UNICORE Summit 2013

Proceedings, 18th June 2013 | Leipzig, Germany

Valentina Huber, Ralph Müller-Pfefferkorn, Mathilde Romberg (Editors)

JÜLICH
FORSCHUNGSZENTRUM

Mitglied der Helmholtz-Gemeinschaft

Schriften des Forschungszentrums Jülich

IAS Series Volume 21

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

UNICORE Summit 2013

Proceedings, 18th June 2013 | Leipzig, Germany

Valentina Huber, Ralph Müller-Pfefferkorn, Mathilde Romberg
(Editors)

Schriften des Forschungszentrums Jülich

IAS Series Volume 21

ISSN 1868-8489 ISBN 978-3-89336-910-2

Bibliographic information published by the Deutsche Nationalbibliothek.
The Deutsche Nationalbibliothek lists this publication in the Deutsche
Nationalbibliografie; detailed bibliographic data are available in the
Internet at http://dnb.d-nb.de.

Publisher and Distributor:	Forschungszentrum Jülich GmbH Zentralbibliothek 52425 Jülich Phone +49 (0) 24 61 61-53 68 · Fax +49 (0) 24 61 61-61 03 e-mail: zb-publikation@fz-juelich.de Internet: http://www.fz-juelich.de/zb
Cover Design:	Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH
Printer:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2013

Schriften des Forschungszentrums Jülich
IAS Series Volume 21

ISSN 1868-8489
ISBN 978-3-89336-910-2

The complete volume is freely available on the Internet on the Jülicher Open Access Server (JUWEL) at
http://www.fz-juelich.de/zb/juwel

Persistent Identifier: urn:nbn:de:0001-2013102109
Resolving URL: http://www.persistent-identifier.de/?link=610

Preface

When the foundations of UNICORE were laid in the late 90's of the 20th century the "ancestors" intended to create a uniform interface to computing resources for a (small) number of computing centres. Today - in the era of eSciences and large distributed eInfrastructures - UNICORE has become one of the most innovative and major middlewares in Grid Computing serving users around the world.

The UNICORE Summit is a unique event for users, developers, researchers, administrators and service providers to meet. They hear the latest news, share their experiences, present recent and intended developments, get new ideas for projects, find partners and discuss the future of UNICORE. Since the first Summit in 2005, the organisers have received and reviewed a significant amount of distinguished contributions.

This year, it has been held as a satellite event at the ISC Conference in Leipzig, Germany, on 18 June 2013. About 30 researchers from Germany, Italy, Switzerland, the Netherlands, Poland, France, Russia, Belarus, and the United States participated in the conference.

The event was opened with the keynote speech "Quo vadis EGI Clouds?" delivered by Michel Drescher, the EGI Technical Manager, who presented the EGI Federated Cloud Infrastructure. His talk was well-received by the attendees and provided a lot of discussions about UNICORE integration into cloud infrastructures.

Recent developments in integration of applications from community projects, interoperability use cases, security aspects, virtualization techniques, performance evaluation, experiences from end users and administrators, data management, the UNICORE web portal as well as new ideas and concepts and related topics were highlighted in the various talks and demonstrations. Those selected and presented topics guaranteed lively discussions about the state-of-the art and the future of UNICORE, Grids, and distributed computing in general. The slides to the presentations can be found on the web at `http://www.unicore.eu/summit/2013/schedule.php`

We would like to thank all contributors for their presentations and papers as well as the organisers of the UNICORE Summit 2013 in Leipzig for their excellent work. Our deepest thanks go to the program committee members and external reviwers for their hard work in reviewing papers. We are sure every participant will keep this wonderful event in mind. More information about the UNICORE summit series can be found at `http://www.unicore.eu/summit/`.

We are looking forward to the next UNICORE Summit 2014!

October 2013

Valentina Huber
Ralph Müller-Pfefferkorn
Mathilde Romberg

Program Committee

Valentina Huber (*Forschungszentrum Jülich*, Germany)

Ralph Müller-Pfefferkorn (*Dresden University of Technology*, Germany)

Mathilde Romberg (*Forschungszentrum Jülich*, Germany)

Piotr Bała (*ICM Warsaw University*, Polen)

Giuseppe Fiameni (*CINECA*, Italy)

Daniel Mallmann (*Forschungszentrum Jülich*, Germany)

External Reviewers

Mariya Petrova (*Forschungszentrum Jülich*, Germany)

Bernd Schuller (*Forschungszentrum Jülich*, Germany)

Contents

Data-oriented Processing in UNICORE

Bernd Schuller[1], **Richard Grunzke**[2], **and Andre Giesler**[1]

[1] Jülich Supercomputing Centre,
Research Centre Jülich, 52425 Jülich, Germany
E-mail: {b.schuller, a.giesler}@fz-juelich.de
[2] Technische Universität Dresden, Germany
E-mail: richard.grunzke@tu-dresden.de

In order to support the ongoing shift from traditional batch-job oriented computing towards a stronger focus on efficient processing of "big" scientific data sets, we plan to extend the UNICORE middleware with a new data-oriented processing feature. This feature allows to associate processing rules with a storage, where data processing is triggered automatically without explicit user activity. Compared to the similarly rule-driven data management system iRods, the rules are completely controlled by the end user, and all computations are done on account of the user by virtue of UNICORE's security system. Furthermore, the system seamlessly integrates with the existing backend storage. This paper describes the initial design, current implementation status and some first results. The implementation is still in progress, with the first release scheduled for late Autumn 2013 as part of the UNICORE 7 major release.

1 Introduction

More and more, users have data-driven problems, where the main issue is not the computational requirements of the application, but the sheer amount of data, be it in terms of overall volume or number of files. Traditional batch jobs and also Grid middleware systems are ill-equipped to handle these data requirements.

Motivated by the requirements of user communities in the German LSDMA initiative[1], we are looking to support several highly interesting use cases with UNICORE. One example is a high throughput microscopy biology use case[2], where parts of cells are highlighted to examine cellular processes. During different stages of these processes, images are created by automatic microscopes. The processing of these images is controlled by an analysis software. The data-oriented processing has the potential to greatly simplify the job management in the mentioned software. Load on the client caused by job submissions is avoided and the software can be simplified by shifting job management to the UNICORE server. In short, the data-oriented processing has the potential to greatly reduce complexity and overhead in job management and to make the design of the analysis software and its overall usage much more convenient.

A second example is an ongoing campaign at Forschungszentrum Jülich to manage and process a large set of brain scans, which will require not only a large amount of storage space in the multi-petabyte range, but will also involve a complex processing workflow to generate useful information from the raw image data. The processing workflows involve multiple computational steps including checks of intermediary results by a scientist. As algorithms are developed and improved, for example by the availability of new software versions, existing data may be recalculated. The long term goal of this campaign is to develop a 3D brain atlas.

Tackling these use cases with UNICORE will most probably require new features and even novel approaches to achieve the required levels of throughput, scalability as well as reliability. UNICORE is built around the traditional model of HPC batch job processing, where the user sets up a job description, defines input and output data, selects a site and finally submits the job. This model is also the foundation for UNICORE's workflow system. When high-throughput processing is required, this model can be too limited, since creating and managing single Grid jobs incurs a fairly high load on both client and server. Several approaches have been pursued to introduce better support for high throughput data processing into UNICORE. For example, the space-based approach described in[3] may provide some improvement. This approach introduced a new component, the space, which acts as a central job queue from which the execution sites can pull jobs for execution. In this way, the problem of choosing a compute resource is offloaded from the client to the server, resulting in significant speedup. However the space-based approach still requires an explicit action by the client, which has to create and submit a job description to the central queue.

To complement the existing processing models in UNICORE we propose a new feature called "data-oriented processing", where data storages play a central role. Processing rules describe actions that are triggered in certain cases. For example, the creation of new files could trigger job submission to the site-local batch system or to the Grid. We will describe the current state of implementation, as well as look at some examples, initial experiences and some first performance data.

2 Related Work

The rule-oriented approach described in this paper is similar to the iRODS system[4]. iRODS has a very elaborate rule engine and server-side rule processing logic. User-defined rules can be executed with the `irule` command, that can use any of the so-called microservices built into the iRODS server. An administrator can additionally use the global rule file to setup system-level tasks such as replication or metadata extraction. In contrast to iRODS, our approach will work with all types of data stores supported by UNICORE. Specifically, the data can be written also by external tools, not under UNICORE control. Additionally, our approach allows to trigger Grid-level actions such as submissions of Grid workflows and access to remote data. Security and user access control is built-in from the start.

3 Design and Implementation

The basic design of the data-oriented processing feature is rather simple and is depicted in Figure 1. A *rule file* resides in the the base directory of a UNICORE storage. In case of an appropriate external *triggering event*, the UNICORE server will evaluate the rules and create so-called *actions*. These will be executed by the UNICORE server, potentially creating new data.

3.1 Triggering Events

The first design decision concerns the type of triggering events. As the system should also work in case files are written by external tools, we chose to implement a periodic directory

scan which checks for new files. To ensure scalability, the directory scan is controlled by the user, who can include or exclude certain directories and control the scan interval as well as the scan depth.

Additionally, there is the possibility for triggering manually (i. e. via an explicit client call).

3.2 Rules and Actions

The rule file (named `.UNICORE_Rules`) is used to control the directory scan discussed in the previous section as well as define one or more rules which will be evaluated.
The rule file syntax and the full semantics of rules are still under development. We chose a JSON notation for rules, which is simple enough and allows to re-use the job definition syntax from the UNICORE commandline client (UCC)[5], which will already be familiar to many users.

A rule definition contains

- the rule name
- matching conditions: which file name patterns should match
- action definitions: what should be done for each match

The action section defines the actual processing. There are different action types, which take different options:

- short computations on the local cluster's login node
- batch jobs on the local cluster
- automated metadata extraction and indexing using the UNICORE metadata system
- UNICORE jobs submitted to other sites
- UNICORE workflows

In principle it is possible to trigger any other activity that can be executed via UNICORE, for example using the XML spaces component or initiating automated file copy processes for replication.

Currently the first two of these action types are implemented. Submitting UNICORE jobs and workflows will require some more work in providing the necessary client features (such as efficient resource discovery and workflow support) in UNICORE/X.

Rule definitions may contain context variables such as current directory, file name, etc, which will be filled in by the UNICORE/X server.

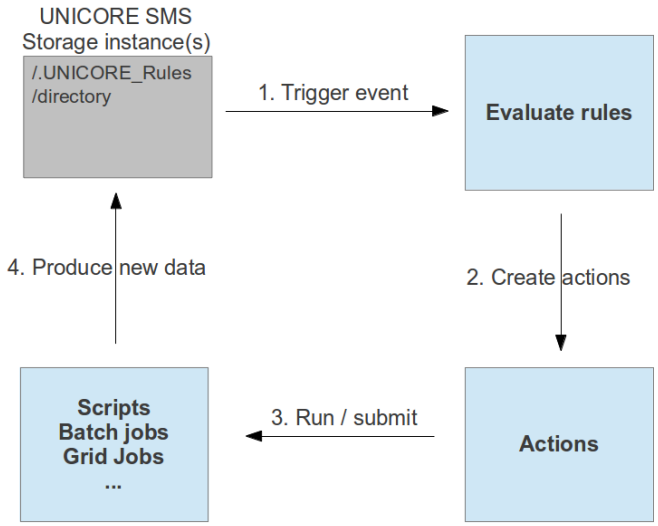


Figure 1: Basic architecture of the data oriented processing

4 Some Examples

Some initial tests of the new feature have been performed. As the rule syntax is still under development, the examples in this section are subject to change.
As a first use case, let's consider computation of an MD5 hash of files having the extension "pdf" on the cluster login node.
The rule would look like this.

```
{
  Name: computeMD5Sum,
  Match: ".*\\.pdf",
  Action:
  {
     Type: LOCAL,
     Command:  "/usr/bin/md5sum $\{UC_FILE_NAME\}",
     Outcome: "$\{UC_BASE_DIR\}/out",
     Stdout:  "$\{UC_FILE_NAME\}.md5",
  },
 },
```

The `match` clause is a Java regular expression used to match file names. The resulting shell script is executed via the UNICORE Target System Interface (TSI). The user can control the output location.

