IST-2001-32133
GridLab - A Grid Application Toolkit and Testbed

Deliverable D8.7 - Evaluation of Data Management & Visualization Services

Author(s):	Felix Hupfeld, Andrei Hutanu, Thorsten Schütt, Brygg Ullmer, Stefan Zwierlein
Document Filename:	
Work package:	WP 8 - Data Handling and Visualization
Partner(s):	
Lead Partner:	ZIB
Config ID:	GridLab-08-DATA-0007-M30
Document classification:	IST

Abstract: This document provides an evaluation of data management application prototypes discussed in D8.9, and reports on the results for WP8's data management and visualization software and services.

Contents

1 Introduction

This document reviews the Data Management Services in the context of the Requirement Analysis document (D8.1). We will consider the compliance to defined functional and non-functional requirements, the system's suitability for the use cases, and the software's quality.

The remaining sections of this document are organized by services as they were specified in our Technical Specification document (D8.2). To provide the reader with the needed context, we will first list the use cases and requirements as they were specified in D8.1. For each use-case and functional requirement, we will list the services that are responsible for their implementation.

2 General Remarks

Since the Requirement Analysis (D8.1) and the Technical Specification (D8.2) were written, the field of information technology has continued its usual pace of advancements. For the Data Management system, this influenced the choice of underlying technology and frameworks. While Grid and Web Service standards were not very mature when the specifications were written, Web Service in particular have grown into widely supported technology.

We decided to base our software on W3C Web Service standards (SOAP, WSDL) where possible, and to rely on the Grid's defacto-standard GSI for authentication, delegation and encryption. The use of these technologies eases interoperability with other software and opens a wide range of tool support for software development. This choice also obsoletes the considerations for the Generic Service Architecture that was specified in D8.2. As our services are now reachable via standard interfaces, there is no more need for an abstraction layer inside the services. GridLab's flexibility of being able to work on different Grid systems is not affected because this abstraction layer is still present in the Grid Application Toolkit (GAT).

3 Use Cases

3.1 Data Management

A number of driving data management use cases were defined in D8.1. These are summarized in Table 1.

3.2 Visualization

A number of basic use cases were defined in D8.1. These are summarized in Table 2. In practice, we have divided these into four classes of functionality.

3.2.1 Access to grid services from visualization tools

First, there are many use cases where scientists wish to access grid services from within existing visualization tools. One common use case is browsing and loading remote files. One rationale for this is to access large datasets which are unable to be fully accommodated on the local workstation. Toward this, another use case involves remote hierarchical and partial file access, where temporal, spatial, or successive level-of-refinement subsets of the remote dataset are retrieved and visualized.

Table 1: Data Management Use Cases as defined in D8.1.

Identifier	Description	Responsible software
DMS1	migration of data files from A to B	Data Movement (DM)
DMS2	accompanied selection and migration of data files, from A to B	DM; Replica Catalog
DMS3	fast transport of data sets/files from A to B,	DM
DMS4	discover data sets,	Metadata Catalog (MDC)
DMS5	locate data sets,	RC
DMS6	archival of data files,	DM & MDC
DMS7	recombination of parted data sets/files,	– (§10)
DMS8	requests information about a data set or data file and its contents/history/...	MDC
DMS9	requests information about a storage system and its optimal I/O parameters/variables,	RC using adaptive

Table 2: Visualization Use Cases as defined in D8.1.

Identifier	Description	Responsible software
VS1	visualization of past data sets	grid plugins to viz software (§9.1); viz reports (§9.2.2)
VS2	visualization of online data	grid plugins to viz software (§9.1)
VS3	visualization output for further use	viz reports (§9.2.2)
VS4	visual interaction with simulation code at runtime	grid streaming from viz apps (§9.2.3); param steering from viz artifacts (§9.3.2); live infrastructure visualizations (§9.4)
VS5	visual interaction with the Grid environment	live infrastructure visualizations (§9.4)

Table 3: Functional Requirements as defined in D8.1.

Identifier	Description [1]	Responsible software
FR1	migration of data	§7
FR2	archival of data	§6 & §7
FR3	replication of data	§5
FR4	recombination of distributed/parallel written data	– (§10)
FR5	annotation of data	§6
FR6	location of data [2]	§5
FR7	discovery of data [3]	§6
FR8	remote (online and offine) visualization of data	§9.1
FR9	adaptive visualization (progressive, hierarchical) of data	§9.1
FR10	data transformation [4]	§9.3.1
FR11	data extraction [5]	§8 and §9.1.2
FR12	secure data transport [6]	– (§10)

3.2.2 Access to visualization tools from grid services

Another set of use cases involve accessing remote visualization tools via grid services. A common use case involves accessing automatically-rendered visualizations via the portal, where the visualizations are triggered after (and perhaps during) job execution. Toward this, our *visualization service* supports remote visualization automation. Among its features are support for the generation of *visualization reports*, which structure visualization results within an HTML layout. Finally, *visualization streaming* supports remote access to live visualizations via common videoconferencing protocols.

3.2.3 Access to visualization tools via grid-enabled mobile devices

Scientists may wish to monitor the progress of long-running simulations while "on the move." Thus, access to visualizations from mobile devices such as PDAs or even cell phones/"handies" can be of value. Additionally, scientists may wish to access and control visualizations from within immersive or collaborative environments, such as virtual reality and videoconferencing. These environments often pose eyes-busy demands, frequently in the absence of a keyboard or mouse, and motivate the utility of new kinds of physical grid-enabled interaction devices.

