issue in the IO layers.

Active Data as developed in SI6 combines existing middleware, including GriddLeS, Kepler and several replica systems, to support the Grid Data Life cycle. It provides support for data access, replication, selection, deletion and regeneration. Importantly, no source code modification is required and almost all existing applications can be supported.

The GriddLeS library is used extensively in Active Data because it supports legacy software components. It also employs a flexible architecture that provides applications transparent access to data in the Grid. The support for data streaming can also significantly improve the program performance. Besides, Active Data extends the virtual data concept to support data removal, where data files within workflows should not be deleted unless the system knows how to recreate them when required. Also, data generation information is stored as metadata and is not specific to any workflow systems. We plan to support more systems in the future

6 Publications

1. Ho, T. and Abramson, D. “The GriddLeS Data Replication Service”, IEEE Conference on e-Science and Grid Computing, Melbourne, Dec 2005.
2. Abramson, D., Kommineni, J. and Altinas, I. “Flexible IO services in the Kepler Grid Workflow Tool”, IEEE Conference on e-Science and Grid Computing, Melbourne, Dec 2005.
3. Ho, T. and Abramson, D. “A Unified Data Grid Replication Framework”, 2nd IEEE International Conference on e-Science and Grid Computing. Dec. 4- 6, 2006, Amsterdam, Netherlands.
4. Ho, T. and Abramson, D., “Active Data: Supporting the Grid Data Life Cycle”, CCGrid 2007, Brazil.
5. Kommineni, J., Abramson, D. and Tan, J. “Communication over a Secured Heterogeneous Grid with the GriddLeS runtime environment”, 2nd IEEE International Conference on e-Science and Grid Computing. Dec. 4- 6, 2006, Amsterdam, Netherlands.

7 Report Signoff

It is agreed between

Professor David Abramson

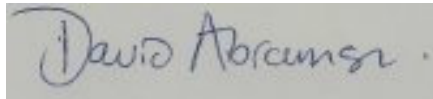
and

DART Project Director

That the **Final Report Document** for the DART S15 & S16 gives a full account of the work undertaken for the DART Project.

- has been read and reviewed by all parties,
- shows that the work package has been completed satisfactorily,
- clearly outlines the functionality that was delivered.

Dated this 6th day of June 2007



Signed by David Abramson for
and on behalf of the Chief
Investigator

Signed for and on behalf of DART by
the Project Director Andrew Treloar