Similarly, using a batch job as action would be expressed as

```
{
  Name: computeMD5viaBatch,
  Match: ".*\\.pdf",
  Action:
  {
     Type: BATCH,
     Job:  {
       Executable: "/usr/bin/md5sum",
       Arguments: [ "$\{UC_FILE_PATH\}" ],
       Stdout: "$\{UC_FILE_NAME\}.md5",
     }
  },
},
```

The simple features described so far can already be used to great effect and allow to submit batch jobs with a high submission rate. While testing we found that the UNICORE XNJS limits the job submission rate to several jobs per second.

From the other action types listed above, we will certainly implement the metadata extraction in time for the first release. The Grid-level actions require some more effort. We would like to involve end-users, too, to see what their data-driven workflows will look like.

5 Summary and Outlook

The new feature described here has a lot of potential to enable data-driven use cases such as the high-volume brain scan processing described in the introduction. Compared to the similar iRODS solution (which is of course much more advanced at the current time) we see a major advantage in the openness of the UNICORE system. In a way, iRODS is yet another silo solution, which takes some control away from the end user. Our solution has the user still in full control of his data and all the processing that is done.

The upcoming UNICORE 7 release will contain this feature at least in an initial state, so users can start using it and give valuable feedback as to which direction should be chosen for further development.

Acknowledgements

The authors wish to thank Jedrzey Rybicki for fruitful discussions. Part of this work was supported by the LSDMA initiative of the Helmholtz foundation.

References

1. "Large scale data management and analysis (LSDMA)." http://www.helmholtz-lsdma.de/.
2. U. M. R. Grunzke, R. Müller-Pfefferkorn and M. Müller, "Advancing cutting-edge biological research with a high-throughput unicore workflow," in *Proceedings of the 2011 UNICORE Summit, IAS Series Volume 9*, pp. 35–43, 2011.
3. B. Schuller and M. Schumacher, "Space-based approach to high-throughput computations in unicore 6 grids," in *Euro-Par 2008 Workshops - Parallel Processing, VHPC 2008, UNICORE 2008, HPPC 2008, SGS 2008, PROPER 2008, ROIA 2008, and DPA 2008, Las Palmas de Gran Canaria, Spain, August 25-26, 2008. - Revised Selected Papers / ed.: E. Cesar, M. Alexander, A. Streit, J.L. Träff, C. Cerin, A. Knüpfer, D. Kranzlmüller, S. Jha. - Springer, Berlin, Heidelberg, 2009, Lecture Notes in Computer Science Vol. 5415, 978-3-642-00954-9. - S. 75 - 83*, 2009.
4. "irods: Data grids, digital libraries, persistent archives, and real-time data systems." http://www.irods.org.
5. "Ucc job definition syntax." http://unicore-dev.zam.kfa-juelich.de/documentation/ucc-6.6.0/ucc-manual.html#ucc_jobdescription.

A Data Storage Solution Using the PL-Grid UNICORE Infrastructure

Rafał Kluszczyński[1], **Marcelina Borcz**[1], **Grzegorz Marczak**[1,2], **Marcin Stolarek**[1], **Piotr Bała**[1,2]

[1] Interdisciplinary Center for Mathematical and Computational Modelling, University of Warsaw, Warsaw, Poland

[2] Faculty of Mathematics and Computer Science Nicolaus Copernicus University, Toruń, Poland
E-mail: {klusi, marbor, lielow, mstol, bala}@icm.edu.pl

Storage is similarly important as compute resources. Moreover, recent developments in cloud technology allow users to access distributed storage in an easy and efficient way. The option is especially important for the processing of biomolecular and genetic data which is necessary for diagnostics and research. Recently we have introduced a UNICORE based solution which allows for the uniform access to the experimental data, their handling and analysis in the distributed environment which significantly reduces processing time. In this paper we present details of the solution including mechanisms for secure data backups using external resources. We also present performance data for the data transfer using UFTP protocol which has been used to speed up data upload from the users workstation to the grid. The results have been obtained using the example sequence data consisting of a large number of files. The transfer setup has been optimized to handle it.

1 Introduction

The experience with the grid deployment shows that storage is similarly important as compute resources. Recent developments in cloud technology allow users to access distributed storage as folders on the desktop or through simple web application allowing for easy (often drag and drop) upload and download of files. The users would like to use similar technologies to access grid resources. Unfortunately, existing grid middlewares have been designed with the focus on efficient CPU utilization rather than data storage.

With the increasing number of data sources, the data storage and processing becomes an important issue. Almost all processes we are involved in, are now digitalized and produce large amount of data. In particular, the modern life sciences research widely uses large data sets both stored in the existing databases or obtained in high throughput experiments. The amount of data is significant and is rapidly growing as experimental equipment becomes more affordable. The data storage is especially important for the processing of biomolecular and genetic data. There is a growing demand to process experimental data effectively and provide results in the time required for the diagnostics and medical treatment which is within hours rather than days or weeks. As a result we observe ithe ncreasing demand for the disk space and computer power to store and process data. Emerging technologies such as Next Generation Sequencing (NGS) generate large amounts of data which has to be stored and processed as soon as possible. The sequencing experiment is usually controlled by the laptop or simple workstation but the amount of data is too big to be stored there for a longer time. The data processing using desktop or even more powerful single computers available in the laboratory is possible, but takes too long to be practical.

Up to now, the research teams were building their own infrastructures, sometimes quite extensive ones, to store and process data. In addition they had to provide dedicated IT staff to operate and maintain it. We observe that in the last few years high throughput systems are widely spread and are used by smaller groups or even become diagnostics tools at hospitals. By building and maintaining a dedicated computer infrastructure capable to process data in the required time is very costly and simply not feasible.

A good solution is to acquire external resources for storage and computations. The obvious choice are grids and clouds since they can provide the required infrastructure. Because security of the data is a concern, grid solutions with the strong security model based on the X509 certificates are the natural choice.

Recently we have introduced a UNICORE based solution which allows for the uniform access to the experimental data, their handling and analysis in the distributed environment which significantly reduces processing time[3]. The key component of the developed system is data storage and management using UNICORE[4]. Using UCC the user is able to perform automatic transfer of sequence data from the experimental system to the distributed storage. The proposed solution allows to use grid resources for processing genetic data efficiently which is crucial for Next Generation Sequencing.

In this paper we present details of the data storage solution including mechanisms for secure data backups using external resources. The results have been obtained for the example sequence data consisting of a large number of files. The transfer setup has been optimized to handle it. We also present performance data for the data transfer using the UFTP protocol which has been used to speed up data upload from the users workstation to the grid. The paper is organized as follows: first we present an overview of the UNICORE infrastructure used, than we describe details of the data related components and finally we present results of the performance measures and conclusions.

2 UNICORE Deployment in the PL-Grid Infrastructure

The Polish National Grid (PL-Grid) Infrastructure consists of 5 HPC centers which act as individual sites providing computational resources. The main aim of the PL-Grid is to build a sustainable grid infrastructure for the scientific community in Poland[1]. The hardware resources can be accessed using different middlewares such as gLite, QCG and UNICORE.

The schematic view of the UNICORE deployment is presented in Figure 1 and is described in details elsewhere[2].

Here we will summarize the key elements of the UNICORE installation. Most of the services are installed at the main site located at ICM. The execution services are installed in all sites providing resources including ICM. The UVOS service is populated with the user data received from the PL-Grid Portal which contains information about users, their privileges and certificates. This data is synchronized with the UVOS and is used to authorize and authenticate users.

The users willing to use PL-Grid infrastructure have to register in the PL-Grid portal. The registration process is common for all services and provides some mechanism to check if the user is allowed to use them. In particular users are verified in the external databases of the Polish R&D community or are asked to provide contacts to their advisor or collaborator. In addition, the PL-Grid portal provides an interface to obtain self-generated certificates

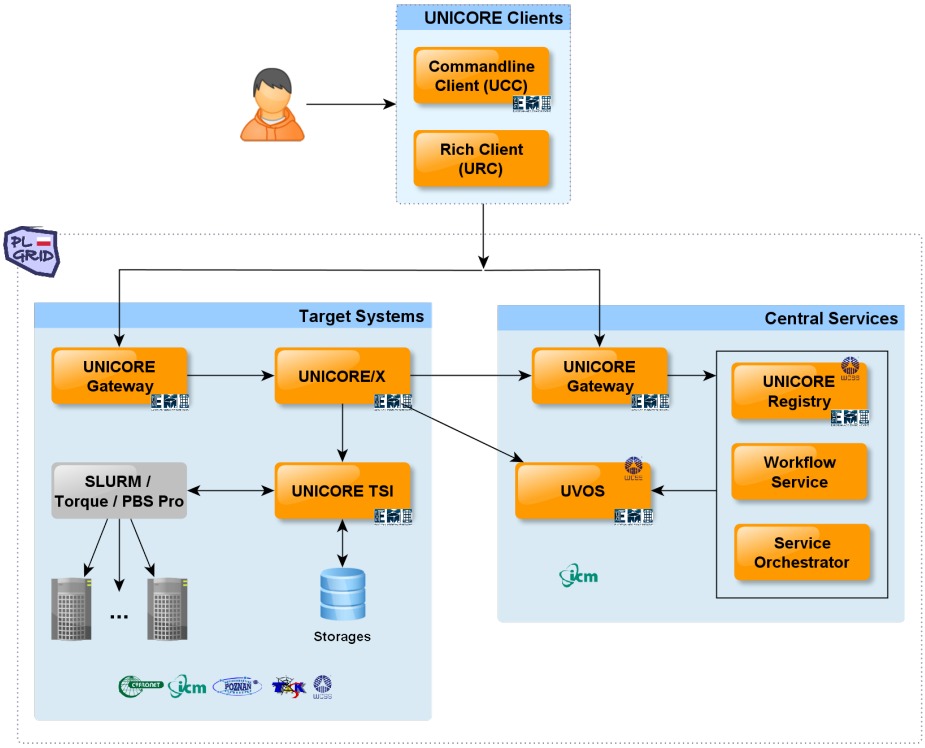


Figure 1: Schematic view of the UNICORE deployment in the PL-Grid.

issued by the Simple CA. This solution allows to hide from the user less intuitive parts of the certificate issuing process and facilitates the placement of additional information in the form of groups or attributes in the UVOS. These informations are necessary to grant users with the proper privileges.

After successful registration the user can apply for the different services including UNICORE access to the resources or other domain specific services like processing or storing of genetic data.

The utilization of the PL-Grid resources is monitored based on the CPU time allocations called computing grants. Every new user obtains an individual allocation which is limited (currently 1000 CPU hours). For larger allocations one has to apply providing a description of the proposed use of the resources. The proposals are evaluated and then an allocation is granted. The allocation can be shared by the group of coworkers organized in the groups.

The groups are maintained within the PL-Grid portal. The grant leader (Scientific Coordinator) can create a group and connect it to the particular resource allocation. The members of the group will share the particular resource allocation and will have the possibility to share disk space and files. The information on the groups and group members is synchronized with the PL-Grid LDAP and with the UVOS.