3.2.4 Visualization of grid infrastructure

Scientists, administrators, and grid developers may wish to understand and monitor underlying intercommunication between grid services. Toward this, support for live and recorded-playback diagrams of grid service intercommunications can hold value.

4 Requirements

4.1 Functional Requirements

Table 4.1 presents the functional requirements defined for WP8 in D8.1. The "responsible software" column identifies the sections of this document which discuss how we have addressed each requirement.

Table 4: Non-functional Requirements as defined in D8.1.

Identifier	Description
QR1	be application oriented
QR2	be usable on all types of resources
QR3	be usable in firewalled environments
QR4	be usable in disconnected environments
QR5	be usable in minimalistic environments
QR6	support and enforce use of security policies
QR7	support synchronous and asynchronous operation
QR8	provide abstractive interfaces
QR9	provide complete set of interfaces
QR10	provide ability to be discovered
QR11	provide ability to discover and use services dynamically
QR12	provide audit trails and verbose error messages
QR13	provide a test suite
QR14	be well documented
QR15	be extensible and 'future proof'
QR16	be informative, transparent
QR17	be lightweight (where possible)
QR18	be robust and fault tolerant
QR19	be adaptive to variable Grid infrastructure
QR20	be able to recover from interruptions
QR21	be independent from other services (if possible or
QR22	behave consistent and reproducible
QR23	allow integration of 3rd party software/services
QR24	allow instrumentation for monitoring purposes.

4.2 Non-functional Requirements

As each service maps directly or indirectly to an abstraction in the Grid Application Toolkit (GAT), the services are compliant with QR1 and QR9. QR3 is ensured by SOAP's use of HTTP as the transport protocol. QR10 and QR11 are handled by the use of the service registration and discovery facilities of WP-10. We implemented unit tests for the services' external interfaces (QR13) which are run constantly, ensuring the permanent functionality (QR22). In the remaining months of the project, we will extend the current documentation (QR14). All services have logging facilities that can be tailored to the user's needs (QR12).
The compliance to other non-functional requirements is evaluated in the section of the respective services (where applicable).

5 Replica Catalog

The replica catalog was designed based on the scenarios DMS1, DMS2, and DMS6 (FR3, FR6). To support future enhancements, we invested substantial effort in realizing a modular design. This design allows us to replace existing components, as well as to add new components without significant overhead. This approach proved very useful during the development. As a consequence of its functionality and underlying design decisions, the catalog is currently in use by

several projects at ZIB, and has become a research platform for distributed data management. The larger user base and developer team provide the necessary resources to implement better user support and to do more thorough testing.
User feedback from GridLab and non-GridLab users, as well as the integration with the GAT, have indicated that we have fulfilled the requirements stated in D8.1.
The catalog is written in ANSI C++. For communication with users and other services we use standard protocols such as CORBA (omniORB) and SOAP(gsigsoap). The curent implmentation was tested on a variety of Linux distributions on IA32 and IA64.

5.1 Attribute Management

In the specification phase of the project, we envisioned the management of file and file collection attributes to be part of the Replica Catalog's responsibility. In the meantime, further investigation showed that attribute management is generic enough to become a stand-alone component and thus, it has been refactored subsequently into a more general Metadata Catalog. As a result, this Metadata Catalog is now not only used for file and file collection metadata, but also serves as the persistent communication backend for the Grid Application Toolkit's (GAT) Advert Service abstration, and gathers message metadata of WP-12's Messaging Service. Please refer to the Metadata Catalog section 6 for a detailed evaluation of this component.

6 Metadata Catalog

The Metadata Catalog (MDC) is a database system that stores attribute-value pairs for data references. As it does not deal with the data itself, it can keep the metadata for all types of persistent data that is stored somewhere in a Grid. Currently, Gridlab's MDC stores metadata of files, file collections and messages. Additionally it keeps all GAT objects that were made persistent via the GAT's Advert Service abstraction.

An MDC is is used primarily as an archive. Data that is generated by computing applications or by sensors is stored on data servers and its description/metadata is registered in the MDC (FR2, DMS6). Registration can take place either directly by using the service's Web Service interface, or by using the GAT's Advert Service abstraction. The registered metadata must comply to the namespace's metadata schema, which is extensible on demand, and can keep arbitrary metadata (DMS8). Subsequently, data can be discovered by querying the metadata (FR7, DMS4).

Metadata is kept in one of several namespaces that can be added on demand. Each namespace has its own access policies, and thus can be configured flexibly as (e.g.) public data archives, project or task specific collaboration spaces for a small group of researches, or as private namespace for annotation and personal information management (FR5). This concept is used for example by WP-12's (Mobile) messaging service to store the metadata of received and sent messages in the user's private namespace. Having one place that gathers all metadata of persistent data in the Grid, the data can be searched and organized using one interface.

The Metadata Catalog is highly portable (currently Win32, Linux, Mac OS X) and can thus be deployed on most available devices, including servers and mobile systems (QR4, QR5). The Metadata Catalog as an access policy interface where administrators can chose from a wide range of access policies or easily implement their own ones (QR6).

The current external interface of the MDC consists only of small grain functions, which are executed very quickly. Therefore, an asynchronous version of the functions is not needed. We plan, however, to add a function for bulk import of metadata to the interface, which then will have an asynchronous version (QR7).

