Computer Science • 13 (4) 2012 http://dx.doi.org/10.7494/csci.2012.13.4.21

Thomas Röblitz

TOWARDS IMPLEMENTING VIRTUAL DATA INFRASTRUCTURES – A CASE STUDY WITH iRODS

Abstract *Scientists demand easy-to-use, scalable and flexible infrastructures for sharing, managing and processing their data spread over multiple resources accessible via different technologies and interfaces. In our previous work, we developed the conceptual framework VISPA for addressing these requirements. This paper provides a case study assessing the integrated Rule-Oriented Data System (iRODS) for implementing the key concepts of VISPA. We found that iRODS is already well suited for handling metadata and sharing data. Although it does not directly support provenance information of data and the temporal provisioning of data, basic forms of these capabilities may be provided through its customization mechanisms, ie rules and micro-services.*

Keywords data management, virtual infrastructures, metadata handling, provenance information, data provisioning

1. Introduction

Scientific discoveries increasingly require the storing and processing of vast amounts of data owned by international collaborations that need to share different data sets and analytical tools operating on them. Ensuring reproducibility of scientific results necessitates provenance information for the complete research lifecycle: from data taking or simulation to data analysis to publishing and eventually to its long-term preservation. State-of-the-Art data intensive computing utilizes world-spread storage and compute resources to implement different types of systems, APIs, programming models and security infrastructures. Over the last decade, research in Grid computing essentially followed two approaches to relieve the scientists from learning many of the resources' specific details: basic low-level services such as Globus toolkit [6], gLite [3] or UNICORE [23] and comfortable application specific portals such as MultiDark [13] or PANGAEA [14]. While the former (i.e., low-level services) provide a high degree of flexibility for implementing a wide range of scenarios, they typically require a deep understanding of the underlying concepts, and require many manual operations for analyzing data, its maintenance and preservation. In contrast, the latter (i.e., high-level application specific portals) do not require expert knowledge, but they are usually limited to very specific use cases.

In [18] we developed VISPA, a conceptual framework of a virtual infrastructure for storing and processing scientific data. The two key concepts of VISPA are: (1) views which encapsulate data in specific context and (2) the declarative description of views. We devised VISPA after studying applications from different scientific domains exhibiting various requirements on the sharing and processing of data. Besides flexible and dynamic data sharing schemes, the key requirements are to easily incorporate different types of resources (eg, clusters, Grids, Clouds, servers, PCs, laptops), support for different access methods / programming models (eg, flat file I/O, relational database operations, data parallel computing), and being able to capture the whole lifecycle of data, that is from data taking over filtering, combining, moving, analyzing, publishing to its long-term preservation. The key concepts of VISPA are implemented by a runtime system operating in a feedback loop that retrieves view descriptions from a store, monitors the state of data storage and processing, and compiles operations to let the descriptions eventually conform with the state of the data. We are not aware of any standard Grid computing toolkit or portal that provides such a complete data management solution. In the Cracow Grid Workshop series, research on virtual research infrastructures has seen some attention in recent years, for example ViroLab [10]. However, most of them focus on orchestrating workflows of compute activities instead of managing data sets.

This paper evaluates iRODS (Integrated Rule-Oriented Data System) [8] as the runtime system for implementing the two key concepts mentioned above. Particularly, we evaluated the four key features: (1) core and user metadata, (2) provenance information, (3) data sharing, and (4) temporal aspects of provisioning data.

The remainder of the paper is structured as follows. We give an overview of two studied application scenarios in Section 2. Section 3 briefly introduces the core ideas of the VISPA framework. Section 4 reviews the core components of iRODS. In Section 5, we demonstrate how the metadata model of VISPA can be implemented by iRODS. Thereafter, Section 6 evaluates the implementation of advanced features of VISPA such as sharing data and temporal aspects of provisioning data. Section 7 discusses related work. We conclude and outline directions of future work in Section 8.

2. Application scenarios

We introduce two application scenarios that led us to develop the conceptual framework of a virtual infrastructure for storing and processing scientific data (VISPA).