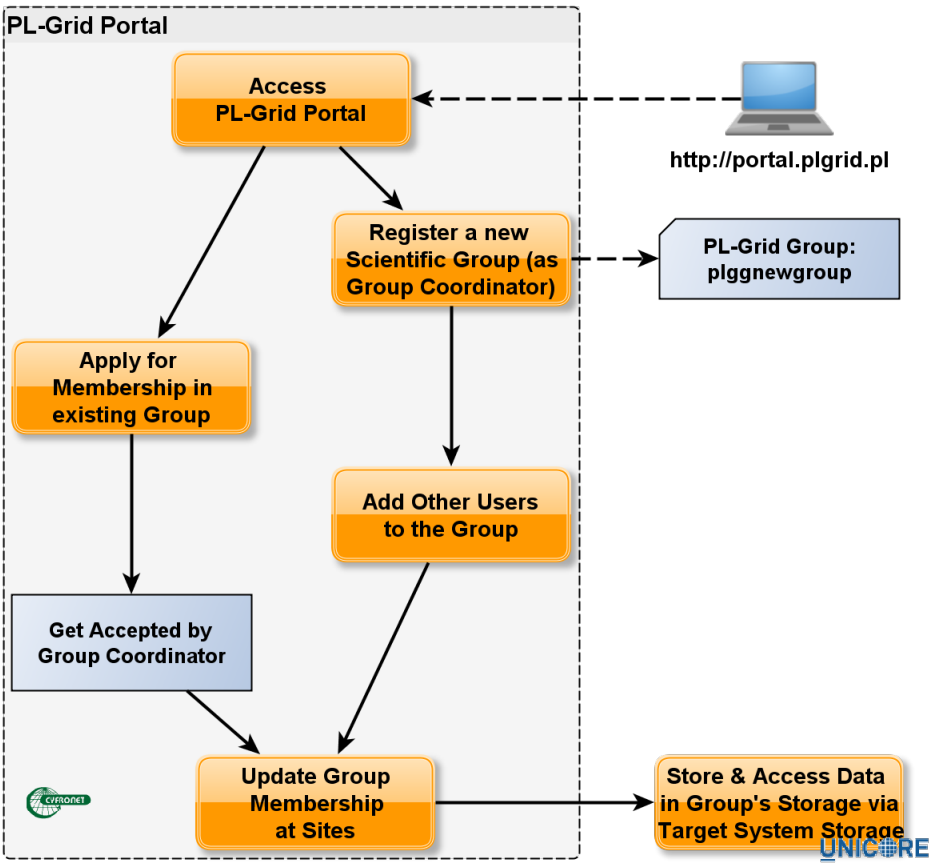


Figure 2: Schematic view of the processing of users and groups in the PL-Grid.

The Scientific Coordinator can add other users to the group and has rights to remove them. Other users can apply for the group membership through the PL-Grid portal. Once the application is approved by the group leader, the membership is granted. The schematic view of this process is presented in the Figure 2.

3 Data Storage in the PL-Grid UNICORE Infrastructure

For the data, UNICORE provides access to the individual, global and group wide storage by a simple configuration of the Target System. The user can use individual storage which is in most cases his home directory on the target systems. He can also use global storage which is a storage service for storing files shared between workflow tasks. All of them can be accessed using one of UNICORE standard protocols: BFT and multistreaming UFTP.

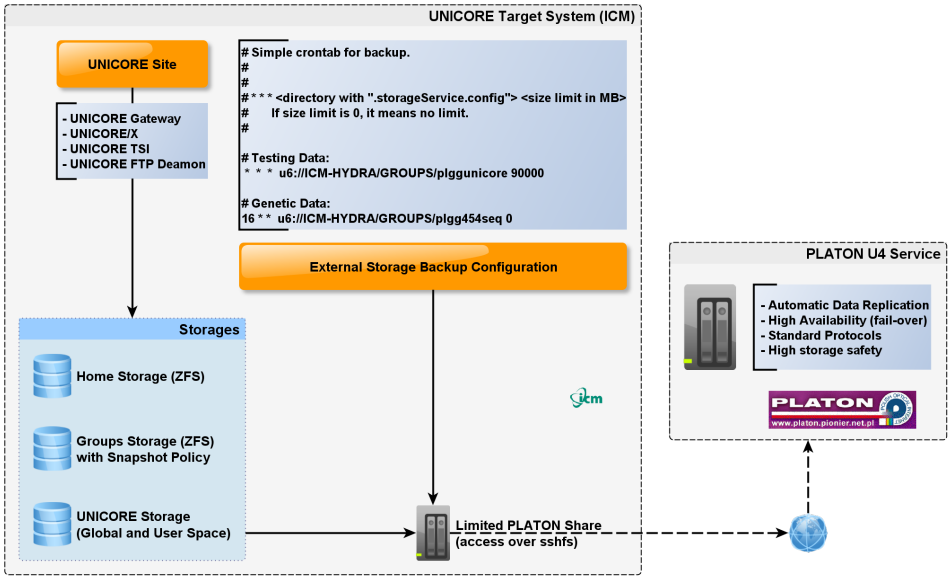


Figure 3: Schematic view of the backup of the genetic data using Platon U4 service.

In the PL-Grid infrastructure we have provided group storage which is similar to the global one but access is restricted only to the group members. Default access privileges (unix `umask`) are used to ensure that newly created folders and files are accessible by default to all of the group members. The group storage can be used to share files between group members and to store them for a longer time as it was frequently requested by the users.

Once group storage became widely used, the data persistence had to be ensured. Especially the need for fast and reliable backups of large amounts of data appeared.

The file systems used to built group backup can be periodically stored on the tapes, but this process is relatively slow and complicated. Especially the data recovery is quite challenging and cannot be done by the unexperienced user.

Therefore we have decided to used remote archiving and backup services provided by PIONIER, the national network of academic research. The PIONIER backup service (Platon U4) is addressed to the academic community, including universities, research and development units and hospitals run by medical schools and therefore is a good solution for protecting UNICORE group storage data. The service increases data protection in real time and can be used to increase the reliability.

Finally, ICM ends up with multi-level backup of data stored on official PL-Grid group storage. This ensures accessibility to data by user at any time. For group storage the snapshots are run in daily or weekly mode which is accessible by the user. An additional layer is backup with the use of external services (Platon U4) which is configured based on user request and needs. Data stored with those external services are additionally encrypted and at the moment can be accessed by the user only by contacting the administrator.

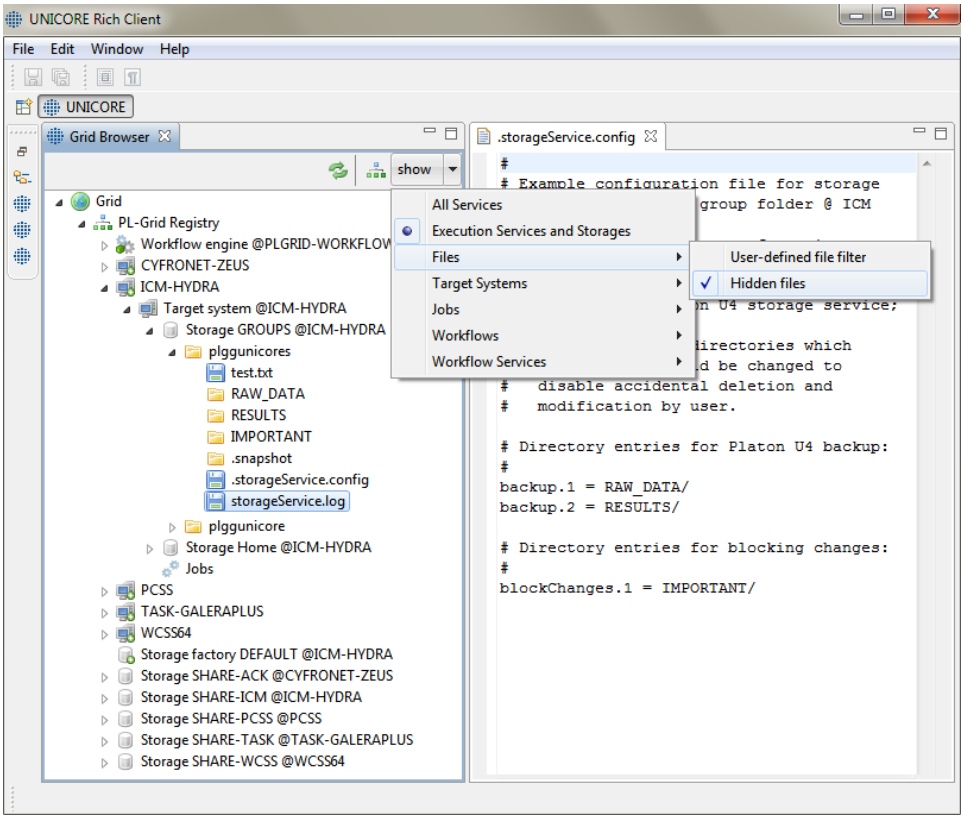


Figure 4: View of the users backup configuration file with the use of URC.

The external backup process is configured based on the users request submitted to the PL-Grid HelpDesk. The system administrators verifies the user and his request for safe storage. During this process the backup parameters are determined. In particular:

- data localization (which can be user's home storage or group storage),
- backup frequency,
- data size limit.

Then the administrator adds an entry to the service configuration file and confirms that the backup is ready.

The user has the possibility to configure which files will have external backup. This is done through a configuration file located in the particular storage. The structure of the file is simple, an example is presented in Figure 4. The backup process can be monitored by the user through a log file located in the same directory. This is especially important, because there are some limitations for the backup process, like:

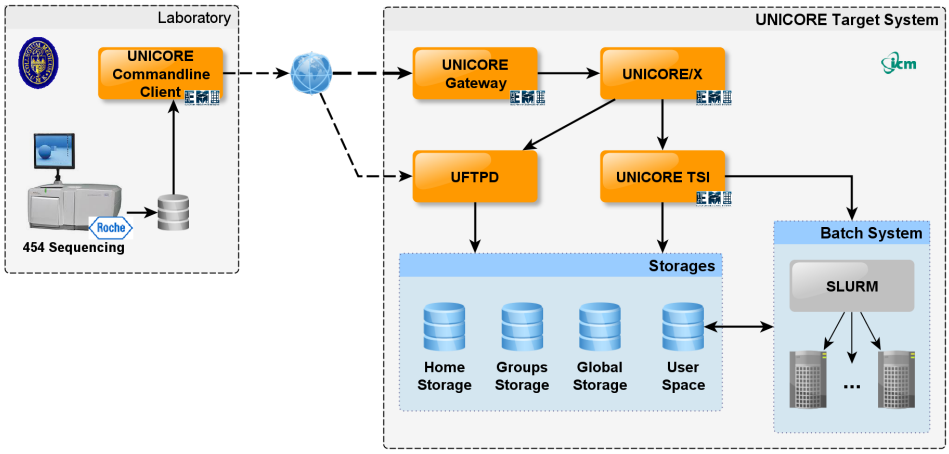


Figure 5: Details of the data solution for processing data from the sequencing experiment.

- users can not use as many or directories as they want, this value is set by the administrator (for example: 10) and the rest of the entries is ignored,
- if data size is greater than the value agreed with administrator, process will be skipped with appropriate warning in the log file,
- all directory or file entries have to be in a location agreed with administrators (directories and files located outside the agreed path will be omitted).

It is worth to mention that both files can be accessed by the UNCORE clients. In particular user can use URC to look at the file content and to modify it.

One should note, that during the process, the backup files are encrypted before they are sent to the remote systems. This ensures that data is stored safely and cannot be accessed by unauthorized persons. The schematic view of the external backup of the UNICORE storages is presented in the Figure 3.

4 Data Storage for Processing Genetic Data

The solution described above has been used to store and process users' data. Answering the needs of groups running genetic research and thus facing problems of storing and processing large data sets we have created a dedicated solution allowing to store and process genetic data on the UNICORE Grid.

Such a solution was also needed because the built-in capacity of scientists instruments can usually only store data up to few experiments. Moreover, the solution should support requirements such as: fast upload for processing, easy access to data, secure transfer between instrument and storage, confidentiality of data stored with external services.

In particular we have integrated the GS FLX Instrument available at the Collegium Medicum, N. Copernicus University with the PL-Grid distributed infrastructure as the data

storage and processing system[3]. Similar work for the Illumina sequencer located at the Warsaw Medical University is in progress and should be finalized soon.

Once the experiment is finished, the data is automatically transferred from the workstation attached to the sequencer to the dedicated group storage. This is done with the help of the UNICORE Commandline Client integrated within the scripts used to run the experiments.

A single sequencing experiment generates ca. 30 GB of data which is organized in 834 files of the size about 33 MB each. This amount of data requires an efficient and reliable transfer mechanism. Therefore for the data transfer between the workstation (located geographically in different cities) and the grid infrastructure the UNICORE File Transfer Protocol (UFTP)[5] is used. The UFTP is also used to copy files between different UNICORE sites. The details of this solution are presented in the Figure 5.

In April and May 2013 we have observed almost 2000 UFTP uploads of files over 30 MB in size. Most of them used 4 streams and were performed using WAN. The average upload transfer was at the rate of 10.7 MB/s. Files with size 10 GB and more have achieved 11.1 MB/s.

One should also note, that during processing of the results of the sequencing experiment the size of the data is still remarkable. The GSRunProcessor processes files of the size of 27 GB, Blast results may have up to 6 GB, other programs generate files of the size of at least a few gigabytes.

