DataFinder – A Scientific Data Management Solution

Tobias Schlauch [1], Andreas Schreiber [2]

[1] German Aerospace Center (DLR)
Simulation and Software Technology
Lilienthalplatz 7, D-38108 Braunschweig, Germany
EMail: Tobias.Schlauch@dlr.de

[2] German Aerospace Center (DLR)
Simulation and Software Technology
Linder Höhe, D-51147 Cologne, Germany
EMail: Andreas.Schreiber@dlr.de

ABSTRACT

The data management system DataFinder primarily targets the management of scientific technical data. Data can be attached with individual meta data that is based on a free-definable hierarchical data model to achieve data structuring and ordering. Moreover, the system is able to handle large amounts of data and can be easily integrated in existing working environments.

The purpose of this paper is to present the key concepts of the DataFinder as well as current and future usage scenarios. In addition, the paper shows the extension of the existing architecture to integrate the Grid transport protocol GridFTP and the archiving system Tivoli Storage Manager (TSM) which leads in a general Grid-enabled data management application that is capable of handling distributed storage resources and archives.

Keywords: DataFinder, Data Management, Grid Computing, GridFTP, Tivoli Storage Manager, TRACE

INTRODUCTION

The development of the data management system DataFinder is motivated by different DLR-institutes that have early recognized the need of reorganizing their data management process. On the one hand the storage, backup and archiving of data is an important expense factor for these institutes but on the other hand the data represents the institute's knowledge. Therefore an evaluation process of commercial product data management systems was initiated. The outcome of the evaluation showed that the considered commercial systems would require high investments whereas delivering a lot of functionality which is not required by scientific research institutes. At the same time the DLR-Institution Simulation and Software Technology [1] developed a prototype of the DataFinder based on a flexible scientific data management concept, which has been designed for the simulation environment TENT [2]. The DataFinder prototype was easily able to solve the evaluation scenario so it was decided to enhance the prototype with the requirements of the involved institutes. The details of the evaluation process are described in [3].

The DataFinder is currently used and introduced at DLR for a couple of different applications and user communities. New requirements arise from these applications in particular the usage of archive storage systems. Moreover, the amount of data produced by current and future scientific applications is increasing exponentially. Hence the DataFinder is enhanced with interfaces for usage of Grid storage resources in the D-Grid Initiative [4].

The following sections give a general overview about basic concepts and functionality of the DataFinder system. Furthermore, a usage scenario is presented. Afterwards the integration of the archive storage system Tivoli Storage Manager (TSM) and the Grid data transfer protocol GridFTP are described.

ARCHITECTURE

The DataFinder system is based upon a client-server-architecture. Important design goals were the usage of open and stable standards and the limitation of the own development effort to a necessary minimum. Therefore a basic concept is the usage of the standardized Web-based Distributed Authoring and Versioning (WebDAV) protocol. WebDAV is based upon the Hypertext Transfer Protocol (HTTP) and defines server-side versioning, searching, and access control among other HTTP extensions. Moreover, on server side only existing third-party systems are used. Figure 1 shows the overall architecture.