2.1. Constrained Local Universe Simulations

Constrained Local UniversE Simulations (CLUES) are handled through a semi-automatic workflow (cf. Fig. 1a). The simulations are run on remote HPC resources and generate 120–150 snapshots each approx. 6.1 TB in size. Today, all snapshots are copied to one (or more) storage centers which should additionally provide special resources for post-processing (eg, distributed databases for scalable data analysis or GPGPU hardware to create video sequences). Some post-processing methods may only be applied at special resources requiring additional data transfers and even re-transmissions if the data was removed from the simulation sites.

Today, most data management operations in the CLUES workflow [5] are **manually** performed by scientists. This results from the following observations: (1) the involvement of several scientists from different institutes, (2) the distributed processing of simulations at HPC centers, (3) distributed storage of significant amounts of data products and (4) subsequent post-processing steps.

2.2. Distributed parameter sweep simulations

Parameter sweep studies (PSS) are used to analyze data sets with a large number of parameter sets. Often scientists use their personal workstation to coordinate the execution of the PSS and process the results. The analysis itself is performed on parallel machines (local or remote SMPs or clusters) and available Grid infrastructures. Hence, already in the simplest case, data resides on distributed resources. The management of a PSS gets further complicated if certain "unexpected" events happen, for example the unavailability of processing resources, the need to partially re-run executions or the exchange of results with other researchers. As a result, additional data operations are required to maintain the desired progress of the study and – at the same time – ensure scientific standards (ie reproducibility).

Figure 1b illustrates the main components of a basic scenario: a personal computer (PC), a set of multi-core servers (SMP) and a storage which holds both the data to be analyzed and the obtained raw data results. The PC is used for creating

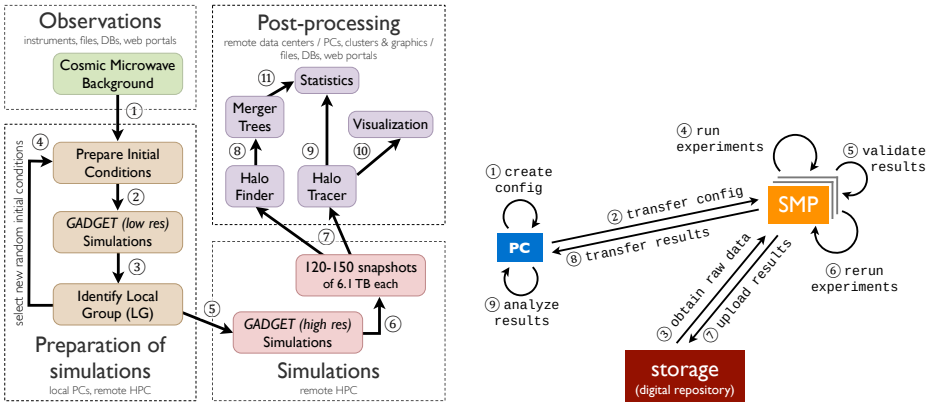

Figure 1. Application scenarios: (a) CLUES workflow, (b) parameter sweep simulation.

the configurations of the experiments and analyzing the validated results. The SMP (or any other parallel machine) is used for executing the experiments, validating their results and re-running experiments if necessary.

3. Virtual Infrastructure for Storing and Processing Data

The key idea of the VISPA framework is to organize data in **views** which are **declaratively** described. Views are compiled by the VISPA runtime to data management operations that are enacted at appropriate resources and times. The results of their execution is monitored and reflected in adjustment operations to maintain or obtain the desired set of views. We present the main aspects of views, how they are declaratively described and how they are managed at runtime.

3.1. Main aspects of views

A view encapsulates data in a specific context, which is defined by different categories (cf. Fig. 3): metadata (core and user defined), provenance, content, permissions, technology and resource mapping. Views are addressable (by unique identifiers) and stateful. Their life cycle may involve time periods when they are inactive or active. Figure 2 illustrates the possible states and allowed state changes. A scientist only has to declare a view and may use it in its active period. All state changes are managed by the VISPA runtime (for details on the state changes see [18]).