While we have not done any formal software quality assurance, we use a development model with short turnaround cycles that produces runnable software. Revisions of the MDC have been running and undergoing testing in Gridlab's Testbed environment for over 20 months. The code itself is written in modern C++ making use of best practices like design patterns and unit tests.

7 File Services

Currently, two file services are deployed: the file movement service, which handles third-party transfer of files; and the file browsing service, which handles operations involving only one remote machine (directory listing and creation, retrieval of information about files).

Unit tests were provided for both services, and a unit test for the file movement services was implemented and deployed on:
`http://www.gridlab.org/WorkPackages/wp-5/testbed/datamovement.html`
by the testbed group.

Given the continuing success of the unit tests and the feedback from our users (GRMS, the C and Java GAT, and the visualization services) we consider that the file services are very reliable in providing the functionality that they were designed to offer. We experienced some problems during the development process with the underlying libraries. Some of them were fixed which determined the migration of the services to the newest version of Globus (3.2), and for others we have found acceptable workarounds.

We believe the usability of the service is very good. The service package is easily configurable and installable, the interfaces are well documented, and example clients are available. The software and detailed interface documentation is available from our web page:
`https://www.gridlab.org/WorkPackages/wp-8/`

In terms of development effort and code quality, we think that it will be useful to combine the two file services into a single larger service for the final release. This will also help to extend the functionality of the service by combining operations offered by the (currently) separated services.

We measured the performance of the file movement service on the GridLab testbed to examine the effect of TCP parameter tuning using the predictions given by the adaptive service on the file transfer performance. The results are available in Table 5 and in graphical form in Figure 1

8 Remote File Access

During the development of the visualization components we noticed that the performance of traditional remote file access methods are very poor. For visualization, a typical use case may

Source Host	Destination Host	Transferred Data (MB)	Time (s) Not Optimized	Time (s) Optimized	#Parallel Streams	TCP Buffer Size (bytes)
litchi.zib.de	skirit.ics.muni.cz	310	126	30	9	65535
litchi.zib.de	hitcross.lrz-muenchen.de	310	134	150	5	131071
litchi.zib.de	gridentry.uni-paderborn.de	50	77	36	6	65536
litchi.zib.de	rage1.man.poznan.pl	310	319	61	9	131071
litchi.zib.de	n0.hpcc.sztaki.hu	310	164	36	9	65535
litchi.zib.de	fs0.das2.cs.vu.nl	300	163	39	5	131071
litchi.zib.de	mike4.lsu.edu	102	284	78	16	65536

Table 5: Each column represents a transfer operation, with source, destination, file size, number of seconds needed to transfer the file using untuned third-party GridFTP, and number of seconds needed for the transfer using tuned GridFTP and the number of parallel streams and the buffer size for the tuned transfer. The operations were performed using the file movement service which uses the adaptive service to retrieve estimates for the number of streams and buffer size parameters

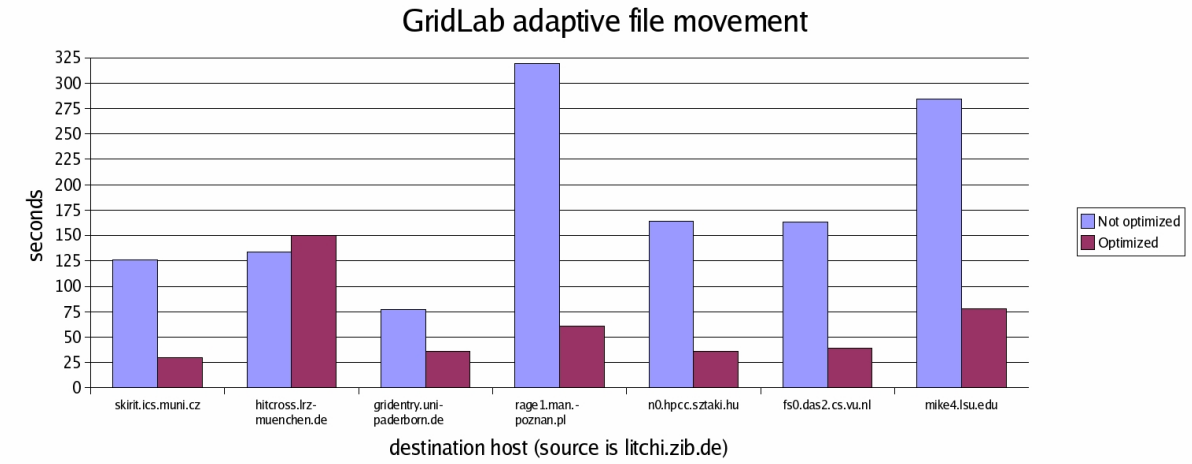

Figure 1: *The picture shows how the usage of parallel streams and tuned TCP buffers improves the performance of the file movement service*

infolve file operations as simple as obtaining a downsampled version of remote data. Traditional methods like remote partial file access perform particularly badly in this scenario. We developed a remote file access service based on GridFTP which is optimized for regular access patterns while still performing comparable to other remote methods while performing traditional (read/write/seek) remote I/O.

Table 6 shows the performance of GridFTP partial file access (GPFA) vs. our method (RemFALLS). The experiments were performed on the GridLab testbed using machines in Amsterdam and Berlin. The network latency is about 15-20ms. From the table, it can be seen that the performance numbers are strongly connected to the fact that GPFA has to perform a large number of remote operations, while RemFALLS needs only a single remote call to read a regular selection.