5 Performance Issues

The significant size of the data generated by the sequencing experiments requires an optimization of the external backup to reduce processing time. The direct method based on the tar program and OpenSSL for encryption is too slow to perform this task in reasonable time. In particular, it took about 6 to 11 hours to perform an external backup of a directory of the size exceeding 200 GB containing data from the 18 sequencing experiments and it will increase with every new experiment. The time may vary based of the network traffic, thus a faster solution is needed which will not process again the whole subdirectory if only one file has changed. We have also observed that encryption on-the-fly using OpenSSL reduces storing speed by half.

We have investigated other solutions used on top of the Platon U4 service like TrueCrypt[8] or EncFS[9]. The main idea is to mount the Platon share using FUSE[7] and to use this mounted filesystem to store copy of important data.

For that purpose we have tested for three solutions the execution time of storing file of size 1 MB, 10 MB, 100 MB and 1 GB. Every file was stored at least 10 times and finally the average value was estimated.
The first method was the one used already: simple tar program with OpenSSL encryption.

The second method used the TrueCrypt program (version 7.1a). With the use of that command, a volume was created and stored in a mounted Platon share with the following parameters:

- created (normal) volume was of size 2 TB and it was containing a Linux Ext4 file system,

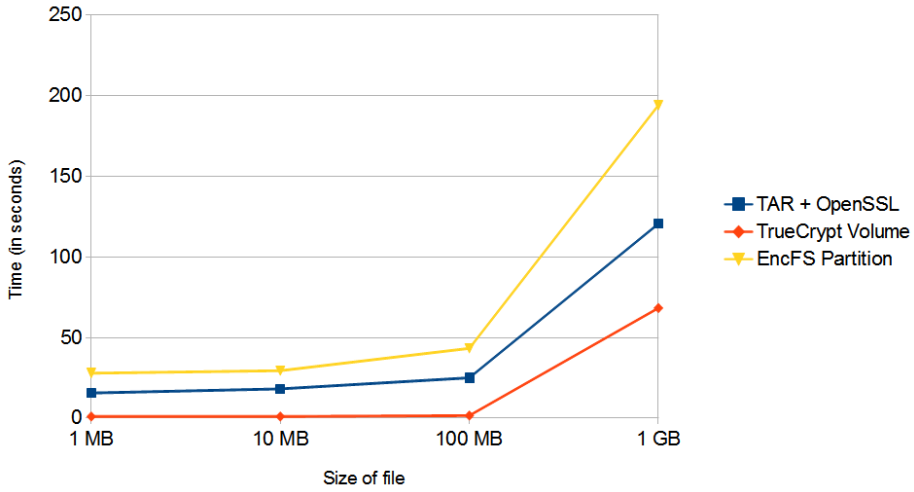


Figure 6: The external backup performance of using different encryption methods.

- AES encryption algorithm and RIPEMD-160 hash algorithm,
- using password without keyfile.

This solution occurred to be very efficient. Moreover it has such advantages that volume size limits the user's data and thus no additional process is needed to limit it.

The last tested solution was EncFS version 1.7.4-1 installed from EPEL repository. This additional storage was prepared using the following default configuration:

- filesystem cipher: "ssl/aes", version 3:0:2,
- filename encoding: "nameio/block", version 3:0:1,
- key size: 192 bits, block size: 1024 bytes,
- each file contains 8 byte header with unique IV data,
- filenames encoded using IV chaining mode.

This solution has the advantage of a separate encryption of individual files. This way the change in one file does not affect all dataset and only the changed file can be updated on external storage.

The comparison of all solutions is presented in Figure 6. It can be seen that the method using TrueCrypt behaves extremely well which is related to the Ext4 file system used on prepared volume. In the case of EncFS the results are a bit surprising: data was stored much slower than simple TAR and OpenSSL solution.

Both methods (TrueCrypt and EncFS) need to be investigated further, especially as there are many encryption parameters (like type of algorithm, key size) to setup. Nevertheless, a solution using an external backup service provides an additional safe and reliable place to store data which PL-Grid users can fully take advantage of.

6 Acknowledgements

This work has been supported by the PL-Grid Plus project and partially by EMI project (RI-261611). The PL-Grid Plus project is co-funded by the European Regional Development Fund as part of the Innovative Economy program.

References

1. J. Kitowski, M. Turała, K. Wiatr, Ł. Dutka Building a National Distributed e-Infrastructure – PL-Grid Scientific and Technical Achievements *(Lecture Notes in Computer Science 7136)* Springer 2012
2. K. Benedyczak, M. Stolarek, R. Rowicki, R. Kluszczyński, M. Borcz, G. Marczak, M. Filocha, P. Bała Seamless Access to the PL-Grid e-Infrastructure Using UNICORE Middleware In: J. Kitowski, M. Turała, K. Wiatr, Ł. Dutka Building a National Distributed e-Infrastructure – PL-Grid Scientific and Technical Achievements *(Lecture Notes in Computer Science 7136)* Springer 2012 pp. 56–72
3. M. Borcz, R.. Kluszczyński, K. Skonieczka, T. Grzybowski, P. Bała Processing the biomedical data on the grid using the UNICORE workflow system. In: I. Caragiannis et al. (Eds.): Euro-Par 2012 Workshops, *Lecture Notes in Computer Science 7640*, Springer, Heidelberg (2013) pp. 263–272.
4. T. Rękawek, P. Bała, K. Benedyczak Distributed Storage Management Service in UNICORE In: M. Romberg, P. Bała, M. Romberg, D. Mallman (Eds.) *UNICORE Summit 2011 Proceedings, 6-7 July 2011 Toruń, Poland* IAS Series vol. 9, pp. 125–134 Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 2011
5. B. Schuller, T. Pohlmann, "UFTP: High-Performance Data Transfer for UNICORE," In Proceedings of UNICORE Summit 2011 (editors: M. Romberg, P. Bala, R. Müller-Pfefferkorn, D.Mallmann), Forschungszentrums Jülich, IAS Series Vol. 9, ISBN 978-3-89336-750-4, pp. 135–142
6. P. Jędrzejczak, M. Kozak, C. Mazurek, T. Parkoła, S. Pietrzak, M. Stroiński, J. Węglarz Long-Term Preservation Services as a Key Element of the Digital Libraries Infrastructure In: *Intelligent Tools for Building a Scientific Information Platform Studies in Computational Intelligence* vol. 467, Springer (2013), pp. 85-95.
7. Filesystem in Userspace (FUSE) website: `http://fuse.sourceforge.net`
8. TrueCrypt website: `http://www.truecrypt.org`
9. EncFS website: `http://www.arg0.net/encfs`

Combining HPC with Data-Intensive Research via UNICORE for Seismological Applications

Michele Carpené[1], **Graziella Ferini**[1],
Alessandro Spinuso[2], **Luca Trani**[2], **Marek Simon**[3]

[1] CINECA, 40033 Casalecchio di Reno, Bologna, Italy
E-mail: {m.carpen, g.ferini}@cineca.it
[2] The Royal Netherlands Meteorological Institute (KNMI), Netherlands
E-mail: {alessandro.spinuso,luca.trani}@knmi.nl
[3] Ludwig-Maximilian-University, Department of Earth and Environmental Sciences, Germany
E-mail: {marek.simon}@geophysick.uni-muenchen.de

Advances in computational seismology require to realize complex geophysical simulations in order to predict consequences of seismic phenomena. Data collected by seismic stations must be analysed and compared with results of simulations in order to gain new information about the Earth's subsoil composition in a fast and efficient manner.

In order to provide high-level tools to run complex simulations as HPC jobs geoscientists need to be supported by a wide distributed scientific infrastructure able to ensure transparent access to resources. Within the VERCE[1] project this goal is addressed exemplifying such challenges and implementing specific use cases, providing end users with software tools and application environments and defining a basic service-oriented architecture.

UNICORE comes in handy providing a software layer capable to run complex workflows and supporting end users with graphical tools and GridBeans to model complex simulation jobs to run on the HPC resources.

In this paper we describe how UNICORE can be used in VERCE to enhance scalability of experiments combining the HPC file-based model with data-processing operations.

The aim of this work is to demonstrate the flexibility of the VERCE architecture, showing possible advantages of our UNICORE-Based solution.

1 Introduction

One of the biggest challenges in modern seismology research consists in building complex and sophisticated computational models that simulate Earth's physical aspects such as surface and mantle characteristics. Simulations of seismic wave propagation as well as seismic tomography are powerful means to gain insight into seismic phenomena in order to increase the knowledge of the Earth and earthquake processes, as well as to study the consequences of catastrophic events.

Since one main goal of seismology communities is to achieve a deep knowledge of Earth's inner structure and composition in order to predict the dynamic of seismic phenomena, seismologists need to gain access to information recorded by instruments, as for example seismographs installed on seismic stations.

Moreover, such data can be compared against data generated in a numerical simulation in order to derive a wave propagation model, representing the real velocity structure of the Earth. The accurate numerical simulations of seismic wave propagation can in fact help us to understand complicated phenomena and determine the characteristics of sources

and material structures. Sophisticated scientific simulation software (i.e. Specfem3D[2], SeisSol[3]) is used by the community for the calculation of wave propagation fields, making use of tetrahedral and hexahedral computational meshes to approximate complex 3D media geometries and realistically approximate geological subsurface properties. The principle at the basis of such simulation software is to represent the medium as a set of interconnected computational elements (e.g. tetrahedra or hexahedra), each adhering to specific boundary conditions, and to resolve the dynamic of the wave equation as an interaction between the elements by a numerical calculation.

Since for realistic 3D applications it is required to solve complex differential equations over very large parameter-sets, such kind of simulations are tractable only when relying on advanced, high-performance computing facilities and the resulting output is both large in number and in storage size. In order to enhance distributed data-access, as well as to make effective use of distributed computational resources, an adequate e-Infrastructure must be provided. The e-Infrastructure should offer availability of storage resources and software tools to orchestrate data mangement and HPC simulations.

2 Background

The EU-funded project VERCE aims to provide a service-oriented architecture delivering software tools and services providing access to HPC/HTC and data-intensive facilities and connecting them to remote data-archives. VERCE represents a major contribution to the e-science environment of the European Plate Observing System EPOS[4]. Progresses achieved within the project can be shared with other initiatives, such as EUDAT[5], currently addressing other issues like data management or authentication. VERCE also provides end users with a platform represented by a set of powerful applications to execute simulations, middleware, workflow tools and graphical interfaces. One major innovation in VERCE consists in combining a streaming model with the classic HPC file-based model in order to provide seamless execution of seismology-related workflows[6]. This is done introducing the Dispel gateway, a component able to parse and enact Dispel[7] workflows. Dispel gateway deals with a number of known resources and represents the coordinator of the whole job execution/data processing loop.

Data-intensive tasks, like misfit calculation and noise-correlation, are performed efficiently by using the so-called processing elements (PEs), a set of atomic, logical unit in the form of Python scripts that perform specific stream analysis tasks. These PEs rely on the ObsPy[8] Python framework and use the Python libraries developed in VERCE. The PEs can be combined and used within Dispel workflows to perform complex analysis on huge amounts of data.

The scope of this paper is to propose an alternative UNICORE-based approach, showing how VERCE software components can be reused within UNICORE in order to exploit compute-intensive and data-intensive functionalities in the UNICORE context. This has been investigated installing the ObsPy Python framework on the IBM GPU-based PLX Cluster in CINECA[9], and re-using a set of VERCE data-intensive Python scripts.

In the next sections we present results of our work. First we present the main use-cases, then we introduce the UNICORE-based architecture, we present the processing elements, technologies adopted, testing and conclusions.

This paper presents a proof-of-concept, the considerations exposed in this work are not intended to be a choice in adopting such technology in VERCE, rather we aim to demonstrate the flexibility of the VERCE architecture. Now we are going to present how the problem of modelling complex geophysical systems is addressed in VERCE and how this happens through a continuous loop of forward and inverse simulation.