3.2. Declarative description of a view's content

As early as in the 1970ies, Shu et al. [22] developed EXPRESS to explore the idea of using high-level **non-procedural** languages for defining data (DEFINE) and for

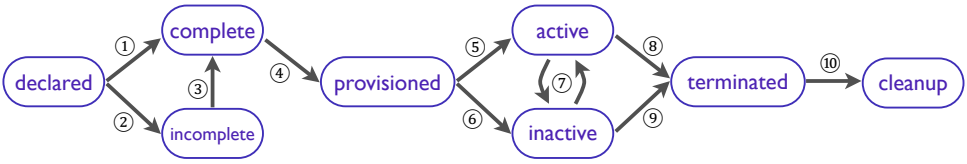

Figure 2. A view's life cycle begins in the state declared and ends in the state cleanup.

data restructuring (CONVERT). VISPA follows this idea to let scientists **declaratively** describe the target state of the virtual data infrastructure. In a feedback loop, VISPA's runtime system compares the current state of the data infrastructure with the described target and compiles necessary operations to maintain the desired data accessibility.

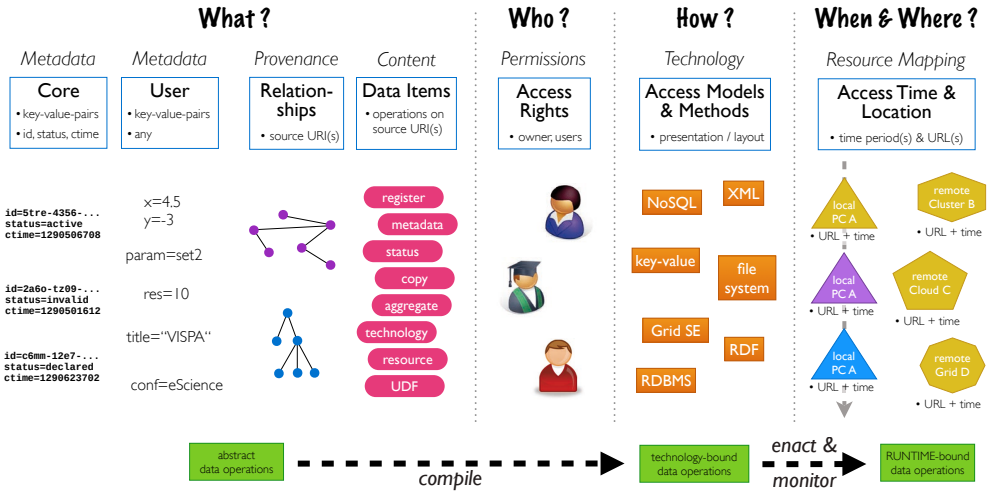

Figure 3. Overview of the main categories for declaratively describing views.

Figure 3 illustrates the different categories to describe a view (see [18] for a detailed description). A description of a view answers the four main questions:

- What is the content of a view?
- Who may access the data?
- How is the data accessed?
- When and where needs the data be accessible?

3.3. Runtime System for Managing the Views

Figure 4 illustrates the main components of the VISPA runtime system. Scientists use a graphical user interface (GUI) or command line interface (CLI) to declaratively describe the views and observe the status of the virtual data infrastructure. The

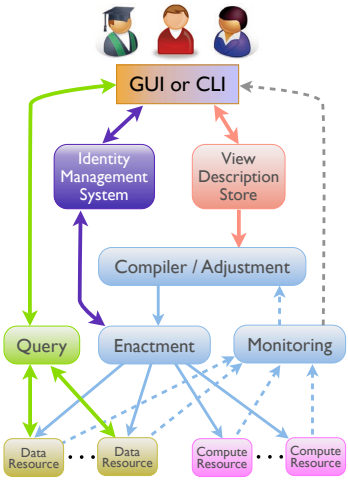

Figure 4. Components of the VISPA runtime implementing the views by interfacing the users and the resources.