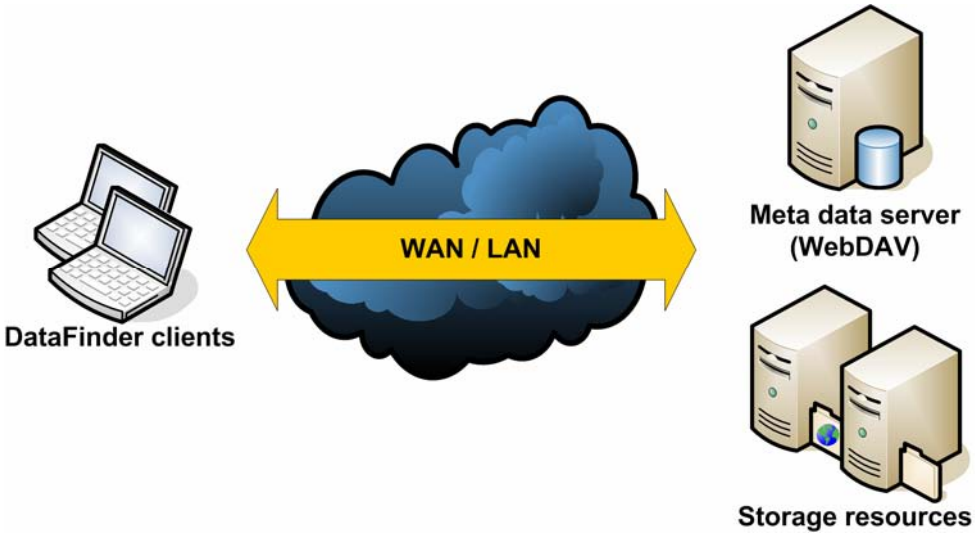

Figure 1: Client-server-architecture of the DataFinder system

The meta data server is accessed via the WebDAV protocol by the client and has to support the WebDAV extensions for server-side searching, versioning, and access control. For that reason the DataFinder system currently supports the commercial WebDAV server Tamino - developed by the Software AG [5] - and the open source WebDAV server Catacomb [6]. The meta data server basically provides access to the system and is used for the storage of central configuration information as well as the meta information and logical structure of the managed data files.

The managed data files can be stored together with the according meta data on the WebDAV server but for performance reasons the usage of heterogeneous, distributed storage resources is supported as well. In this case the meta data server only holds references to the data files. Storage resources are accessed via standardized data transfer protocols whereat the system currently supports FTP, CIFS, NFS and WebDAV.

FUNCTIONALITY

The DataFinder system is described best as a data management framework that offers basic data management functionality but is in particular flexible and easy extensible.

Basically the data files are managed in a hierarchical structure according to a free-definable data model. The structure consists of collections and resources similar to a file system. A collection contains collections as well as resources and corresponds to a directory. A resource contains no sub-items and corresponds to file. Additionally every data item is assigned a set of standard and data model specific properties. This logical data structure is stored on the meta data server.

The DataFinder provides the following standard functionalities:

- Import of data files into different storage locations.
- Creation of collections.
- Downloading, copying, moving, removing and renaming of collections and resources.
- Annotating of meta information to collections and resources.

- Specification of complex search queries based upon meta data.
- Access control based on access control lists (ACLs).
- Extensible graphical user interfaces.

SECURITY

Authentication and access control are dependent on the authentication repository of the meta data server. The DataFinder does not require an additional user management, which would increase the effort for maintenance. The authentication is directly performed against the meta data server. In this context standard HTTP respectively HTTP Secure (HTTPS) mechanisms are used.

The access to data items and configuration resources is controlled by ACLs. An ACL is attached to a specific data item and contains the privileges of specific users and groups. The DataFinder allows the querying of the authentication repository by using the WebDAV protocol or the Lightweight Directory Access Protocol. In current scenarios the DataFinder is used in conjunction with Microsoft Active Directory. Currently the privileges read data object, write data object, read data object ACL and write data object ACL are supported. An ACL defined on a collection is inherited by default. The ACLs are only stored and controlled on meta data level.

Secure communication with the meta data server is provided by using HTTPS. The communication security with the specific storage resource is dependent on the used data transfer protocol.

USER INTERFACES

The client side consists of a user client giving access to the basic data management functionalities and an administrative client allowing the setup of the whole environment. The clients are implemented with Python [7] and the graphical user interface toolkit Qt [8] to support a wide range of platforms. Moreover, the DataFinder provides a Python Application Programming Interface (API) to allow simple integration with existing applications. Figure 2 shows the DataFinder user client.

