A Data Management System for UNICORE 6

Tobias Schlauch[1], Anastasia Eifer[1],
Thomas Soddemann[2], and Andreas Schreiber[1]

[1] Simulation and Software Technology
German Aerospace Center
51147 Cologne, Germany
Tobias.Schlauch@dlr.de, Anastasia.Eifer@dlr.de, Andreas.Schreiber@dlr.de
http://www.dlr.de/sc
[2] Fraunhofer Institute for Algorithms and Scientific Computing (SCAI)
Sankt Augustin, Germany
Thomas.Soddemann@scai.fraunhofer.de
http://www.scai.fraunhofer.de

Abstract. Data produced in scientific and industrial applications is growing exponentially but most resource middleware systems lack of appropriate support for data and metadata management. In particular easy and intuitive retrieval of data for later use is a serious problem.

In this context the paper proposes a pragmatic approach for data management of distributed data with focus on appropriate means for data organization improving data retrieval.

The paper presents the key concepts and architecture of a dedicated data management system for sharing data located on heterogeneous storage resources. The different specifics of storage systems such as data object names, data locations, and data access methods are abstracted to allow transparent data access. Moreover, the system provides means for data structuring and organization by supporting custom data models and annotation of individual metadata on data objects.

Current development status of the system is illustrated by presenting an integration with the UNICORE Rich Client which has been validated in the context of the AeroGrid project.

Keywords: UNICORE, DataFinder, Distributed Data Management.

1 Introduction

The amount of data handled by applications is growing and growing. Especially scientific applications (e.g., in astrophysics) are dealing with hundreds of Terabytes of data today. This trend is similar in the industrial sector, although the absolute numbers are a little lower. Most of the data originates from experiments and simulations. It is obvious that astrophysical simulation and high energy physics experiments produce vast amounts of data. At least with the advent of robust design methods (e.g., for the automotive and aerospace industries) huge amounts of simulation data are also produced by manufacturing companies.

H.X. Lin et al. (Eds): Euro-Par 2009 Workshops, LNCS 6043, pp. 353–362, 2010.

With the increasing amount of data stored on disk and tape silos, it is more and more problematic to find and retrieve needed data sets efficiently. A solution is the annotation of the data with meaningful metadata.

The specific data structuring and metadata depends very much on the field of application. In general, the technical requirements for data management such as the ability for flexibly organizing the data, for annotation with various types of metadata, and for accessing a variety of storage resources are very similar in most applications. To support its institutes in this domain, the German Aerospace Center (DLR) has developed and deployed the data management application DataFinder [1]. DataFinder allows its users to manage data and metadata in an efficient manner.

In addition, resource management is as important as data management. Especially in times with growing energy costs, it is crucial that energy hungry compute resources are used efficiently. The AeroGrid project [2] a cooperation between industry, research centres, and universities is using the UNICORE 6 [3,4] middleware as their tool of choice for resource management. Unfortunately, today most resource management infrastructures lack a serious data and metadata management support. Thus a combination and cooperation of the two existing middleware stacks, UNICORE 6 and DataFinder, would be a good solution for the DLR and its partners. Hence, in this paper, we describe a first implementation of a dedicated data management system keeping compatibility with DataFinder and its integration in the UNICORE 6 middleware.

The paper is organized as follows. Section 2 provides an overview about data management support in selected distributed resource management middleware systems and dedicated systems suitable for distributed data management. Section 3 describes the data management system concept and the architecture in detail. The implementation including a description of the AeroGrid test bed is pointed out in Section 4. Finally, Section 5 gives a summary and describes future work.

2 Data Management–A Brief Overview

A key problem managing large amounts of data is fast and intuitive retrieval of produced data for later use. To accomplish this task, a data management system should support logical organization of data objects independently from storage resource specifics. Thus transparent data access is achieved without worrying about for instance migration tasks running in background, concrete data locations which might get broken over time, or provision of additional authentication information to gain access. Furthermore, the data management system should provide advanced means for data organization such as mapping of data object relations, metadata management, concepts for applying standard metadata sets, or access to data objects through specification of metadata search queries.

Like in most distributed resource management middleware systems, data management support in **UNICORE 6** is offered in form of simple data transfers to and between different UNICORE nodes. In addition, a storage service for

accessing hierarchically organized file systems is available [5]. Nevertheless, this service does not provide means for accessing data objects independently from concrete storage locations or advanced data structuring functionalities.

The **Globus Toolkit** provides a set of Web services for data movement and data replication [6]. In this context replicas can be identified using logical file names and data transfer is performed on basis of GridFTP. Specific means for data structuring or ordering are not provided.