3 Forward modelling and inversion

In VERCE one main use case addresses the problem of generating large sets of synthetic data in a series of a number of Forward Modelling runs and to analyse and process the data, determining the robustness of a specific model/solver combination. The forward modelling and inversion use-case consists in calculating complete 3D wave propagation fields, comparing the synthetic data recorded at the position at the stations with real observed seismograms and using the resulting misfit information to iteratively update the Earth model by means of the adjoint inversion technique. Comparisons between synthetic data and real data are done calculating the misfit by a misfit function, that is selected on demand. The generation of synthetic seismograms as well as the inversion process are typically compute intensive tasks, since both require heavy Message Passing Interface (MPI) communication. The misfit calculation instead is typically considered as data-intensive, since it can be split up in a large number of independent processes. The inversion modelling will not be treated in details since it is out of the scope of this paper, moreover this work has to be considered a work-in-progress and many aspects are still under discussion.

3.1 Misfit calculation

The misfit calculation is completely parallel, without MPI communication, it is therefore considered data-intensive. To derive the misfit, the raw data needs to be preprocessed first and while the misfit function itself can range from very simple $\|L\|_2$ to a more complicated time-frequency functional. The whole process will always require multiple FFT's so it can be considered computationally expensive for each waveform. However, the results do not need to be immediately communicated between the processes, so during computation each data stream can be treated independently.

4 Architecture overview

VERCE builds its service-oriented architecture around the Dispel gateway. The Dispel gateway deals with parsing Dispel workflows and expanding them in enactable graphs forwarding parts of these graphs to appropriate resources for proper execution. The execution of simulation codes on HPC machines is coordinated by the Dispel gateway which actually submits jobs on HPC resources using Globus and UNICORE. In particular the job submission is performed through JSAGA[10] APIs. After job execution has completed the output files are serialised before they are moved back to the Dispel gateway for further processing and data-intensive analysis.

The service-oriented architecture we propose here (Figure 1) is instead completely based on UNICORE, ObsPy libraries as well as application codes and executables have been installed previously on the HPC cluster.

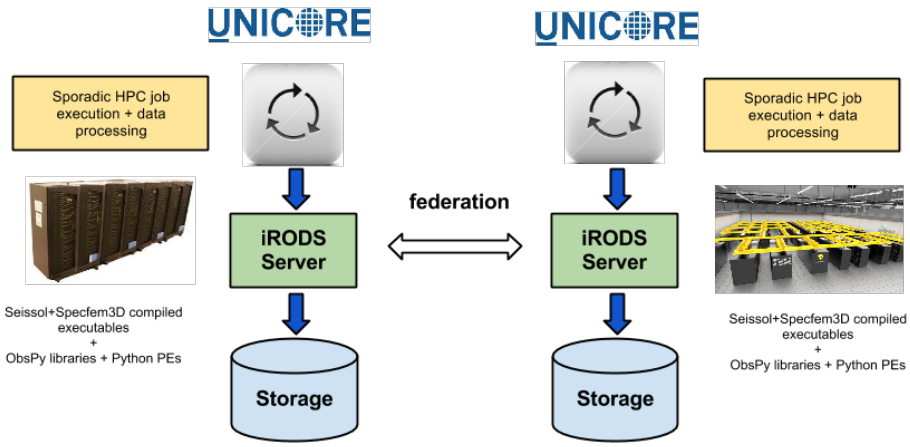


Figure 1: Schema of the UNICORE-based architecture for seismology workflows execution

Adopting this architectural model data-intensive operations are executed directly on HPC resources, thus giving two advantages. First output files from simulations are already stored very close to the place where the post-processing begins (GridFTP transfer is not mandatory to retrieve output files). Secondly the data-intensive part can take advantage of the HPC resource capabilities and pre/post-processing scripts can be submitted as computational jobs.

This approach can be potentially interesting if we think to misfit calculation within the waveform inversion use case. In fact a potential issue could arise for large inversions over a large number of events. Following a distributed approach files are put on different file systems thus reducing access to single ones. In this case the right logic would be properly chosen by the user, setting parameters from the workflow interface and selecting, for each step, the HPC resource where to execute the simulation.

4.1 The Processing Elements

In VERCE the Processing Element (PE) is intended as a conceptual entity able to perform general tasks on data. Behind these elements there's often a script which takes one/more files in input and gives back an output result. One processing element for example is deputed to produce input files for simulations starting from models and meshes taken from the repository, a second PE is designed to submit jobs to external HPC resources and a third PE takes care to stage output files. According to this philosophy we want to show how a little set of UNICORE GridBeans can be generated in order to implement some of the VERCE PEs base functionalities. Moreover, we will see how to exploit capabilities of the "data-intensive" PEs. These are Python scripts able to perform data-intensive tasks on a base64 data stream.

4.1.1 The Data Intensive PEs

The Data Intensive PEs have been implemented from the VERCE Data-Intensive Task Force, providing a single script for each specific operation to be executed. Main tasks involve accessing and processing raw data from an increasing number of stations. The Python scripts used in our tests for post-processing tasks are:

- **MapSeissolOutputToStations.py** - this Python PE takes as input the path of the SeisSol output directory and the used_stations.pkl configuration file to map the SeiSol output files to seismic stations. A file with correct station code/file name mapping is produced;
- **ConvertSeissolOutput.py** - this script takes as input a tuple from the output of the above PE and converts a SeisSol output .dat file in a base64 file;
- **SingleValuedPhaseMisfit_MS.py** - performs misfit calculation comparing synthetic output against real data. Takes as input two different base64 files.

For a more realistic application the data streams would require some pre-processing before executing the misfit-function (i.e. instrument correction and filtering).

As a starting point for this work we assume that all the Python scripts used to implement PEs functionalities import the same Python module verce.py:

```
from admire.verce import *
```

and they receive input and return output in the form of JSON strings. The command line schema used to test the Python processing elements is the following:

```
seismotestcase$ cat <input file> | python <your script>
<VERCE JSON> <PARAMETERS JSON ( defined accordingly to the
PE parameters ) > >< output file >
```

The <VERCE JSON >is a dictionary for data and process management information. This dictionary is used by the user code, for instance, to read the location of the input and output resources within a local file system. The <PARAMETERS JSON >is a dictionary whose items are defined by the developers of the PEs, the developers will specify names and values of the parameters needed by the analysis code to be tested (<yourscript >). More informations can be found in[11].

For example, the pySeisolFileReadTest.in contains the .pkl file path to map synthetic ouput files to seismic stations and the local path to the synthetics output folder:

```
{"streams":[{"data": "./used_stations.pkl"},
{"data": "../out/"}]}
```

Starting from this assumption what we have to do is to create proper GridBeans exposing a graphical interface and input fields for parameters, the GridBeans will be associated to proper scripts and applications to be executed on the cluster. Moreover, we have to handle the DataFlow passing output files through subsequent steps within the UNICORE

workflow. How the Python scripts can been used to build processing elements in the form of UNICORE GridBeans will be explained below.

5 Technology description

In this section we present a set of tools and software libraries that have been installed, each software package has been previously approved by VERCE partners as a result of an use-cases analysis. Applications as Specfem3D, SeisSol and ObsPy have been installed to PLX and therefore deployed on the CINECA facility.

- **Specfem3D** - Specfem3D is one of the solvers used in forward and inverse simulations to compute appropriate wavefields in two or three dimensions. All the software is written in Fortran2003 and employs parallel programming based on the Message Passing Interface (MPI). Specfem simulates wave propagation using a continuous Galerkin technique called SEM (Spectral Element Method)[12];

- **SeisSol** - SeisSol simulation software mimics seismic wave propagation in realistic media with complex geometry[13]. It is written in Fortran and, similar to Specfem3D, uses parallel programming based on the Message Passing Interface (MPI). SeisSol simulates wave propagation using a discontinuous Galerkin technique, which enables the use of tetrahedral meshes;

- **ObsPy** - ObsPy is an open-source Python framework for processing seismological data that includes typical processing routines, parsers for common file formats[14], etc. A module has been created on the PLX in order to setup the ObsPy environment: module load profile/advanced autoload ObsPy;

- **UNICORE GridBeans** - Four example UNICORE GridBeans have been created for testing purposes, the first GridBean is used to submit Specfem jobs on the PLX machine providing proper input files and parameters, the second one supports SeisSol applications, the third performs data-staging activities through iRODS and the last one is used to perform misfit calculation. The DataFlow has been managed by providing right input/output files for each GridBean. In particular output files produced by simulations (i.e. SeisSol) are treated as UNICORE workflow files and connected with a link to the input files of the subsequent steps.

6 Testing

6.1 A first workflow

The first functionality tests (Figure 2) have been performed submitting SeisSol and Specfem simulation jobs to UNICORE on the CINECA PLX system. The SeisSol simulation has been executed using input data related to the Central Italy Earthquake in 2009. This simulation has produced a very large amount of output files with .dat extension even running with a low number of iterations (about 80Mb of data for 10 iterations).

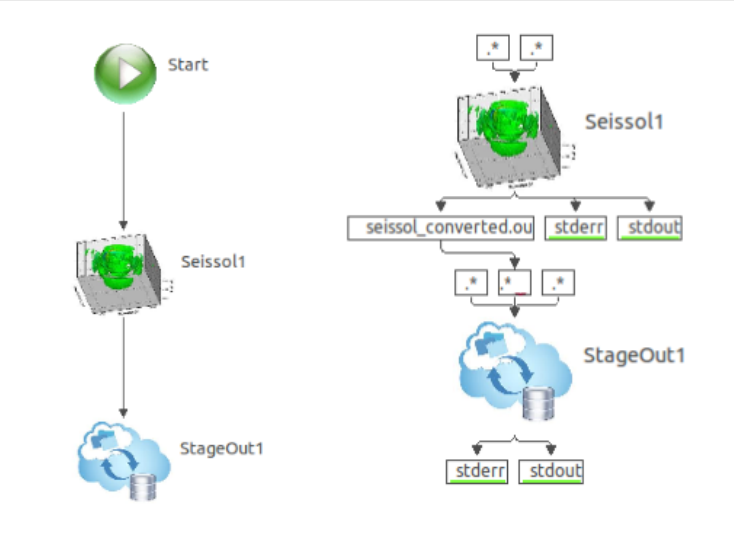


Figure 2: Screenshot of the two-step SeisSol workflow taken from the UNICORE Rich Client workflow editor

As a first test the SeisSol job has been submitted importing all the input files and necessary scripts in the same input directory. In the near future some of these files, as for example the meshes or the used_stations.pkl files, can be imported automatically from the iRODS file system or from the VERCE repository using some pre-command or setting urls and file paths from the GridBean panel. After the .dat output files have been produced (see Figure 4) the ObsPy module is loaded.

```
module load autoload profile/advanced ObsPy
```

Then the MapSeissolOutputToStations.py script is executed automatically in the same SeisSol job directory and a JSON file containing the mapping between .dat files and station codes is generated. This is an example of the JSON output from MapSeissolOutputToStations.py:

```
metadata: {"stateful": false, "streams": [{"content":
"({'code': 'PLOR1', 'elevation': 706.0, 'network': 'RO',
'longitude': 26.6466, 'depth': None, 'station': 'PLOR1',
'starttime': '2007-11-01T00:00:00', 'latitude': 45.852},
u'.//test-resources/seissol_converter/out/
out-pickpoint-00001-00001.dat')", "annotations": "",
"id": "node342-021aada4-c204-11e2-8692-00215ec779a8",
"location": ""}, {"content": "({'code': 'KEV'...
```

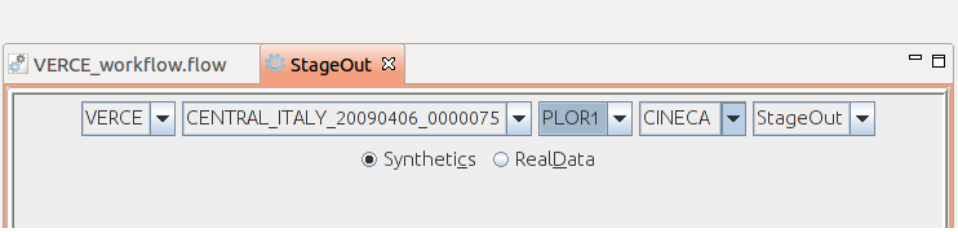


Figure 3: Screenshot of the iRODS GridBean, field values must be selected in order to perform stagein/stageout of files from/to iRODS file system

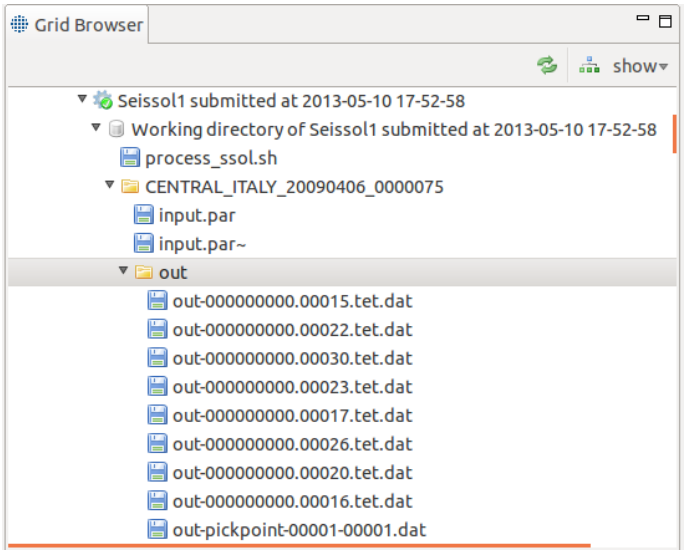


Figure 4: SeisSol output files from the grid browser

The ConvertSeissolOutput.py receives a tuple from the output above and produces the base64 output file for the selected station.