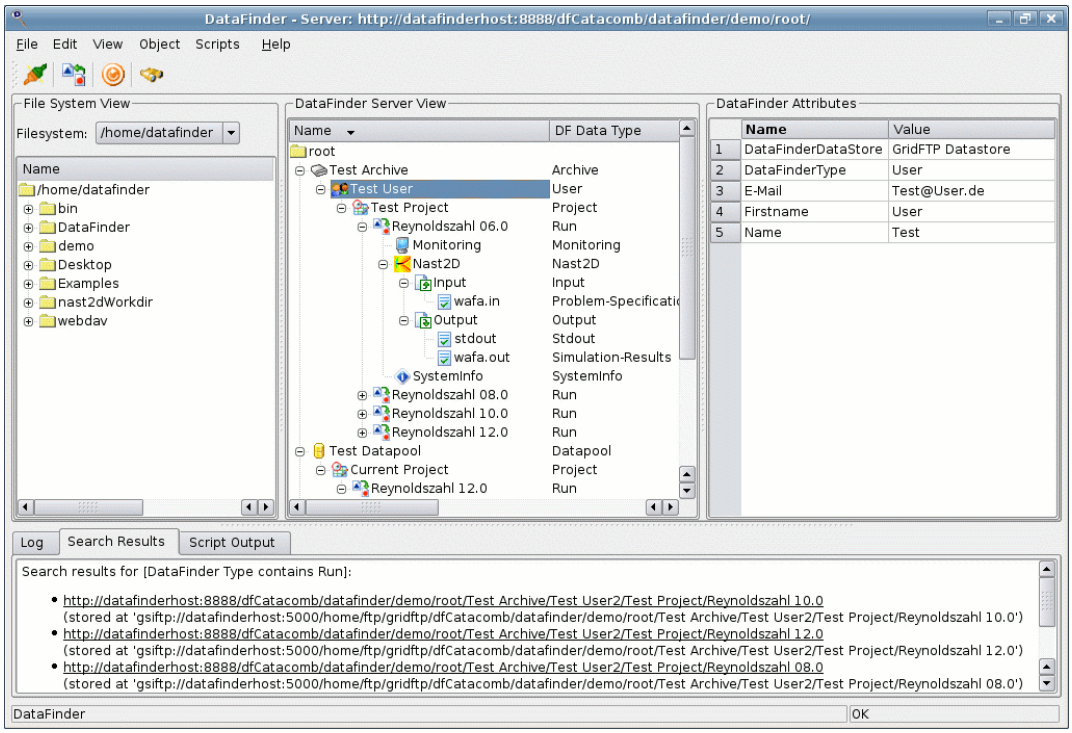

Figure 2: DataFinder user client

DATA STORES

In the DataFinder context, a specific storage resource accessed via a data transfer protocol is referred to as data store. A concrete data store is configured with the administration client whereat the configuration is stored on the meta data server. The configuration is centrally available and contains all required information for accessing the specific storage location including authentication credentials. The access to the configuration resources can be controlled using ACLs.

A data file imported with the DataFinder is assigned a meta information indicating the data store it is stored on. Therefore the storage location is fully transparent to the user. Moreover, this approach allows an easy migration of an existing data store to another storage location.

DATA MODELING

The data model consists of data types and parent-child relations between them. Every data type consists of a set of mandatory and required properties. The relations between the data types are always one-to-many whereat the definition of recursive relations is allowed. Both resources and collections own a data type. The difference is that resources correspond to a leaf node and collections to an inner node in the hierarchical structure. The modeling task is supported by the administration client.

EXTENSION VIA PYTHON SCRIPTS

The functionality of the DataFinder user client can be easily extended using the available Python API and Qt. In this context local and server scripts are distinguished. Local scripts are imported from the local file system and are only available on the specific client machine. Server scripts are stored on the meta data server and are available to all users. Every script owns a set of properties that, for instance, provide a description of the extension. Moreover, the script execution can be bound to specific data types. Examples of this script extension mechanism are presented in the following section.

USAGE SCENARIO

In the DLR scope the project-oriented management of relevant data of numeric simulation systems is a typical usage scenario of the DataFinder. In this section the Turbo machinery Research Aerodynamic Computational Environment (TRACE) developed by the DLR-Institute of Propulsion Technology [9] is used as an example. The purpose of this simulation system is the computation and investigation of complex turbo machinery flows. This example in particular demonstrates the necessary customization steps to optimally use the DataFinder in a specific environment.