More details on our remote file access methods can be found in [3].

Driver	Cache Size	Transferred Data	#Remote Ops	Time / s
GPFA	-	4 B	1	0.83
GPFA	-	256 B	64	18.09
GPFA	-	16 kB	4,096	1,202.64
GPFA	-	128 kB	32,768	11,119.82
GPFA	1 kB	4 B	1	0.95
GPFA	1 kB	256 B	64	17.84
GPFA	1 kB	16 kB	2,048	70.40
GPFA	1 kB	128 kB	8,192	278.91
GPFA	1 MB	4 B	1	1.45
GPFA	1 MB	256 B	2	3.87
GPFA	1 MB	16 kB	8	7.14
GPFA	1 MB	128 kB	8	7.38
RemFALLS	-	4 B	1	1.14
RemFALLS	-	256 B	1	1.20
RemFALLS	-	16 kB	1	1.14
RemFALLS	-	128 kB	1	1.25

Table 6: Results of the performance evaluation remote accesses to a large binary data set. The various transfer sizes for partial access correspond to different resolutions of the resulting data selection (1^3, 4^3, 16^3 and 32^3, ...).

9 Visualization Services

WP8 deliverable 8.1 defined five high-level use cases for visualization within the GridLab project (Table 2). However, while the visualization efforts have been discussed in previous quarterly reports, we have not elaborated on these in prior deliverables. Given this, we report on the visualization efforts here in more detail than the other work package components, which have been documented in past deliverables.
As first introduced in our quarterly report #5, our core visualization efforts consist of three major components. These are:

1. Access to grid services (data services, etc.) from mainstream visualization tools
2. Access to mainstream visualization tools from the grid environment
3. Grid support for visualization involving mobile devices

In addition, another component – live visualization of grid infrastructure – was added late in the project, in response to demands from both within and outside of GridLab.
In the following sub-sections, we introduce our research results in each of these major areas. We defer detailed documentation of these components to our final deliverable (D8.10).

9.1 Access to grid services from visualization tools

The first major component of WP8's visualization efforts is making grid resources available from within existing mainstream visualization environments. Here, our efforts focused on extensions to Amira, a mainstream visualization environment under development at ZIB). We implemented two major grid-based features within Amira: support for remote file browsing and loading, and support for remote hierarchical and partial file access.

9.1.1 Remote browsing and loading

For more than two decades, users have taken for granted the ability to load, save, and browse files from any application through standard "File" menubar commands. However, to date, few applications (outside of Web browsers and dedicated file transfer programs) provide comparable abilities to access and browse remote file spaces. This is important for several reasons. The first is for reasons of convenience. In general, people are busy. Requiring users to leave an application to download data with an external application is an inconvenience, and can reasonably be expected to decrease use of remote data.

In the use cases of GridLab's astrophysicist collaboration partners, the importance of integrated remote file access and browsing is even more important. Here, the remote files of interest, often gigabytes or terabytes in size, are frequently too large to be downloaded locally (both for reasons of network bandwidth and local file storage capacity). Correspondingly, these files must be directly downloaded by the visualization application itself, using hierarchical and partial-file access techniques such as discussed in §9.1.2. As a result, there is a strong motivation for browsing remote files directly from within the visualization environment.

In collaboration with the DFN GriKSL project, we successfully implemented these features within the Amira environment. We implemented a remote file access module which cleanly integrates with the mainstream Amira release. In addition to manual URL specification, it supports browsing of partially-specified URLS, in a manner directly analogous to user interaction with local file space. As a convenience measure, our module caches previously-accessed URL's, which reduces the need for retyping (often long) remote paths.

These features have been released, supported, and used among the AEI astrophysicist community of GridLab scientific collaborators. (It has not been "deployed" on the testbed, as the Amira software itself is not deployable in this fashion.)

9.1.2 Partial file access for visualization

Numerical simulations (e.g., the numerical relativity simulations performed within the Cactus simulation framework) or image acquisition systems (e.g., as used in 3D medical scans) produce datasets of ever-growing sizes. In the context of remote collaboration, the growth of dataset sizes can make the replication of the datasets of interest to all of the participating scientists very difficult in practice. Consequently, the data often remains at at the location where it was generated, or is copied to a central storage system (sometimes using data tapes as the fastest effective transport mechanism).

In order to provide meaningful visualization of large remote datasets, efficient partial file access mechanisms are needed. The new techniques should minimize the requirements on local hard disk space, overall transfer time, and network bandwidth to values that allow an end-user with (e.g.) a laptop and broadband network connection to analyze his data.
Toward this, we developed two parallel directions. A generic approach designed to work with many binary file formats which uses regular byte pattern descriptions for the file selections was described in detail in our deliverable D8.5.

The second approach, significantly influenced by the existing visualization systems (Amira) and its support for the HDF5 file format was to develop a file-format dependent remote file access schema. In short, an efficient remote read mechanism was developed for HDF5 that combines

the many small operations needed to gather information about various datasets (metadata) stored in a file into a single remote operation and executes the remaining H5Dread operations separately on the remote side.

The strong performance yielded by this approach (Table 7) made the usage of interactive and progressive visualization techniques possible for HDF5 data accessed over the grid. Table 7 shows partial results (more details in [1]) of the performance measurements made for accessing, loading, and displaying a large remote HDF5 data set. We compare the performance obtained using our approach (*GridFTP HDF5*) with a comparable remote access technique: HDF5 over GridFTP partial file access (*GridFTP PFA*). For comparison, we also include measurements of local (*local access*) and Network File System (*NFS access*) times.