Basically, **gLite** Grid middleware supports organization of files in logically arranged hierarchies [7]. Furthermore, means for replica and basic metadata management are provided. However, this functionality cannot be used separately from gLite.

The data management system **dCache** [8] manages disk pools distributed on different server systems. Data access is achieved independently from concrete data location through specific dCache commands. However, dCache provides no support for metadata management.

The **Storage Resource Broker** [9] is originally developed by San Diego Super Computing Center (SDSC) and focuses on realization of data federations beyond location and organization boundaries. In SRB no central control or administration exists. Thus every organization keeps control of data of its domain. SRB supports organization of files in a global, logical name space, data replication, and basic metadata management with a metadata catalog. Now, work is focused on the Integrated Rule-Oriented Data System (iRODS) which is the official successor of SRB.

iRODS [10] is a complete rewrite of the SRB following a rule-based approach. Customization of iRODS (e.g., adoption to a specific set of data management policies) is achieved by mapping these policies to iRODS rules without touching the core system. Beginning with release 2.0 iRODS has nearly caught up with the SRB functionality and now migration efforts from SRB to iRODS in existing SRB projects are on the way. However, metadata management functionalities have conceptually not been extended.

The **Chemomentum** [11] project focuses on collection, storage, and usage of shared data and metadata. Key features of data-related services are the provision of a global, logical data view, flexible metadata management functionalities, and support of data replication. The provided services are implemented as atomic Web services and can be used independently from a specific Grid middleware. However, no separated production release of these services is available so far.

DataFinder [1]is general purpose software for data management with focus on scientific and technical data. The data is annotated with meta-information and ordered into data structures. The customization of these structures is achieved through support of free-definable data models which also facilitate standard metadata annotations of managed data objects. Thus the system provides a standardized logical view on the managed data. In addition, DataFinder allows simple workflows to be automated with scripts and can be easily extended with additional functionality to achieve integration in an existing working environment. Furthermore, DataFinder acts as a single point of access to heterogeneous

data storage resources. DataFinder has been designed as a client-server system and provides rich user clients to allow usage of the basic data management functionalities encapsulated by a Python Application Programming Interface (API). Through the API the various storage resources and the metadata server are accessed. The communication with the metadata server takes place via the standardized Web-based Distributed Authoring and Versioning (WebDAV) protocol [12]. DataFinder is available as open source software under BSD license [13].

Most of the considered systems support a kind of storage virtualization (i.e., they provide means for arranging data objects independently from storage resource specifics). Additionally, some systems provide metadata management functionalities allowing simple annotations on data objects which are achieved using a metadata catalog component. Unfortunately, further means regarding data organization are missing. An exception from that rule are the data-related services developed in the Chemomentum project and the DataFinder system which are providing sophisticated means for data organization.

3 Data Management System Concept

A user expects an open standardized interface from a data management system. Unfortunately, standardization efforts in this direction have not yet gone far enough to cover our needs (Section 2). Hence, here a pragmatic approach on the basis of the DataFinder concepts has been chosen. The choice for DataFinder has been taken as it offers required means for data structuring and metadata management and its productive use in various scenarios provides a good basis demonstrating and investigating the developed system.

To provide the desired functionality, it is necessary to introduce abstractions for common data management concepts like data object names, storage resources, user and groups, and the methods for interacting with them. These abstractions hide the specifics of storage systems such as data access methods, data locations, as well as user authentication and authorization from the user. Realization of these abstractions can be achieved by defining the following logical name spaces.

1. **Logical data object names:** The system allows logical organization of data objects into hierarchies of collections and files. Additionally, the logical context of every data object can be expressed by specification of suitable metadata.
2. **Logical storage resource names:** The system identifies specific storage resource by a logical, unique name. This can for instance be used to realize transparent addition or removal of storage resources.
3. **Logical user names:** Every user is identified by a unique name within the system which enables the system to maintain access constraints for specific data objects.

By the logically organized data objects the system supports the definition of custom data models for specific parts in the logical data object name space. Thus

fine-grained restriction of the logical data structure is facilitated. Moreover, every data type specifies a default metadata set to enforce default metadata annotation of data objects.

The data management system possesses a client-server architecture which consists of two principal components, as shown in Figure 1.