The first step covers the development of an appropriate data model. This step is important because the data model indicates the relations between the managed data that are often only known to a few employees. In particular the data model makes theses relations visible to all users. Figure 3 shows the simplified TRACE data model without attributes.

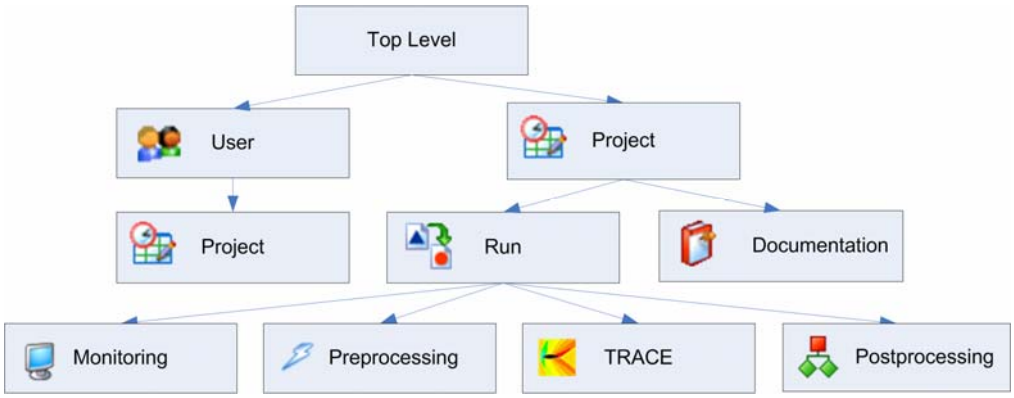

Figure 3: Simplified TRACE data model without attributes

The data model can be seen as a template of the hierarchical structure that can be created with the user client. On top level it is only possible to create collections with data type *Project* and *User*. Moreover, a recursion was modeled allowing the mapping of specific projects to a user. These types of collections are used to organize different simulation runs. A project itself consists of documentation resources and collections of data type *Run*. The *Run* data type entirely describes a TRACE simulation including input and output files, pre- and postprocessing data and specific monitoring information (e.g. simulation duration). Due to the standardized structure and meta data a user can easily identify relevant simulations.

The creation of the structure and import of files are standard functionalities of the DataFinder user client. The manually carrying out of these actions can become quite time-consuming. Therefore the development of script extensions automating common tasks marks the next customization step. The following additional functionalities exist in this usage scenario:

- Automated creation of *Run* collections whereat the user only needs to specify the required input files. The script extension imports these input files, creates the required sub-collections and adds required meta data to the created data items.
- Asynchronous workflow management. By using these script extensions the user is able to start a specific simulation on the local or on a remote machine, check the status of the simulation and cancel a running simulation. In this context input data respectively output data is automatically transferred between the DataFinder and the target machine performing the calculation.
- Project-specific simulation overview.
- Integration of common tools for input and output data visualization.

To sum up the described usage scenario demonstrates the close integration of a common workflow with the data management task that can be achieved with the DataFinder system. In this case even non-TRACE experts are easily able to run and evaluate TRACE simulations whereat the results are centrally available. Moreover, the standardized structure in conjunction with the search functionality allows fast and easy access to specific simulations.

INTEGRATION OF TIVOLI STORAGE MANAGER

Due to legal regulations it is necessary to preserve scientific results a prescribed retention period. In general the results do not require fast access wherefore they can be stored on existing archive systems. Moreover, the outsourcing of data to offline storage leads to a reduction of costs. The DLR IT service provider offers the usage of the archive system Tivoli Storage Manager [10] for backup and archiving tasks. In this context arose the requirement to directly archive DataFinder resources using the TSM from the DLR-Institute of Aerodynamics and Flow Technology [11].

In this first integration approach the DataFinder supports the archiving of:

- single resources managed by the DataFinder and
- single data files directly from the local file system.