Finally the ouput file is uploaded to the iRODS repository using the StageOut iRODS GridBean. This GridBean implements a sort of data staging PE and provides a simple interface to select parameters: archive name, station code, event name, zone, staging mode. Once the options have been selected from the tent menu (Figure 3) and the SeisSol simulation has completed the GridBean invokes a script which loads the irods module:

```
module load irods
```

and executes the iRODS *iput* command to perform the stageout on the iRODS file system, the command is executed interactively on the login node.

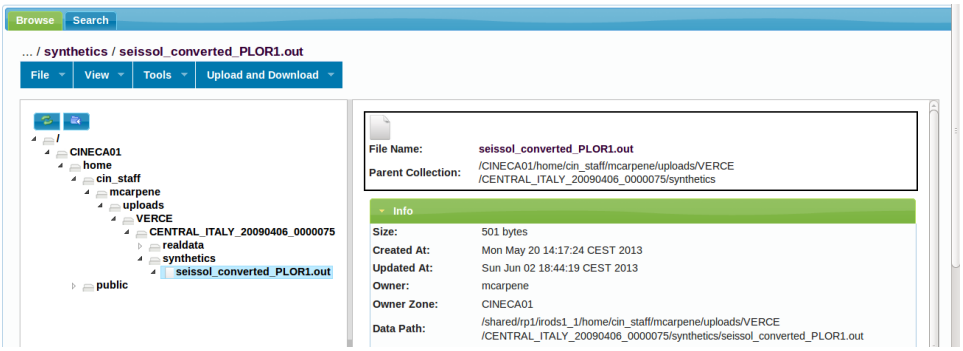


Figure 5: Screenshot of the iDROP web portal, for testing purposes files have been automatically uploaded to the user home folder

Metadata are also added (*imeta* command) to associate the right station code to each file, this is done in order to retrieve later the file for further processing. For the first test we have used only one file for one station (seissol_converted_PLOR1.out) that has been staged out on the file system. The iRODS GridBean must be capable to receive all the base64 files with proper extension and upload them with corresponding metadata. For this reason all the base64 output files have to be associated with the proper station and the mapping file must be sent to the iRODS bean.

Figure 5 shows the CINECA iDROP web interface where the files stored can be visualized in a filesystem view. Files can be managed by the user directly from the web interface as well as metadata.

6.2 The Misfit

The second use-case that has been tested is the misfit calculation (Figure 6), for this use-case a Misfit GridBean has been created. This GridBean basically is associated with the SingleValuedPhaseMisfit_MS.py Python PE, this Python script takes as input two base64 strings inside a JSON file and computes the phase misfit between these traces. We assume that the file with real data has been previously staged in the iRODS file system to be compared with synthetics and the size of each trace is in a range between 300kb and 1MB. Default parameters for the Python script have been provided but proper values can be set through the GridBean interface.

```
class SingleValuedPhaseMisfit(SeismoPreprocessingActivity):
...
st1=InputData['stream1']
st2=InputData['stream2']
...
parameters=default_parameters.update(self.parameters)
#time step between two samples in st1 and st2
dt = float("%s" % (parameters.get("dt")))
```

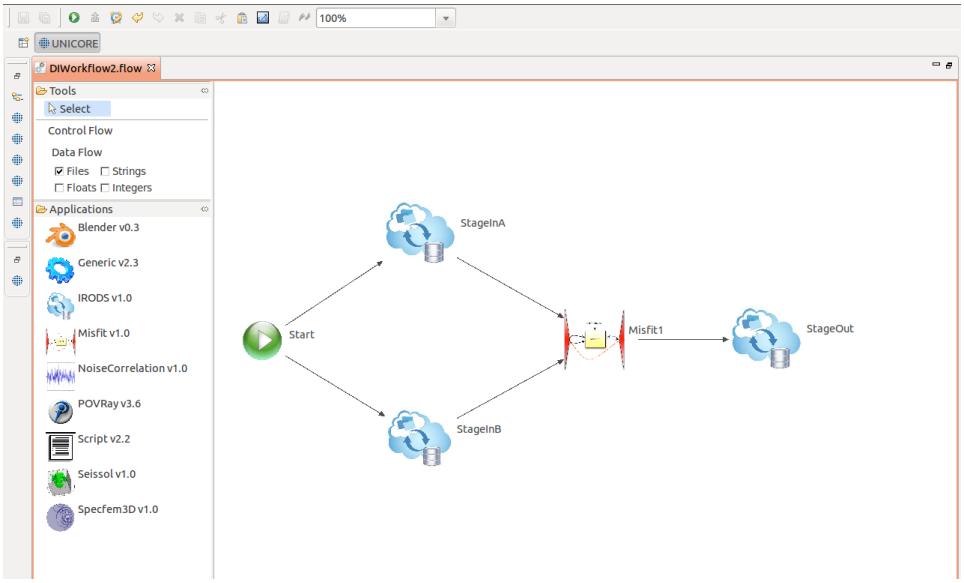


Figure 6: Screenshot of the ControlFlow for the misfit calculation, taken from the UNICORE Rich Client workflow editor

```
#minimal frequency to be analysed
fmin = float("%s" % (parameters.get("fmin")))
fmax = ... #maximal frequency to be analysed
nf = ... #number of frequencies (will be chosen with
logarithmic spacing)
w0 = ... #parameter for the wavelet, tradeoff between
time and frequency resolution
norm = ... #'global' or 'local' normalization of the misfit
phase_misfit = pm(st1[0].data, st2[0].data, dt, fmin, fmax,
nf, w0, norm, st2_isref=False)
...
```

Data-intensive tasks (like multiple misfit calculations) can be executed in parallel on the HPC infrastructure. Multiple job submission commands would be executed for each couple/group of traces (synthetic and real data) within the same GridBean script. Exploiting the SingleValuedPhaseMisfit_MS.py PE can be done introducing a bit of additional logic, some scripts must be run to retrieve files to be processed as well as passing right parameters to the PE.

Input files are passed as UNICORE workflow files to the Misfit GridBean for computation. In case of large number of traces to be processed input files should be accessed directly on the file system exploiting the file paths. Since in the case of the misfit the processes are completely independent there is no risk to incur in conflicts accessing single files during a single workflow execution.

Finally, after the misfit computation has completed the result can be stored again on iRODS with an additional StageOut step. Other tasks like filtering and resampling should not require too much computation to be executed and they could be easily added as pre-commands in order to save time between subsequent steps.

7 Conclusions and future work

In this paper we have presented how the VERCE Python framework for data-intensive operations can be utilised also outside the Dispel context. We have shown in particular how the Python scripts created from the VERCE Data Intensive Task Force can be reused to implement UNICORE GridBeans in order to model the multi-step processing of large amounts of data through UNICORE workflows. Proceeding down this road we have also tested and proved the flexibility and reusability of VERCE's products.

The UNICORE-based approach can be potentially interesting for example because it could simplify the problem of moving files across sites in VERCE. Another potential advantage can be found with respect to the misfit calculation and full-waveform inversion use-case. We posit in fact that a fully-distributed approach could lead us to exploit capabilities of the HPC infrastructure and to reduce the number of accesses to single file systems.

The model presented in this paper is an ongoing work, in the future more GridBeans can be created to implement different use-cases, for example for the noise-correlation or remote visualization. Finally, it could be very interesting to define metrics and compare performances obtained submitting the same workflows through different service-oriented architectures.

8 Acknowledgements

VERCE is funded by the 7th Framework Programme of the European Commission, grant agreement number 283543.

References

1. `http://www.verce.eu`
2. `http://www.geodynamics.org/cig/software/specfem3d`
3. `http://www.geophysik.uni-muenchen.de/~kaeser/SeisSol`
4. `http://www.epos-eu.org`
5. `http://www.eudat.eu`
6. Michele Carpene', Iraklis A. Klampanos, Siew Hoon Leong, Emanuele Casarotti, Peter Danecek, Graziella Ferini, Andre Gemuend, Amrey Krause, Lion Krischer, Federica Magnoni, Marek Simon, Alessandro Spinuso, Luca Trani , Malcolm Atkinson, Giovanni Erbacci , Anton Frank, Heiner Igel, Andreas Rietbrock , Horst Schwichtenberg , and Jean-Pierre Vilotte; *Towards Addressing CPU-Intensive Seismological Applications in Europe*. Lecture Notes in Computer Science series - volume 7905, presented at ISC 2013, Leipzig.

7. Martin, P., Yaikhom, G.: *Definition of the DISPEL Language.* In Atkinson, M., Baxter, R., Brezany, P., Corcho, O., Galea, M., van Hemert, J., Parsons, M., Snelling, D., eds.: *THE DATA BONANZA: Improving Knowledge Discovery for Science, Engineering and Business.* John Wiley and Sons Ltd. (April 2013) 203–236.
8. `http://www.obspy.org`
9. `http://www.cineca.it`
10. `http://grid.in2p3.fr/jsaga/`
11. A. Spinuso (KNMI), C. Ramos (KNMI), L. Trani (KNMI), I. Klampanos (UEDIN): *D-SA3.2.1 – Scientific gateway: Updated report with usage and user feedback*, 30/03/2013.
12. J. Tromp, D.K., Liu, Q.: *Spectral-element and adjoint methods in seismology.* Communications in Computational Physics 3(1-32) (2008).
13. Martin Käser and Cristobal Castro and Verena Hermann and Christian Pelties: *SeisSol – A Software for Seismic Wave Propagation Simulations*, High Performance Computing in Science and Engineering, Garching/Munich 2009
14. `http://github.com/obspy/obspy/wiki`

On Enabling Hydrodynamics Data Analysis of Analytical Ultracentrifugation Experiments

Morris Riedel[1,4], Shahbaz Memon[1], Florian Janetzko[1], Norbert Attig[1], Thomas Lippert[1], Borries Demeler[2], Gary Gorbet[2], Raminder Singh[3], Lahiru Gunathilake[3], and Suresh Marru[3]

[1] Jülich Supercomputing Centre,
Research Centre Jülich, 52425 Jülich, Germany
E-mail: {m.riedel, m.memon, f.janetzko, n.attig, th.lippert}@fz-juelich.de

[2] The University of Texas
Health Science Center at San Antonio, San Antonio, USA
E-mail: demeler@biochem.uthscsa.edu, gegorbet@gmail.com

[3] School of Informatics and Computing
Indiana University, Bloomington 47405-7000, USA
E-mail: smarru, ramifnu@iu.edu

[4] School of Engineering and Natural Sciences,
University of Iceland, 101 Reykjavik, Iceland
E-mail: m.riedel@morrisriedel.de

A new modular approach to enable scientific applications with UNICORE has been designed which employs modern elements of seamless web based access for abstraction and encapsulation. The approach uses the Apache Airavata software framework which is applicable and used by a wide variety of scientific gateways, which represent a community-oriented web based interface to computing and storage resources. Scientific gateways have shown a broad impact in computational science and brought many advances to various scientific communities in the last decade and even more research advances are expected when using them with the ever-increasing amounts of large quantities of datasets that are often referred to as 'big data'. This paper provides insights into benefits for the UNICORE community in working with existing scientific gateways and their strong user communities. While this new approach is more general in nature, we offer throughout the paper a concrete example of how initial work with scientific gateway communities enable a hydrodynamics data analysis of analytical ultracentrifugation experiments. The approach improves the ease of use in using UNICORE by performing seamlessly computational science so that more scientists can benefit from the strong UNICORE capabilities and its underlying computational resources. Furthermore the approach allows for creating 'collaborative workspaces' specifically optimized for specific scientific communities, including data and computing resources and to make existing datasets accessible and useful to the broader scientific communities that use scientific gateways on a daily basis. The design approach has been tested in a real supercomputer deplomentwhich has been used by several researchers of a larger scientific community in the bio-chemistry field. We report first results of the inter-working of UNICORE with a specific scientific gateway while not losing sight of the more general applicability of the approach with other gateways.