The usage of the GridLab technology for remote visualization in the GriKSL and Bone3D projects is documented in detail in [1] and [2] respectively. The software and detailed documentation is currently available from http://www.zib.de/visual/projects/gridlab/hdf5/

Access Type	Net	Meta Data t_1	Root Block t_2	Complete t_3
local access	-	7 sec	1 sec	3 sec
NFS access	LAN	8 sec	5 sec	8 sec
GridFTP HDF5	LAN	11 sec	2 sec	11 sec
GridFTP PFA	LAN	165 sec	10 sec	200 sec
GridFTP HDF5	WAN	14 sec	3 sec	68 sec
GridFTP PFA	WAN	430 sec	53 sec	960 sec

Table 7: The table lists performance measurements for the various access techniques we explored. The results have been obtained by timing the visualization process for a 32 GB HDF5 file, containing 500 timesteps, each timestep with the resolution of 256^3 data points (double precision).

In Table 7, t_1 is the time needed to gather and transfer the HDF5 meta data (needed to parse the HDF5 file). t_2 is the time needed to filter and transfer the subsampled first timestep. t_3 gives the access time for a full resolution time step (excluding t_1). The tests have been performed on the GridLab testbed using machines in Berlin and Amsterdam.

9.2 Access to visualization tools from grid services

The second major component of our visualization efforts is a set of mechanisms which provide access to visualization tools and results via Grid-based services and software. This functionality is delivered via three mechanisms: the visualization service, visualization reports, and visualization streaming.

9.2.1 Visualization service

The visualization service retrieves data files for generating plots and images from a remote site or the local machine. It then manages the generation of resulting images, and deliver these as HTML reports in local filespace or on a Web-accessible visualization server. This latter functionality is realized in combination with the visualization report functionality, which is described

in §9.2.2.

Interaction with the visualization service takes place in several ways. First, it is usable on local files directly from the command line. Secondly, it may be remotely invoked both via a Globus gatekeeper service, and through a SOAP interface through a SOAP wrapper of the service. These latter mechanisms are used by the GridLab portal (GridSphere). The portal both automatically triggers the generation of visualizations via Gatekeeper calls, and provides Web-based access to these visualization results.

If the requested visualization requires the transfer of data resources from remote file servers, the grid file movement service is invoked to transfer the data files to the visualization server. The service then triggers the generation and transformation of these images (discussed in §9.2.2), and stores the images within its local web-accessible space.

The visualization service is currently integrated with the GridSphere portal. This has been demonstrated at GGF10 (March 2004), as well as in our March EC project review. The local version is deployed at the site of our astrophysicist collaborators, and is now entering active use.

9.2.2 Visualization reports

The visualization service is responsible both for data transfer and data visualization. However, the identity of files and the visualizations performed to them may vary widely, and must be specified to the service. In addition, it is often desirable to integrate the resulting images into a "report," which can provide contextual names and parameter information, as well as aggregate multiple images into a single screen view.

Toward this, we have developed a method for specifying the behavior of the visualization service, and describing the parameters and layout of visualization reports. This is done through parameter (.par) files, which have evolved as a domain-independent specification format within the Cactus framework. The parameter file can specify one or more visualization service transformation scripts, used for rendering 1D, 2D, or 3D visualizations. The parameter file is also used to describe visualization reports.

We will illustrate several examples outputs from this report generation facility. Figure 2 illustrates a "top-level" report generated by the visualization reporting facility. This integrates several 1D, 2D, and (potentially) 3D visualization images. In addition, it provides contextual data and parameter information describing the associated simulation run.

Figure 3 illustrates output from the 1D visualization service processing script. This currently processes `.xg` (xgraph) and `.asc` (ASCII) files, as used by our astrophysicist colleagues. It then uses custom scripts for controlling the GNUplot program to transform these data files into `.gif` graphic files (internally using `.eps` files as an intermediary format). The script then integrates tens or even hundreds of individual images into an aggregated output, automatically organizing these into visual hierarchies which are heuristically detected based upon file name formatting.

As a variation on this, another visualization service script takes a series of images as input. An example output is illustrated in Figure 4. These images are often produced by visualization programs such as OpenDX, which are already in use by our astrophysicist colleagues. These are then automatically transformed into "thumbnails," and visually organized in the same fashion