In both cases the meta data and the archived data files are still accessible via the standard DataFinder functionalities. Moreover, additional meta data is stored for archived resources to be able to identify them in the archive system and to indicate the expiration of their retention period. At the moment data files are stored in the TSM archive without specific meta information.

The encapsulation of the TSM archive access is achieved by using the data store concept. Furthermore, the Python API and the user client have been adapted to support the two archiving use cases. On data store level it is possible to specify different archive options including indication of read-only archives and retention period of archived data. Due to the fact that the configuration is centrally provided by an administrator the archiving task becomes as simple as the import of data files.

In the current scenario the TSM archive used is only accessible by a specific TSM gateway machine. The gateway contains the TSM client required for TSM archive access. Hence it is necessary to transfer the data files to the local file system of the gateway machine and to initiate the different TSM actions on

this machine. So it was decided to use the standard protocols SFTP and SSH supported by the gateway machine for transferring the data to the gateway respectively controlling the archiving process via the TSM command line interface. This approach allows all users the usage of the archiving functionalities from their specific client machine. Figure 4 shows the different interactions between the components when archiving a single resource managed by the DataFinder system.

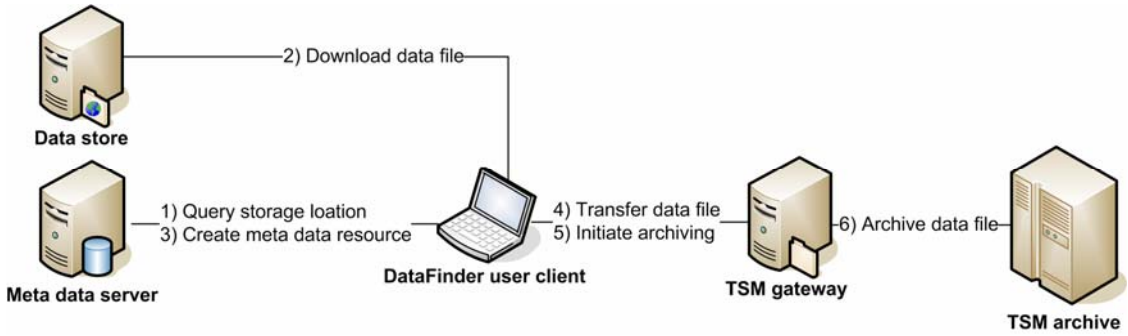

Figure 4: Archiving of a DataFinder data item

At first the storage location of the specific resource is retrieved from the meta data server and the data file is downloaded into the local file system of the user's client machine. In the next step a new meta data item with specific information is created on the meta data server. Finally, the data file is transferred to the TSM gateway and stored in the TSM archive.

The described integration is currently improved in different areas. For instance the support for archiving complete collections including all sub-collections and resources is going to be added. Moreover, it is planned to archive the meta information of the managed resources together with the data files according to the ISO reference model of an Open Archival Information System (OAIS) [12]. Additionally further approaches reducing the number of required data transfers are currently evaluated.

INTEGRATION OF GRIDFTP

For preparation of the DataFinder system to handle large amounts of data produced by future scientific application the system is enhanced with the ability to access Grid storage resources within the D-Grid Initiative respectively the D-Grid Integration project (DGI). Grid storage resources are accessed via a Grid middleware that provides among others efficient protocols for data transfer and virtualizes the concrete storage resource. One result of the project is the integration of the GridFTP protocol [13] in the DataFinder system.

The GridFTP protocol is developed and maintained by the Globus Alliance and is a part of the Grid middleware Globus Toolkit [14]. In general the GridFTP protocol can be considered as an extension of the FTP protocol with the requirements of data transfers in Grid infrastructures. GridFTP for instance supports parallel, striped, partial or third-party data transfers. Moreover, it provides transfer monitoring and automatic restarts at specific transfer points. Additionally GridFTP supports the Globus security infrastructure that allows the usage of different data transfer security levels including encryption and integrity checks of transferred data.