Query component enables the automatic retrieval of metadata for a data set specified by its URL in the category *Resource Mapping*. Hence, scientists may be relieved from repeated and cumbersome manual inputs of existing information. The *Identity Management System* provides information about the identity of users to which a scientist may wish to grant certain data access capabilities. All views are stored in the *View Description Store* (VDS).

4. iRODS: The integrated Rule-Oriented Data System

The *integrated Rule-Oriented Data System* (iRODS) is a distributed, highly customizable system for managing data. In the following, we briefly introduce iRODS' data and the metadata model, its main architectural components, and its capabilities for customizing its behavior. Further details are revealed in sections 5 and 6.

iRODS logically organizes data in files and collections (of files and/or collections). Hence, collections are hierarchically structured. Both, files and collections are allocated to resources (hosted on storage servers) and are described by metadata. System metadata covers information such as creation time, logical and physical path, size, and access control lists. Arbitrary *attribute-value-unit* triples may be used to capture user metadata. Data may be replicated to several resources.

Figure 5 shows the four architectural components of iRODS. Data sets managed by iRODS are split into zones, each being served by a single metadata catalog (iCAT) and one to many data servers. Data may be stored on different types of resources such as traditional file servers, relational databases and storage provided as a service (eg Amazon S3 [1, 24]).

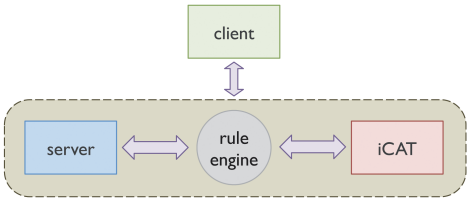

Figure 5. Architectural components of iRODS.

The **client** provides interfaces to the users and interacts with the other components to manipulate data and metadata as well as controlling the system's behavior via the rule engine.

The **iCAT** stores all metadata about the data, their hierarchical organization in collections, information about users and resources. It serves as the first contact to users for querying and locating data.

The **server** encapsulates the actual data storage. Servers host resources to which data is allocated. After obtaining the actual storage location of data from an iCAT, the client accesses the data directly at servers.

The **rule engine** provides a means to automatically execute maintenance operations and to customize the behavior of an iRODS environment. It allows us to execute rules that may manipulate data and metadata.

The behavior of iRODS is implemented by *rules* which are defined by developers (for iRODS basic behavior), by administrators (to customize a whole installation), and even users (to implement specific use cases). Rules may be invoked manually or automatically triggered by events such as uploading a file or an expiring timer. The core functionality – operations for manipulating data and metadata – available to rules is encoded in *micro-services*. There exists an extensive list of built-in micro-services as well as an API for adding new special-purpose micro-services.

5. Implementing VISPAs data and metadata model

We demonstrate the implementation of VISPAs data and metadata model with iRODS. In this study, we solely consider the access model *file system* in the category *Technology* of VISPAs model (cf. Fig. 3). Other access models such as relational databases are subject to future work.

5.1. Mapping core and user metadata

Table 1 shows the mapping of core metadata. User metadata in VISPA is simply mapped to *(attribute-value-unit)* triples in iRODS. The optional *unit* is not used.

Other system metadata in iRODS such as permissions and physical locations of data is not part of the core metadata in VISPA. However, the setting permissions is relatively simple and briefly discussed in Section 6.1. Arbitrary user metadata

Table 1
Mapping of core metadata in VISPA to iRODS.

VISPA	iRODS	Remark
`id`	`DATA_ID`	–
`status`	user attribute `VISPA_STATUS`	The internal persistent variable `DATA_STATUS` is not used to not compromise the semantics of iRODS.
`ctime`	`DATA_CREATE_TIME`	–

in VISPA is modelled as key-value-pairs. Hence, any user metadata can be easily mapped to iRODS user metadata which have the form attribute-value-unit (AVU). The *unit* of iRODS is only optional and not used in our mapping.