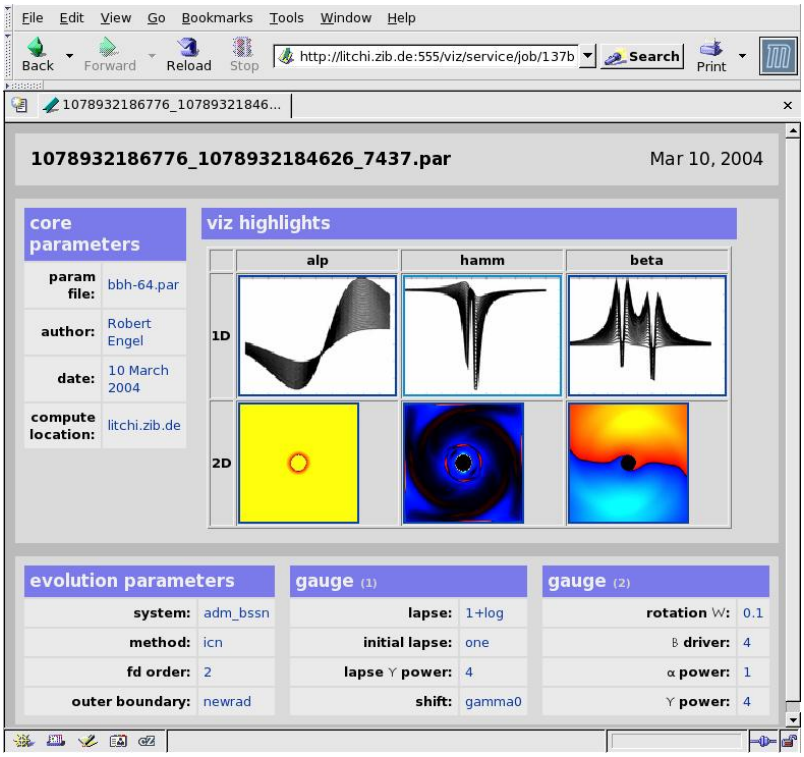

Figure 2: Visualization report: overview

as 1D graphical output.

As a final variation, we have also worked to provide automated 3D data visualization within the visualization reports. Most of these efforts have been in conjunction with the Amira software and supporting grid services discussed in §9.1. Here, we encountered a major standing challenge of the computer graphics volume rendering community. Specifically, it is widely recognized that the automatic determination of effective color and opacity maps for volume rendering is a fundamental, difficult, and essentially unsolved problem.

We successfully demonstrated automatic 3D visualization of some sample data sets widely used as test cases by our astrophysicist colleagues. An example output is shown in Figure **??**. However, we found that the underlying code did not readily generalize to many of the more complex actual datasets in use by our astrophysicist collaborators. Given the apparent complexity of automated volume rendering, we are now working to provide basic orthoslice and heightmap automation, even if only as a proof-of-concept, for use by the astrophysicists. We will report on the results of this effort in D8.10.

9.2.3 Image services and remote steering of visualizations

In order to give our users "live" access to the the grid-enabled visualization tools, we investigated in cooperation with the DFN CoDisp project the possibility of remote steering of the visualization software (in our case, Amira), and we have implemented a live video service based on the standard real-time protocols RTP and RTSP.

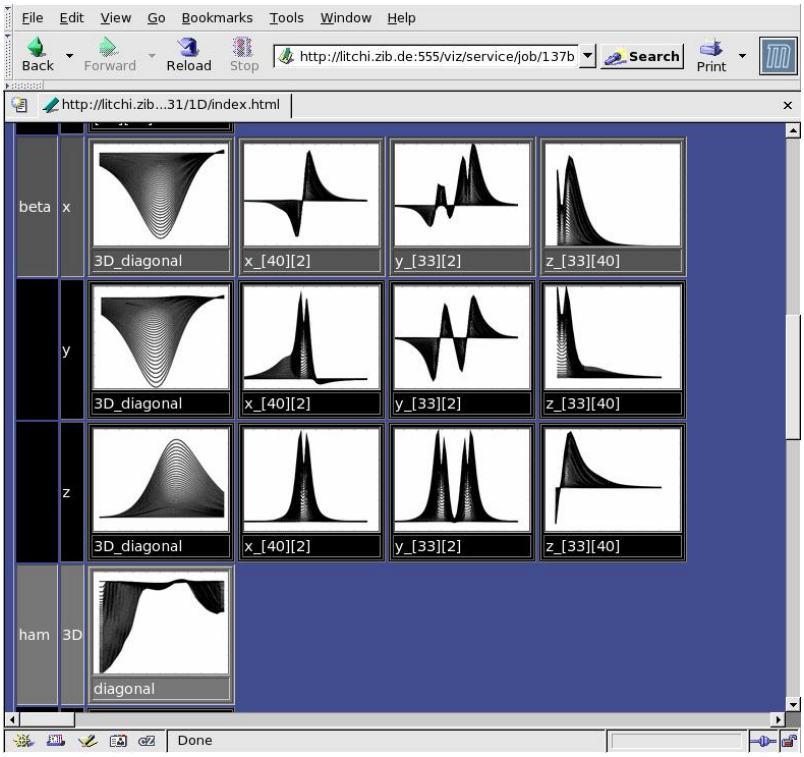

Figure 3: Visualization report: 1D graphics

We used open source software for video encoding (FFMPEG : http://ffmpeg.sourceforge.net/) and for the real-time video transmission (LIVE.COM : http://www.live.com/liveMedia/) to implement a first prototype for this kind of service in Amira.

As the live video streaming is limited in producing high-quality images (constraints given by the CPU power for video encoding/decoding and by the network bandwidth and latency for the video transmission), we have also implemented a prototype service that produces high-quality static snapshots of the visualization session.

Initially we used Amira's TCL interface to implement these services. However, we switched our implementation to use the SOAP protocol because of the limitations in handling error codes and the difficulty of transferring large data structures that we encountered when using TCL.

We continue to develop these services, and will report on ther final status (including detailed interface documentation and technical details for the video parameters) in our final deliverable D8.10.