The basis of the integration forms the GridFTP client reference implementation of the Globus Toolkit. Through the library pyGlobus [15] this reference implementation is available in Python. In general pyGlobus is an API allowing the usage of services provided by the Globus Toolkit. The encapsulation of the GridFTP client implementation is achieved by provision of a specific data store. The configuration of the GridFTP data store is centrally provided by the administration client. The administration client allows the specification of the whole range of options concerning authentication, performance and security. Thus the complexity of the data transfer protocol is completely hidden from the user.

This integration allows the usage of efficient secure data transfers and accessing storage resources provided by the Globus Grid middleware. The current integration is going to be extended to allow partial data transfers. Therefore the DataFinder Python API has to be adapted to provide the required user interface.

CONCLUSION

The approach of the presented system allows an efficient and flexible management of scientific data. This is achieved by provision of structure and meta information to the managed data according to a free-definable data model. Moreover, the described data store concept allows the flexible integration of heterogeneous storage resources and easy migration to another storage location. The details of the specific data transfer protocol are hidden, so the storage location is entirely transparent to the user.

The described usage scenario at DLR demonstrates the flexible extensibility of the DataFinder user client for a specific scenario via the provided script extension mechanism. In particular in conjunction with the implemented script extensions a close integration of the typical TRACE workflow with the data management task is achieved. The TRACE usage scenario is going to be enhanced with interfaces to the Grid middleware UNICORE 6 in the D-Grid community project AeroGrid [16]. In particular, the workflow management functionality is extended with the ability to use computation and storage resources provided by this Grid middleware. Moreover, the DataFinder is going to be extended to document and trace the detailed history of a computational process that leads to a certain result ("Provenance" [17]). The integration of a Provenance service which records detailed information of all conducted execution steps, increases the dependability of the results and improves the user's confidence in their quality.

In the last part of this paper the integration of the archive system Tivoli Storage Manager and the GridFTP protocol were presented. In general this has been accomplished by using the existing data store concept. Currently the TSM integration enables the DataFinder to archive single data files. This approach is refined to support archiving of complete collections. Furthermore, it is aimed to achieve an archiving process compliant to the OAIS reference model.

REFERENCES

[1] – Simulation and Software Technology: http://www.dlr.de/sc/ (2007)

[2] – A. Schreiber: The Integrated Simulation Environment TENT. Concurrency and Computation: Practice and Experience 2002; 14; 1553-1568 (2002)

[3] – E. Hoffmann, M. Wagner: Design of a Management Tool for Scientific Data at the DLR-Institute of Aerodynamics and Flow Technology: DataFinder. NAFEMS2003, Wiesbaden (2003)

[4] – D-Grid Initiative: http://www.d-grid.de/ (2007)

[5] – Software AG: http://www.softwareag.com/de/products/tamino/ (2007)

[6] – Catacomb: http://catacomb.tigris.org/ (2007)

[7] – Python Programming Language: http://python.org/ (2007)

[8] – Trollltech: Qt: Cross-Platform Rich Client Development Framework. http://trolltech.com/products/qt/ (2007)

[9] – Institute of Propulsion Technology: http://www.dlr.de/at/ (2007)

[10] – IBM Tivoli Storage Manager: http://www-306.ibm.com/software/tivoli/products/storage-mgr/ (2007)

[11] – Institute of Aerodynamics and Flow Technology: http://www.dlr.de/as/ (2007)

[12] – Consultative Committee for Space Data Systems: Reference Model for an Open Archival Information System (OAIS). CCSDS 650.0-B-1, http://public.ccsds.org/publications/RefModel.aspx (2002)

[13] – GridFTP protocol: http://www.globus.org/toolkit/data/gridftp/ (2007)

[14] – Globus Toolkit: http://www.globus.org/toolkit/ (2007)

[15] – Python Globus: http://dsd.lbl.gov/gtg/projects/pyGlobus/ (2007)

[16] – AeroGrid: http://www.aero-grid.de/ (2007)

[17] – L. Moreau, P. Groth, S. Miles, J. Vazquez, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga. The Provenance of Electronic Data. Communications of the ACM, 2007 (to appear)