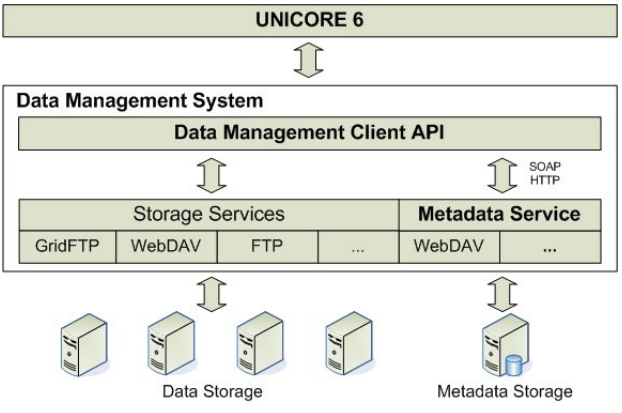

Fig. 1. Architecture of the Data Management System

Metadata Service is the core component which implements the logical name spaces. The name spaces are implemented by maintaining mappings and specific metadata in persistent metadata store. In detail the service provides file system-like functionality for data object organization within a global logical name space (e.g., creating, copying, moving, deleting data objects) and enforces restrictions of the specifically valid data model. Moreover, it provides information about data locations of data objects by maintaining global storage service configurations and links to the concrete storage locations. Furthermore, the service is intended to manage metadata and access privileges of data objects. Additionally, administrative functionality for maintenance of custom data models and storage service configurations are provided.

Data Management Client API uses the underlying storage services and the metadata service to provide data access through a uniform interface. By this API integration with UNICORE as well as other Grid middleware or software systems can be achieved. Basically, the client exports the functionalities provided by the metadata service (e.g., manipulation of logical structure or metadata access). Additionally, the client handles data access (i.e., it queries the metadata service for concrete data locations and initiates data transfers). In this context it is intended to bundle the data management client with existing standardized data transport interfaces to support interoperability and to reuse existing storage infrastructures.

4 Implementation

A first version of the data management client API bundled with data access over WebDAV and the metadata service component using a WebDAV server as storage backend have been implemented. The service fully implements all described functionalities except the access privilege management. Moreover, a plug-in for the UNICORE Rich Client has been developed to integrate the data management system with the UNICORE 6 Grid middleware. A first test installation of the system for further investigation has been established in the AeroGrid project. The software packages are available on the project site [14].

4.1 Metadata Service Implementation

The metadata service functionalities are provided through an atomic Web service. Technically, the service is implemented in Java using the Axis2 Web service framework. The Axis2 framework [15] has been chosen because it is widely adopted, supports relevant Web service standards, and allows deployment of Web services using standard web application containers.

The service has been designed in accordance to the contract first principle, i.e., beginning with the definition of the service interface using the Web Services Description Language (WSDL) version 1.1 [16] to provide a clean and simple interface. The metadata service owns a layered architecture which separates the core functionality (e.g., creation of a specific data object) from the concrete storage backend by using a dedicated persistence layer. The usage of the persistence layer allows the interchangeability of the concrete storage backend.

The metadata service uses the WebDAV protocol for the persistence of the data objects and different configuration resources. In the first place this is required to keep compatibility with DataFinder. However, WebDAV provides functionalities for the organization of files and collections in hierarchies and for the management of custom metadata that can be directly used implementing base concepts of the metadata service. Furthermore, the WebDAV extension *Access Control Protocol* [17] defines mechanisms which would allow an easy extension of the metadata service by access privilege management.

Data Object Name Space. The service manages data objects within a logical name space which is exemplarily illustrated in Figure 2.

The data object name space is organized hierarchically and starts with a virtual root data object. Every data object is associated with a logical name and is clearly addressed by its path (e.g., */Application 1/TRACE/Geometry*). In the hierarchy collections and files are distinguished. Collections can contain further collections and files. Files comply with leafs in the hierarchy and thus contain no further data objects.

Collections below the root data object correspond to a specific data management application (i.e., a specific configuration consisting of a data model and a set of storage service configurations are specifiable for these applications). This allows seamless management of data from different domains which implies different data relations.