9.3 Access to visualization tools via grid-enabled mobile devices

ZIB, the institution with primary responsibility for WP8, also has a 12 person-month commitment to WP12 (mobile devices). We are on-target for meeting this time commitment, and have substantial results to report. Given that WP12's deliverables list does not include a direct analog to this report of results, we briefly summarize these results here. As with other sections of this document, we will include more detailed discussion of our mobile device activities in our

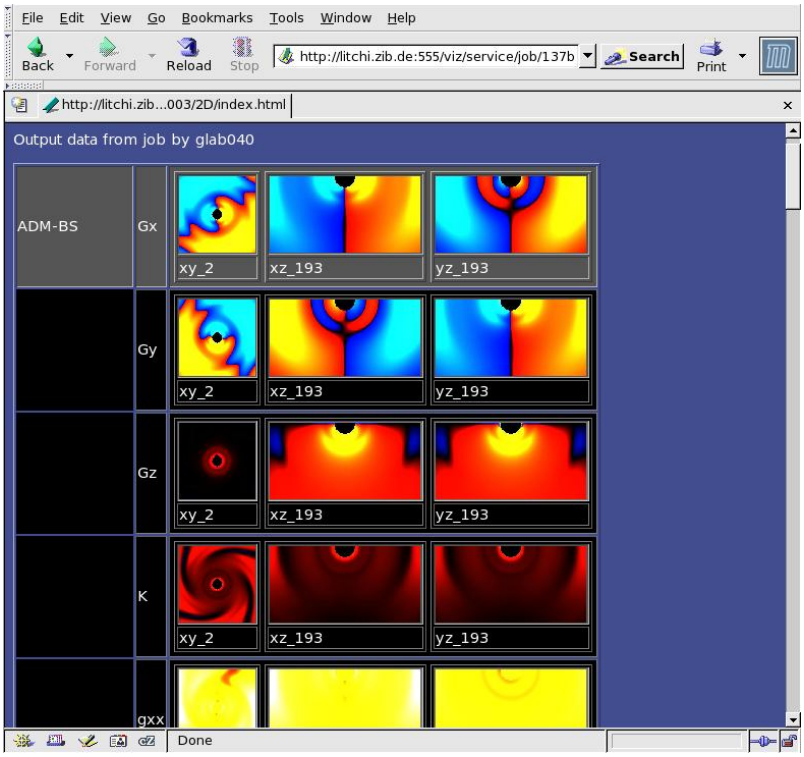

Figure 4: Visualization report: 2D graphics

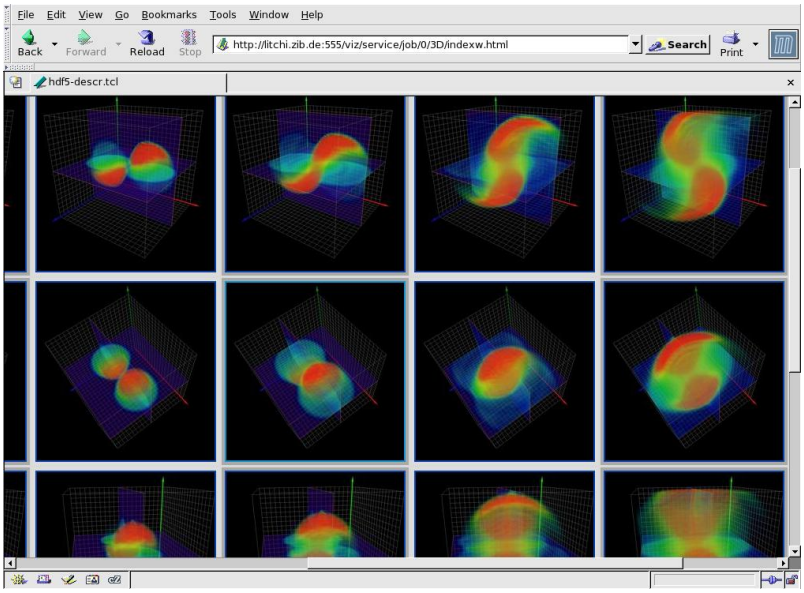

Figure 5: Visualization report: 3D graphics

final deliverable report, D8.10.

9.3.1 Visualization via mobile graphical interfaces

Our base expectation toward WP12 was to support integration of visualization capabilities with WP12's efforts involving grid access from PDA and cell phone/"handy" devices. We collaborated with the core WP12 team toward this end, and produced a grid service which is in active use on the GridLab testbed.

We will document the API in D8.10. In short, the service takes a source URL specifying a remote image as input, as well as a series of transformation parameters; and returns a URL specifying an image with the resulting transformations. A typical use is downsampling a large JPEG (e.g., an image from our visualization service), and transforming this to a small PNG file for display on a cell phone. Another typical use case is supporting interpolated zooming into such images, again for PDA- and cell phone-based image browsing.

This service is located in the GridLab CVS source repository at `wp-8/services/mobile_viz/` and is currently active on machine `litchi.zib.de` on port 17996. The service is in active use by WP12.

9.3.2 Visualization via mobile "tangible interfaces"

Another major outgrowth of ZIB's WP12 mobile efforts lay in the development of new "tangible interface" hardware and supporting software, which we have called "visualization artifacts" (or "viz artifacts"). This effort was first introduced in WP8's fifth quarterly report (submitted April 16, 2003). The motivation for these interfaces was again driven by our astrophysicist collaborators, and has been described in §3.2.3.

These use cases illustrates functionality in the context of immersive and collaborative environments (e.g., virtual reality and video conferencing). Our astrophysicist colleagues have made major investments into both virtual reality and video conferencing environments, and are convinced of the potential values of these technologies for their research.