5.2. Provenance information

The off-the-shelf iRODS does not support provenance information yet. However, the relationships between views may be modelled by additional user attributes. For capturing provenance information, two aspects must be covered: (1) what views were the source of a view, and (2) what operation was performed on these views. The former may be implemented by an attribute named `source_views` which store a list of all view `id`s that are a source for this view. The latter is captured by an attribute named `init_op` which holds any built-in operation given as a keyword (eg `copy`, `change_permission`), a user-defined function (eg name of a script that was executed on all sources), or a descriptive string (eg to describe the relationship between complex views). As built-in operations and user-defined functions may be parametrized such information must also be stored (see the discussion on linking software executions to data sets in [26]). Hence, the value of the attribute `init_op` may be the name of a built-in operation or user defined function (UDF) executed without any parameter. If parameters need to be specified the value is an `id` of another view that corresponds to the operation or the UDF and gives parameters as user attributes (eg `param1`, `param2`,...).

5.3. Discussion

iRODS supports a hierarchical file system-like organization of files and collections. This feature may be useful for efficient recursive operations such as changing the access permissions of large views. Because, VISPA allows more flexible layered structures of views, it may not always be possible to organize the data hierarchically.

VISPA supports provenance information between views. Because such information is not modelled in iRODS, we propose to use two user defined attributes for them. This information can be used to traverse the data dependency graph. An issue for future work is the assessment of the performance and usability of our approach. Although, iRODS uses relational databases for storing the metadata a user cannot change the schema for efficiently storing and querying the information. Moreover, the

provided SQL-like interface does only support very basic queries, which would require an iterative procedure to reconstruct very deep provenance graphs.

6. Data sharing and temporal aspects of data provisioning

Advanced features include the sharing of data across iRODS zones and temporal aspects of provisioning data. Sharing is a very important requirement of contemporary science. Employing two or more zones for sharing reflects the observation that scientists belong to different administrative domains each potentially managing their own zone. Provisioning data at requested times enables the flexible management of the available resources for storing and transferring data.

6.1. Sharing data across zones

Setting up a federation of two zones A and B in iRODS is straightforward. Essentially, in zone A one must create a new remote zone pointing to B by specifying the host name and port of the remote iCAT, and vice versa. Additionally, accounts for the remote users need to be created. The name of the account is augmented by the name of the zone. Thus a scientist only needs a single account but could have different roles (eg administrator, user) in different zones.

If two scientists want to share data, they simply have to grant appropriate access permissions to each other. iRODS provides a simple command (`ichmod`) to change the access permissions of files and collections. Hence, VISPA only needs to wrap this command for a command-line interface or integrate it into a graphical user interface. Because the iRODS command may be used recursively on nested collections it is efficient to store nested views in hierarchically nested collections. Otherwise, changing the permissions of a view with several levels of sub-views would require to issue the `ichmod` command several times up to the number of sub-views of the view.

6.2. Temporal aspects of provisioning data

Provisioning data for a given (future) period of time may not be efficiently performed manually. Such an approach will lead to data provided too late (ie delaying activities that require the data) or too early at the expense of blocking scarce resources. Therefore, VISPA lets a user simply declare the temporal requirements and the runtime issues the necessary operations at appropriate times. The temporal requirements can be just a start time, a start time plus a duration or a sequence of multiple intervals. iRODS does not

support such temporal aspects out of the box. However, by means of iRODS' *delayed execution services* rules may be executed at a specified time. We can exploit this feature to make data available during given periods of time.

The rule that performs the necessary operations just requires information about the input data and the location to where the output data is stored. Although that addition is straight forward, it also has a drawback. Until the data operation is

finished, iRODS has no information about the data. Especially, users may not see the data if they query the system.

6.3. Discussion

Sharing of data is easily enabled employing iRODS' federation capabilities. The basic scheme introduced above may be even extended by exploiting the rule system to replicate and synchronize shared data to achieve improved performance/fault tolerance and provide consistent data content, respectively. Because, iRODS requires fully qualified host names (ie with a DNS entry), configuration must adapt at each partner zone if one member changes its name. Particularly for mobile machines (eg laptops, virtual machines) that aspect requires manual operation.