1 Introduction

Since the early 1990s, digital data acquisition from the analytical ultracentrifuge laid the foundation to analyse sedimentation data using computational resources. The Ultrascan[6] data analysis application became a well-known multi-platform software package that is

able to take advantage of such resources in order to support the interpretation of analytical ultracentrifugation (AUC) experiments. Today exists a scientific community in the biochemistry domain that is basically formed around the UltraScan package and that spans across US, Europe and other regions of the world. The Ultrascan scientific gateway was proved to be useful in TeraGrid by scientists studying the solution properties of biological and synthetic molecules using computational and storage resources.

But a deeper analysis reveals that the resource situation for the US and European researchers is very fragmented within this community, mainly because of the different e-Science infrastructures that provide the computational and storage resources but offer access to those with different technologies, interfaces, and usage models. Examples of these e-Science infrastructures are the Extreme Science and Engineering Discovery Environment (XSEDE) in the US or the Partnership for Advanced Computing in Europe (PRACE). As we published earlier[4], these infrastructure are not interoperable due to a lack of common adoption of open standards in key technologies. We further have published earlier[5] that not only the UltraScan community, but several other scientific communities would like to take advantage of interoperable e-Science infrastructures.

But this raises the demand for seamless access without affecting the daily work practice of running scientific gateways or clients. Furthermore, the identified methods must avoid situations where European users submit jobs to US since they are unable to leverage EU systems from existing end-user environments such as the Ultrascan scientific gateway. Hence, a solution must have the potential to unify user community methods via the most appropriate technologies within a particular research community while not losing sight of the underlying broad range of potential resources in US, Europe, and elsewhere in the world.

In this paper, we will emphasize on the importance of supporting scientific gateways in the next decade within the UNICORE community since we observe an ever increasing trend of Web-based access methods in the context of scientific computing. This is not only the case for PRACE and XSEDE, but also for the high throughput computing oriented infrastructures European Grid Infrastructure (EGI) or the European data infrastructure (EUDAT) that also foresees lightweight Web-based access as part of its services inspired by the 'ScienceTube' design[3]. The paper will describe an approach that is scalable in terms of maintenance efforts by leveraging functionalities of the Apache Airavata framework[9], which is adopted by a wide variety of science gateways in order to provide a single coherent approach to access for computing and storage resources. The approach presented is this paper is thus applicable to many other scientific gateways while the UltraScan scientific gateway is taken as an example if context in order to provide evidence that this can be achieved.

The paper is structured as follows. After the problem space of the paper is introduced, our work is motivated in Section 2. Section 3 presents the modular design approach to extend the usage of UNICORE to new end-users. An application use case on enabling hydrodynamics data analysis is given as an example throughout the paper but Section 4 highlights the major achievements towards supporting this particular use case while outlining potential solutions for similiar scientific applications and other existing scientific gateways. After a brief survey of related work, the paper ends with some concluding remarks.

2 Motivation and Effort Considerations

In order to motivate our work, we model the problem space in four basic layers as shown in Figure 1. The first layer *'Core Technology'* refers to the UNICORE server components such as the enhanced Network Job Supervisor (XNJS)[2], or the UNICORE Target System Interface (TSI)[2]. The second layer *'Channel'* indicates the different main access methods that can we differentiate in three main categories: (a) Apache Airavata, (b) UNICORE Clients and APIs, and (c) UNICORE portals. The *'Infrastructure'* layer represents the major conceptual projects under which the channel is opened to the end-users including XSEDE, EUDAT, PRACE, EGI, or smaller national initiatives such as PL-Grid. The final layer *'End-users'* is represented by the scientific communities including their working environments that are mostly developed and maintained by domain-specific researchers.

Figure 1 further suggests that there is a kind of *'Amplification Factor'* that stands for the gain of influence or uptake via subsequent higher layers. This is important to consider in terms of efforts of a sustainable approach. In other words, we have to put the constraints that we want to influence many infrastructures and end-users (cf. layer 3-4) with relatively low amounts of channel and technology efforts (cf. layer 1-2). This raises the demand for a *'Modular Approach'* where we can re-use as many parts of the software from the core technology and channels as possible (e.g. through open standards, frameworks, etc.).

In the most traditional sense of UNICORE access, UNICORE was mostly used in infrastructures such as PRACE or EGI using the UNICORE Clients or technologies that re-used UNICORE APIs as channel. This method was used by many scientific communities (e.g. life sciences, physics, etc.). Since several years, efforts around multi-disciplinary portals (e.g. P-GRADE[1]) enabled a more general access to UNICORE functionality only tuned for one scientific domain to a limited degree (cf. right part of Figure 1). More recently, we observe a trend towards a massive increased usage of Scientific Gateways (e.g Ultrascan) in XSEDE or lightweight Web access in EUDAT (cf. left part of Figure 1).

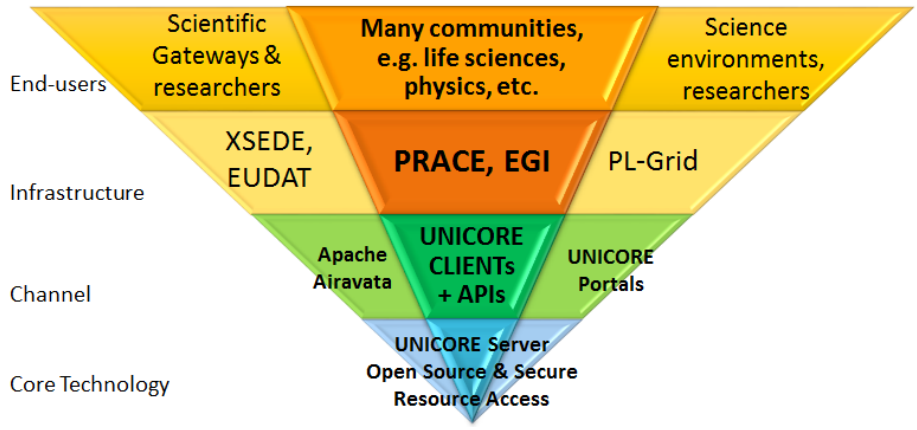


Figure 1: Overview of traditional access to UNICORE and modern Web-based access methods.

3 Modular Design Approach

The modular design approach presented in this section established the conceptual framework of how to support the wide variety of existing and emerging scientific gateways. It is illustrated in Figure 2 and aims to satisfy the following three key requirements we derive from the aforementioned introduction and motivational statements.

The first key requirement is named as *'Community-driven'* meaning that new access methods need to work nicely with existing community supported access tools. This is important to consider in order to (a) not disturbing the scientific way of work, (b) ensure the uptake of solutions, ensuring at least partly (c) shared maintenance of the tool or approach in question between community efforts and tool providers. In particular the latter is important since providing a new tool (e.g. portal) from the outside of the community to an existing community is very likely to not being used and on the other hand these tools are typically highly optimized for their use in their corresponding scientific communities. Figure 2 indicates therefore that our modular design approach keeps existing scientific gateway in the highest layer as important core building block while also providing functionalities that can be re-used by other established access methods (e.g. portals, scientific domain-specific tools like visualization clients) in lower layers. In our concrete example around analytical ultracentrifugation we stick to the already existing UltraScan scientific gateway and just change some of its elements with core building blocks of the underlying layer.

Another key requirement is to provide a *'Scalable solution'* that refers to the fact that we are not able to provide different solutions for each individual scientific gateway in terms of efforts. This is important, because the XSEDE infrastructure alone, for example, is used by 32 scientific gateways today while our particular UltraScan scientific gateway is only one of them. Hence, we need a framework or more generally a modular software approach that we can re-use again and again in different scientific gateways while maintaining only one coherent set of software in order to remain scalable in terms of efforts. Figure 2 therefore shows the abstraction from the access to compute and data resources with two modular software pieces named as Apache Airavata and middleware. Parts of the UltraScan scientific gateway consists of Apache Airavata that is in turn a community-driven open source project that integrates various different methods of accessing computing and storage resources. Alongside already existing access methods such as Java COG for Globus middleware and Amazon EC2 Cloud APIs, we augmented Apache Airavata with an open standard client interface that is adopted by middleware. In our specific example of the biochemistry community around UltraScan, we augmented functionality to Apache Airavata while the look and feel of the Scientific Gateway remained the same.

The third key requirement of our modular design approach is to be *'Standards-based'*. This is important in order to remain modular in terms of middleware systems so that different technologies can be used thus also increasing the trust of end-users in one technology. The use of open standards enables encapsulation of our approach making it possible to use a wide variety of standard-compliant middleware systems that are provided in different e-Science infrastructures today. Figure 2 illustrates that our model adopts the Open Grid Services Architecture - Basic Execution Service (OGSA-BES)[16] interface that is an open standard created by the Open Grid Forum (OGF). It enables computational job submission and management and is adopted in middleware systems such as UNICORE or GENESIS

just to list a few. The OGSA-BES interface in turn relies on another open standard named as the Job Submission Description Language (JSDL)[17] also created by OGF. In our specific example of the UltraScan Scientific Gateway, we just augmented an OGSA-BES client into the Apache Airavata system, including a job submission description adapter for JSDL. In other words, the scientific gateway itself remained unchanged while the changes are well encapsulated within the Apache Airavata that in turn also means that this functionality can be adopted by other scientific gateways as the next Section will reveal.

This section describes the high level modular design approach to show how we jointly addressed all three key requirements, while we refer to an earlier publication[10] for more architectural details in the context of the concrete UltraScan scientific community. But some small architecture concepts are useful to mention here in order to understand that also security core building blocks are parts of the modular design. In terms of security, members of the UltraScan community log-in with username/password at the scientific gateway. During the job submission in the gateway using Apache Airavata, this identity is being forwarded to the middleware encoded in security tokens that follow the open standard Security Assertion Markup Language (SAML)[19] that is also adopted by various middleware systems and one cornerstone of the XSEDE architecture[20].

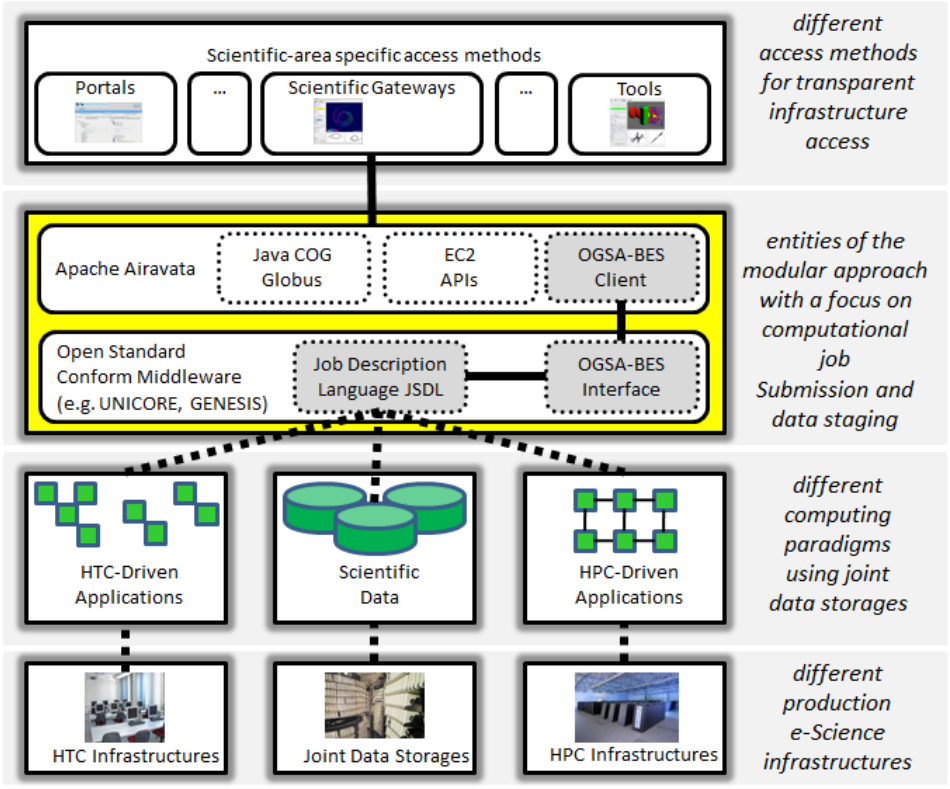


Figure 2: Conceptual view of the modular approach to support scientific gateways.