VR environments typically do not have keyboards or mice, and 3D menus (the state-of-the-art alternative) have proved of limited success. Similarly, in colocated collaboration contexts, it is difficult for multiple users to share a keyboard or mouse. Consequently, in immersive and collaborative environments, there has been an absence of tools for basic tasks such as loading and saving data, starting applications, performing parameter studies, and managing videoconference session. In the absence of these capabilities, our astrophysicist collaborators found it difficult to use visualization tools, with or without our grid extensions, in their VR and videoconferencing environments.

Toward this, we have physically embodied key data and operations in the form of physical cards, wheels, blocks, and pads, in a kind of interface known as a "tangible interface." This interface provides physical control over the astrophysicist's visualization environment; in our case, Amira. These tools should significantly improve the usability of visualization software within immersive and collaborative environments.

We have successfully developed a set of prototype hardware (supported with external funds) and

Figure 6: Visualization artifacts: overview

software. Several illustrations appear in Figures 6 and 7. These devices operate over the network using embedded Ethernet links, and have been designed to integrate securely with GridLab services. We have used this prototype hardware both in collaboration with our astrophysicist colleagues, as well as with surgeons, who also have expressed interest in this approach. Early publications include [4] and [5].

9.4 Visualization of grid infrastructure

While the GridLab's portal provides an interface mechanism for interacting with high-level abstractions like "jobs," it is also desirable to have capabilities for directly representing the function and intercommunication of GridLab's underlying infrastructure. This is valuable in communications with individuals outside of the GridLab project, to help them understand the operation of GridLab infrastructure. It is also valuable within the GridLab project, as developers attempt to understand and debug the functionality of their software.

Toward this, we are developing a series of live, animated diagrams, which can visually display message exchange between different GridLab services, either live (as they happen) or as a playback of recorded events. These will be accessed as Java "WebStart" programs, likely by way of GridLab's portal. Basic functionality is now complete, and integration is now underway.

10 Unaddressed use cases and requirements

As discussed above, WP8 has addressed and satisfied the large majority of its use cases and requirements. As seen in Tables 1 and 4.1, there is one use case (DMS7) and two functional

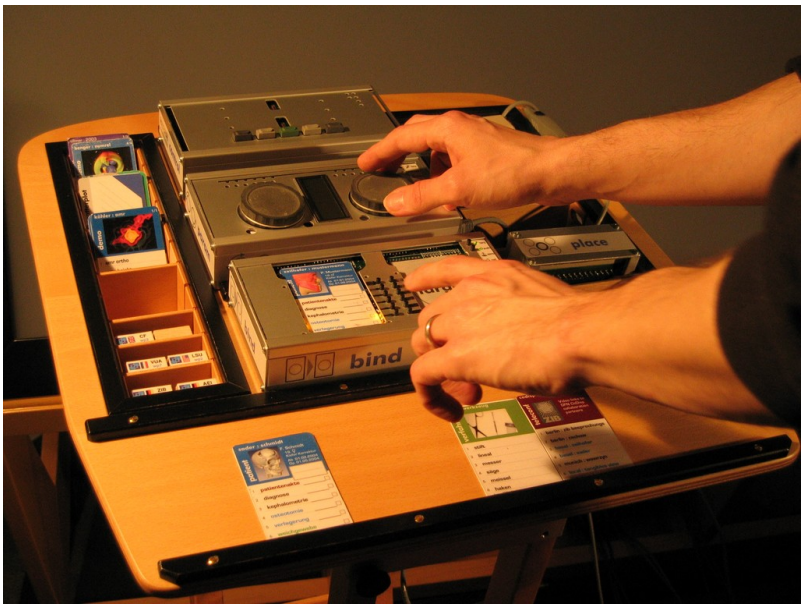

Figure 7: Visualization artifacts: closeup

requirements (FR4 and FR12) which were not addressed.

DMS7 discusses the "recombination of parted data sets/files." This corresponds closely with FR4, "recombination of distributed/parallel written data." In practice, these functionalities were not required by our practical use cases or demanded by our end-user collaborators, and thus were deprecated in favor of more critical functionalities.

Finally, as motivated in the General Remarks section, FR12 (secure data transport) is handled by our use of the GSI protocol. For this reason, it is not mentioned in our evaluation of the individual services.

References

[1] H.-C. Hege, A. Hutanu, R. Kähler, A. Merzky, T. Radke, E. Seidel, and B. Ullmer. Progressive retrieval and hierarchical visualization of large remote data. In *Proc. Workshop on Adaptive Grid Middleware*, pages 60–72, September 2003.

[2] S. Prohaska, A. Hutanu, R.Kähler, and H.-C. Hege. Interactive exploration of large remote micro-ct scans. IEEE Visualization '04 (accepted).

[3] T. Schütt, A. Merzky, A. Hutanu, and F. Schintke. Remote partial file access using compact pattern descriptions. In *IEEE/ACM Intl. Symp. on Cluster Computing and the Grid - CCGrid2004*. IEEE Computer Society, April 2004.

[4] B. Ullmer, A. Hutanu, W. Benger, and H.-C. Hege. Emerging tangible interfaces for facilitating collaborative immersive visualizations (extended abstract). NSF Lake Tahoe Workshop on Collaborative Virtual Reality and Visualization, October 2003.

[5] B. Ullmer, S. Zachow, and H.-C. Hege. Tangible interfaces for facilitating collaborative medical visualizations (abstract). MMVR13 (Medicine Meets Virtual Reality), submitted.