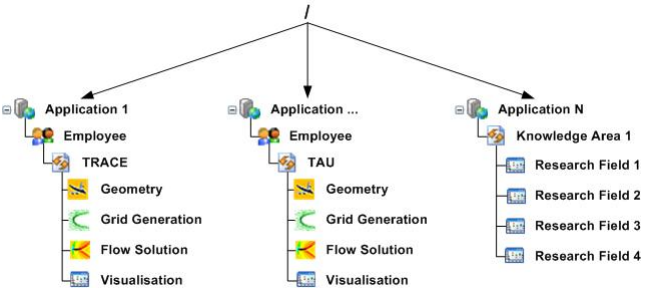

Fig. 2. Exemplary Data Object Name Space

Data Organization. In this context a data model consists of hierarchically arranged data type definitions beginning with a virtual root data type. This hierarchy can be seen as a template for the logical structure of data objects in which every data object is associated with a specific data type. Thus a suitable data structure reflecting data object relations can be defined to help identification of relevant information. Additionally, data types define sets of default metadata to support standard metadata annotations. In this context a further distinction of mandatory and non-mandatory metadata is made. The metadata is annotated on data objects with key value pairs. Metadata values can only be expressed using simple types such as *String*, *Number*, *Date*. The metadata service enforces that structural changes are made in accordance to the data model and that mandatory metadata is provided on creation time and on succeeding metadata updates.

For each data management application a custom data model reflecting the specifics of the managed data objects can be supplied. When creating a new application a default data model is provided reflecting the file system specific entities, namely directory and file. Afterwards the data model can be adapted to the specific requirements of the application by using the administrative interface of the service.

Data Location Service. To provide the data location service the system maintains storage service configurations. These configurations define the interface used to access data and store interface specific configuration parameters. Each data management application maintains its own set of configurations.

The following illustrates the treatment of data locations for importing a file: The data management client is bundled with a specific client-side implementation for accessing data through standardized data transfer interfaces. These interfaces and additional parameters are referenced in the managed storage service configurations. When creating a file data object the metadata service requires the specification of the storage service storing the data. By this information the service determines the concrete location and returns a Uniform Resource Locator (URI) [18] identifying the data location. For later data access the metadata service manages a data location link in the data object‘s metadata.

4.2 Data Management Client for UNICORE

The basis of the client is the data management client API which allows access to the above described functionalities of the metadata service and data transfer using WebDAV. The client is implemented as an Eclipse plug-in with extensions which permit a seamless communication with the UNICORE 6 Eclipse based client application.

Its graphical user interface (GUI) design is very similar to the existing DataFinder client. It allows users to browse and manipulate data object structure. Metadata can be manipulated under the constraints given by the active data model. Integration with the UNICORE 6 client is achieved by allowing users to select input and output files in the usual file browser fashion.

The implementation of the client is done by inheriting the model view controller pattern. The controller receives all events which originates from the GUI itself or UNICORE 6 directly. By using the model and GUI components, it triggers appropriate actions such as transferring data and metadata.

4.3 AeroGrid Test Bed

In the AeroGrid project data resulting from turbo machinery simulation is managed with DataFinder. In particular DataFinder has been extended to automate the aspects of simulation workflow execution. Figure 3 shows the AeroGrid deployment scenario.

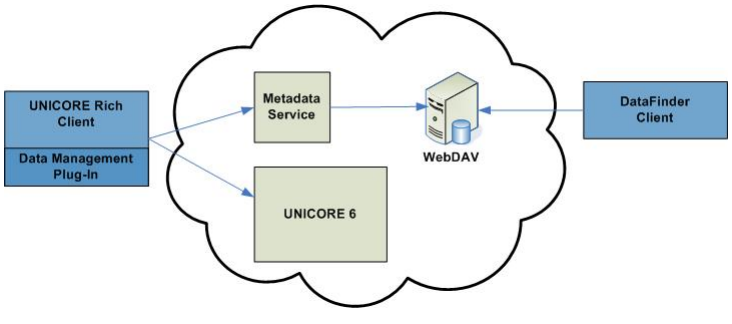

Fig. 3. Deployment Scenario of AeroGrid

The metadata service component has been deployed in the AeroGrid infrastructure independently from UNICORE 6. Both components, the metadata service and the DataFinder, are using the WebDAV server as metadata storage backend. In this context data is stored on this WebDAV server as well. The user is able to access via the UNICORE 6 Client and the developed plug-in logical data structures already managed with DataFinder through the metadata service. Data transfers can be performed identifying data locations with the help of the

metadata service and using the bundled WebDAV data transfer interface. Thus simulation runs can be initiated using UNICORE Rich Client and DataFinder.

The compatibility of the developed data management system with DataFinder is basically achieved on level of the metadata service. Both components are using the WebDAV protocol for storage of logical data structures and configuration resources. Moreover, the metadata service keeps the DataFinder scheme for annotating data objects with metadata as well the format of configuration resources. This provides a seamless integration of the metadata service with DataFinder. Moreover, the DataFinder administration client can be used to configure data models and storage service configurations which is not yet covered by the UNICORE Rich Client plug-in.