Albeit temporal provisioning of data is not explicitly supported it may be implemented by exploiting iRODS' delayed execution service. A drawback of that approach is that the data available in the future is not known to iRODS' iCAT until the delayed operation has been performed. Alternatively, one could register the data with iRODS but do not provision the data immediately. This, however, leads to an inconsistent state between the iCAT and the storage, resulting in access failures at clients. In the future, we will explore two approaches for solving that issue: (1) using compound resources (eg similar to the combination of a cache and an archival system), and (2) integrating VISPA's data life cycle management (cf. Fig. 2) into iRODS. The latter, however, would require us to update all command-line tools and APIs of iRODS and make the users aware of this change.

7. Related work

Fedora [11] is a framework to build digital repositories for managing and sharing digital objects based on the abstractions proposed by Kahn and Wilensky [9]. Fedora uses RDF [17] for representing metadata and thus readily supports core and user metadata as well as provenance information in VISPA. In [12], Marciano et al. studied an integration of Fedora and iRODS. They found that both systems can be integrated to provide interoperability between digital repositories, especially wrt. data sharing. Integrating policies for managing the runtime behavior requires, however, additional research. Over the past decade, many projects have implemented data management and processing environments fitted to the specific needs of their domain. For example, the climate community developed the Generation N Data Management System [4] built upon the Globus toolkit [6] to implement work spaces and timely provisioning of data that is negotiated by a broker. Similarly, MyLEAD [15] extends the Globus Toolkit metadata catalog to let Geo scientists explore huge volumes of atmospheric data. The DataFinder [20] is interfaced with UNICORE [23] to allow organizing data and associate metadata with it.

At a more abstract, technology-agnostic level *digital repositories* are being developed in the arts and humanities domain to manage data products and their relationships [2, 16]. The emphasis of such repositories is on modeling the semantics of data

and curating research results. Furthermore, repositories may be federated to support global collaboration and interdisciplinary research.

At file storage level, distributed file systems such as the network file system (NFS) [19] enable access to files from different nodes and also the easy sharing of files among users. Environments demanding high-performance file access are typically build upon parallel file systems such as Lustre [21] and Ceph [25]. Wide-area network file systems such as XtreemFS [7] must cope with slow or unreliable components causing excess delays or network partitioning.

8. Conclusion and future directions

Science is increasingly based on exploiting digital information not only of huge volumes like for LHC experiments, but maintained by various systems with different interfaces at remote locations and owned by different stakeholders. Today, already basic tools exist to **manually** construct a virtual infrastructure for managing and processing the data. The main issues with the current modus operandi are the required level of understanding of the basic technology and the lack of automation. VISPA is a conceptual framework for letting scientists focus on the use of their data, but let the runtime system take care of all the technology details and the execution of operations to implement the needed virtual infrastructure. iRODS provides a sound basis for implementing the key concepts of VISPA. Although it does not support provenance data explicitly, one may utilize metadata with specific attributes. Data sharing is enabled through iRODS's federation capability. Temporal aspects of provisioning data may be implemented by exploiting the delayed execution service for rules.

Future work is related to aspects which we think need improvements or require different approaches. These are: (1) the efficient support of provenance information, (2) the enhanced support for mobile machines, and (3) the better integration of temporal provisioning of data.

References

[1] Amazon Simple Storage Service (Amazon S3), December 2011.

[2] Aschenbrenner A.: *Reference Framework for Distributed Repositories — Towards an Open Repository Environment.* PhD thesis, Georg-August-Universität Göttingen, Göttingen, 2009.

[3] gLite — Lightweight Middleware for Grid Computing, September 2011.

[4] GNDMS — Generation N Data Management System, September 2011.

[5] Gottloeber S., Hoffman Y., Yepes G.: Constrained local universe simulations. In *Proc. HPCSE '09, Garching, Germany*, 2010.

[6] The Globus Toolkit, September 2011.

[7] Hupfeld F., Cortes T., Kolbeck B., Stender J., Focht E., Hess M., Malo J., Marti J., Cesario E.: The XtreemFS architecture – a case for object-based file systems in Grids. *CCPE*, 20(17):2049–2060, 2008.

[8] iRODS, December 2011.

[9] Kahn R., Wilensky R.: A framework for distributed digital object services. *International Journal on Digital Libraries*, 6:115–123, 2006.

[10] Kasztelnik M., Gubala T., Malawski M., Bubak M.: Development and execution of collaborative application on the virolab virtual laboratory. In *Proc. of the Cracow Grid Workshop 2007*, Cracow, Poland, pp. 41–46. ACC CYFRONET AGH, 2007.

[11] Lagoze C., Payette S., Shin E., Wilper C.: Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6:124–138, 2006.

[12] Marciano R., Moore R. W., Zhu B.: Enabling inter-repository access management between irods and fedora. In *Proc. of the 4th International Conference on Open Repositories*, Atlanta, Georgia, USA. Georgia Institute of Technology, 2009.

[13] MultiDark Database, December 2011.

[14] PANGAEA — Data Publisher for Earth & Environmental Science, December 2011.

[15] Plale B., Gannon D., Alameda J., Wilhelmson B., Hampton S., Rossi A., Droegemeier K.: Active management of scientific data. *Internet Computing, IEEE*, 9(1):27–34, 2005.

[16] Razum M., Schwichtenberg F., Wagner S., Hoppe M.: eSciDoc infrastructure: a fedora-based e-Research framework. *Research and Advanced Technology for Digital Libraries*, pp. 227–238, 2009.

[17] RDF Primer. *W3C Recommendation.* Technical Report, February 2004.

[18] Röblitz T., Enke H., Riebe K., Fritzsch B., Klump J.: *A vision for virtual infrastructures for storing and processing scientific data.* Technical Report 839, University of Technology Dortmund, Germany, September 2011.

[19] Sandberg R., Goldberg D., Kleiman S., Walsh D., Lyon B.: Design and implementation of the sun network filesystem. In *Proc. of the Summer 1985 USENIX Conference*, pp. 119–130, 1985.

[20] Schlauch T., Eifer A., Soddemann T., Schreiber A.: A data management system for unicore 6. In H.-X. Lin, M. Alexander, M. Forsell, A. Knüpfer, R. Prodan, L. Sousa, A. Streit, eds., *Euro-Par 2009 Parallel Processing Workshops*, vol. 6043 of *Lecture Notes in Computer Science*, pp. 353–362. Springer, Berlin / Heidelberg, 2010.

[21] Schwan P.: Lustre: Building a file system for 1000-node clusters. In *Proc. of the 2003 Linux Symposium.* Citeseer, 2003.

[22] Shu N. C., Housel B. C., Taylor R. W., Ghosh S. P., Lum V. Y.: Express: A data extraction, processing, and restructuring system. *ACM Trans. Database Syst.*, pp. 134–174, 1977.

[23] UNICORE — Distributed computing and data resources, September 2011.

[24] Wan M., Moore R., Rajasekar A.: Integration of cloud storage with data grids. In *Proc. of the Third International Conference on the Virtual Computing Initiative*, Research Triangle Park, NC, USA, 2009.

[25] Weil S. A., Brandt S. A., Miller E. L., Long D. D. E., Maltzahn C.: Ceph: a scalable, high-performance distributed file system. In *Proc. of the 7th symposium on Operating systems design and implementation*, OSDI '06, pp. 307–320, Berkeley, CA, USA, 2006. USENIX Association.

[26] Yang E., Matthews B., Wilson M.: Enhancing the core scientific metadata model to incorporate derived data. *eScience, IEEE International Conference on*, 0:145–152, 2010.

Affiliations

Thomas Röblitz
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, Germany,
thomas.roeblitz@gwdg.de

Received: 19.12.2011
Revised: 24.07.2012
Accepted: 3.09.2012