On this basis the functionalities of the metadata service and in particular the compatibility with DataFinder have been successfully validated. Because the Web service stack is additionally involved when accessing metadata, it is expected the implemented system causes specific overhead in comparison to DataFinder. Therefore, an advanced deployment scenario suitable for performance assessments and comparisons to DataFinder will be established.

5 Summary and Outlook

In the first part of the paper an overview about data management support in selected resource middleware systems and dedicated systems suitable for distributed data management has been given. The considered systems provide a kind of storage virtualization but no sophisticated means for data organization. Concerning growing amounts of data produced by current and future applications, data retrieval for later use becomes a serious problem.

Thus the paper proposes a pragmatic approach for a dedicated data management system based on DataFinder. The system delivers logical organization of data objects and abstracts common data management concepts hiding the specifics of storage systems. On this basis advanced means for data organization through metadata management functionalities and support of custom data models are provided. Moreover, the integration with UNICORE 6 is achieved by the data management client API which has been integrated in the UNICORE Rich Client. The developed system and its compatibility with DataFinder have been successfully validated in the AeroGrid project.

The next step concerns a detailed investigation of system performance which will be made in the AeroGrid project. Moreover, additional features are intended to be developed. This concerns the provision of fine-grained access privilege management by *Access Control Lists* and the implementation of a data object retrieval operation by metadata search queries.

Acknowledgments. This work has been partly supported by the German Federal Ministry for Research and Technology (BMBF) under Grant 01IG07006.

References

1. Schlauch, T., Schreiber, A.: Datafinder – a scientific data management solution. In: Ensuring the Long-Term Preservation and Value Adding to Scientific and Technical Data, PV 2007, Oberpfaffenhofen, Germany (2007)
2. German Aerospace Center (DLR): AeroGrid website, http://www.aero-grid.de
3. Erwin, D., Rambadt, M., Streit, A., Wieder, P.: Production-quality grid environments with unicore. In: Wallom, D., Kielmann, T. (eds.) Proceedings of the Workshop on Grid Applications: From Early Adopters to Mainstream Users, Global Grid Forum, pp. 10–17 (2006)
4. Jülich Supercomputing Centre: Unicore, http://www.unicore.eu/unicore
5. Jülich Supercomputing Centre: Unicore wiki - extending sms, http://unicore.wiki.sourceforge.net/ExtendingSMS
6. globus.org: Data management: Key concepts, http://www.globus.org/toolkit/docs/4.0/data/key/
7. Enabling Grids for E-Science (EGEE): EGEE gLite User's Guide - Overview of Glite Data Management (March 2005)
8. Fuhrmann, P., Gülzow, V.: dcache, storage system for the future. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1106–1113. Springer, Heidelberg (2006)
9. Baru, C., Moore, R., Rajasekar, A., Wan, M.: The sdsc storage resource broker. In: Proceedings of CASCON (1998)
10. Rajasekar, A., Wan, M., Moore, R., Schroeder, W.: A prototype rule-based distributed data management system. In: HPDC workshop on Next Generation Distributed Data Management (2006)
11. Schuller, B., Demuth, B., Mix, H., Rasch, K., Romberg, M., Sild, S., Maran, U., Bata, P., del Grosso, E., Casalegno, M., Piclin, N., Pintore, M., Sudholt, W., Baldridge, K.K.: Chemomentum - unicore 6 based infrastructure for complex applications in science and technology. In: Bougé, L., Forsell, M., Träff, J.L., Streit, A., Ziegler, W., Alexander, M., Childs, S. (eds.) Euro-Par Workshops 2007. LNCS, vol. 4854, pp. 82–93. Springer, Heidelberg (2008)
12. Goland, Y., Whitehead, E., Faizi, A., Carter, S.R., Jensen, D.: Http extensions for distributed authoring - webdav (February 1999), http://www.ietf.org/rfc/rfc2518.txt
13. German Aerospace Center (DLR): DataFinder, http://datafinder.sourceforge.net
14. Fraunhofer Institute for Algorithms and Scientific Computing (SCAI): Unicoredatafinder project, http://tor-2.scai.fraunhofer.de/gf/project/unicoredata/
15. The Apache Software Foundation: Apache axis2/java - next generation web services, http://ws.apache.org/axis2/
16. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web services description language (WSDL) 1.1, http://www.w3.org/TR/wsdl
17. Clemm, G., Reschke, J., Sedlar, E., Whitehead, J.: Web distributed authoring and versioning (webdav) - access control protocol (May 2004), http://www.ietf.org/rfc/rfc3744.txt
18. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform resource identifier (uri): Generic syntax (January 2005), http://www.ietf.org/rfc/rfc3986.txt