4 Hydrodynamics Data Analysis Application

This section provides more information of how the modular design approach described in the previous section is used to support a concrete hydrodynamics data analysis application. But while he describe this particular application of the UltraScan Science Gateway community we provide some information in context of how similiar applications, infrastructures, or science gateways can leverage the approach described in this paper. Therefore, Figure 3 also takes a broader view than just Ultrascan and as such includes details on how the UNICORE community can extend its number of end-users by an order of magnitude. As a consequence, we focus in this section on those infrastructures that bear the chance of having new users (e.g. EUDAT, XSEDE) while we neglect other infrastructures (e.g. PRACE, EGI) although the same concepts can be used in them, but end-users already exist using other access methods (e.g. UNICORE clients, APIs, etc.).

Figure 3 illustrates the use of Apache Airavata in the UltraScan Science Gateway in order to access the UNICORE middleware using the OGSA-BES standard or to transfer sedimentation data for analsys using transfer technologies that work with the GridFTP[18] standard. The UltraScan community can be thus considered as a new end-user community of UNICORE, but also other scientific gateways use Apache Airavata and as such have the possibility to use UNICORE underneath. For instance, the Biology Virtual Collaborative Labs (BioVLab)[7] scientific gateway is used to launch scientific bioinformatics applications by researchers. This includes protein sequence workflows, microarray workflows, microRNA and mRNA ingegrated analysis, and methzlation profiling of genomes containing methylated CpG sequences. Apache Airavata lowers the barriers for researchers to setup computational resources (e.g. clouds) to be used with the aforementioned bioinformatics tools (including databases), but in principle also UNICORE resources can be used (given computational time is granted).

Another example of a scientific gateway that works with Apache Airavata is the Ocean Land Atmosphere Model (OLAM)[8] Science Gateway. The community around OLAM uses this new global general circulation/climate prediction model to simulate desired resolutions with a flexible mesh refinement technique on computational resources. It further includes the use of local high-resolution land surface characteristics databases. Also in this case, Apache Airavata lowers the barriers for earth science researchers to setup computational resources to enable complex simulations that predict unrivaled regional climate change predications. Given the use of Apache Airavata and as illustrated in Figure 3, also UNICORE can be used given the community will have applied for computational resources.

The aforementioned examples are just three scientific gateways that leverage US XSEDE resources when considering the infrastructures layer. In order to extend the usage of UNICORE and its users, we have worked on a concrete UltraScan deployment in Europe on the JUROPA HPC system (cf. Figure 3) at the Juelich Supercomputing Centre. JUROPA has 2208 compute nodes based on Intel Xeon X5570 (Nehalem-EP) quad-core processors, 24 GB memory (DDR3), and Infiniband interconnect and as such optimal for being used with closely-coupled parallel applications that take advantage of the Message Passing Interface (MPI). As shown in Figure 3, UltraScan makes use of MPI and works on the JUROPA system being access with the UltraScan scientific gateway and UNICORE using Apache Airavata. The JUROPA system and the allocations that have been obtained from members of the UltraScan community are not part of PRACE, but as it works in

the JUROPA environment the steps to create similiar setups in the US on XSEDE service provider sites with UNICORE are not difficult to perform. Also, given the success of the UltraScan and UNICORE integration, some members of the UltraScan community already plan to submit computational time requests in PRACE. Also, members of the UNICORE community have started to get in contact with other scientific gateways in order to enable the applications of computing systems in Europe and to help to get computational resources on rare HPC resources.

Finally, we would like to outline also possible extensions of UNICORE end-user communities by using the same approach of this paper in the context of big data processing using smart analytics that enable 'high productivity processing'[11]. To provide a concrete example, the EUDAT collaborative data infrastructure (CDI) is implementing a design[3] that also considers very lightweight clients that are very similiar to scientific gateways. The *'Simple Store'* service of EUDAT, for example, is a GUI that will allow for user community branding and offers a YouTube-like service as described in our earlier publication[3]. In order to enable quick processing the Simple Store service requires hooks for methods to initiate easy data processing while open standards such as OGSA-BES as adopted by Apache Airavata offers the best choice in stability when it comes to different execution interfaces that are deployed on production e-Science infrastructures. The EUDAT Simple Store service will allow for hooks that enable the embedding of added value services in context of dedicated datasets and here a hook for the Apache Airavata will enable the submission and management of computational data analysis within a community-specific rendering of the EUDAT YouTube-like service.

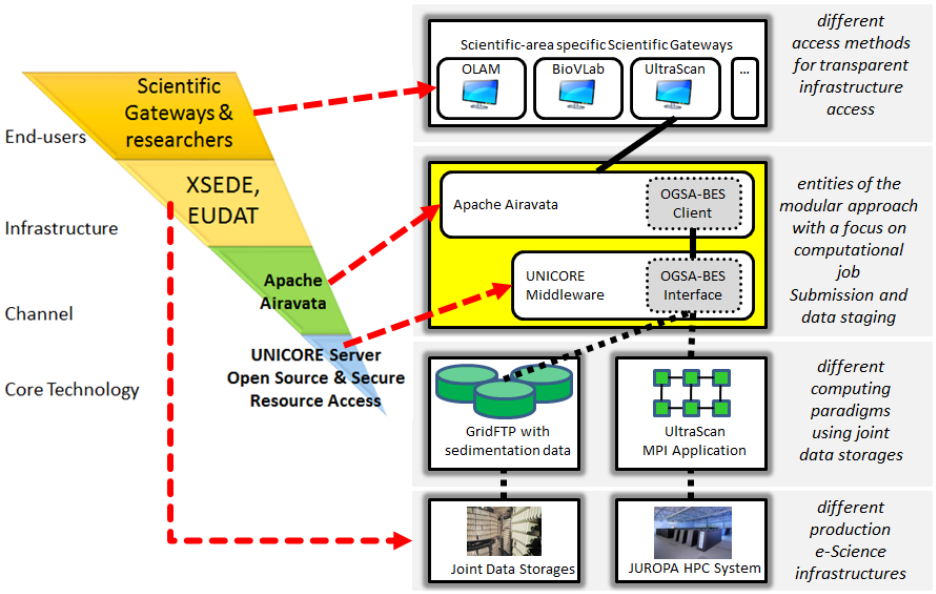


Figure 3: Running Ultrascan deployment in context of other scientific gateways and problem space model.

5 Related Work

There is a wide variety of related work in the field of APIs and Web-based access around UNICORE, while we aim to focus on those that are most relevant in context of scientific gateways and to the best of our knowledge no other work on scientific gateways and UNICORE access are performed today. The P-GRADE Portal[1] abstracts from middleware technologies by using a wide variety of adapters and it is broadly used. In contrast to our approach, we believe that the requirement of being *'community-driven'* as only partly achieved since it is maintained from the P-GRADE community that is rather multidisciplinary and not part of one dedicated community. Apart from these considerations, the Apache Airavata system is already adopted by Ultrascan and as such being an important factor of the principle of the *'Amplification-factor'* described earlier to achieve the scalability in terms of the number of scientific gateways. Another important related work activity is the Simple API for Grid Applications (SAGA) framework[12]. The SAGA framework is similar like the Apache Airavata framework but is currently not adopted by the scientific gateways we have worked with. Nevertheless, SAGA includes adapters for OGSA-BES that would enable similiar approaches than those provides as part of this paper. Finally, we would like to mention the activities around the Vine Toolkit[14] that provides numerous adapters for various middleware technologies but is not integrated in known scientific gateways and as such not completely a scientific community driven approach.

6 Conclusions

We conclude that the taken approach is able to satisfy the three key requirements that we have find the most important ones while working in scientific computing since over a decade now. Hence, we provide an approach that address the requirement of being a community-driven approach (i.e. uptake of existing scientific gateways at least partly maintained in scientific communities) and offering a scalable solutions since our approach can be re-used in a wide variety of existing scientific gateways by using the Apache Airavata framework. We further satisfy the requirement of using open standards that enable the usage of a broad range of underlying standard-compliant middleware while the OGSA-BES interface is a stable specification since nearly ten years now offering the basic functionality needed when working with computational and storage resources. Finally, we point to some future work topics that go beyond the already planned work with other scientific gateways than UltraScan. As in many European and US infrastructures, the security topic is one fundamental complex aspect that hinders interoperability. While EUDAT in EU is considering EduGain[15], the US XSEDE infrastructure is working on solutions towards InCommon federations[13]. While we are following in depth federated AAI solutions in the UNICORE community we acknowledge that work is needed in order to harmonize the security setups.

Acknowledgements

This work in this paper is partially supported by the Extreme Science and Engineering Discovery Environment (XSEDE) project of the National Science Foundation (NSF) grant number OCI-1053575. The particular work on Figure 1 is inspired by work in the European Middleware Initiative (EMI) project together with Alberto Di Meglio (CERN).

References

1. P. Kacsuk *P-GRADE Portal Family for Grid Infrastructures*, Concurrency and Computation: Practics and Experience **23**, 235–245, 2011.
2. A. Streit, P. Bala, A. Beck-Ratzka, K. Benedyczak, S. Bergmann, R. Breu, J. M. Daivandy, B. Demuth, A. Eifer, A. Giesler, B. Hagemeier, S. Holl, V. Huber, N. Lamla, D. Mallmann, A. S. Memon, M. S. Memon, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, T. Schlauch, A. Schreiber, T. Soddemann, and W. Ziegler, *UNICORE 6 - Recent and Future Advancements*, Annals of Telecommunications, Springer **65**, 757–762, 2010.
3. M. Riedel, P. Wittenburg, J. Reetz, M. van de Sanden, J. Rybicki, B. von St. Vieth, G. Fiameni, G. Mariani, A. Michelini, C. Cacciari, W. Elbers, D. Broeder, R. Verkerk, E. Erastova, M. Lautenschlaeger, R. Budig, H. Thielmann, P. Coveney, S. Zasada, A. Haidar, O. Buechner, C. Manzano, S. Memon, S. Memon, H. Helin, J. Suhonen, D. Lecarpentier, K. Koski and Th. Lippert, *A Data Infrastructure Reference Model with Applications: Towards Realization of a ScienceTube Vision with a Data Replication Service*, Journal of Internet Services and Applications **4**, 1, 2013.
4. M. Riedel, E. Laure, T. Soddemann, L. Field, J.P. Navarro, J. Casey, M. Litmaath, J.P. Baud, B. Koblitz, C. Catlett, D. Skow, C. Zheng, PM. Papadopoulos, M. Katz, N. Sharma, O. Smirnova, B. Kónya, P. Arzberger, F. Würthwein, A.S. Rana, T. Martin, M. Wan, V. Welch, T. Rimovsky, S. Newhouse, A. Vanni, Y. Tanaka, Y. Tanimura, T. Ikegami, D. Abramson, C. Enticott, G. Jenkins, R. Pordes, S. Timm, G. Moont, M. Aggarwal, D. Colling, O. van der Aa, A. Sim, V. Natarajan, A. Shoshani, J. Gu, G. Galang, R. Zappi, L. Magnoni, V. Ciaschini, M. Pace, V. Venturi, M. Marzolla, P. Andreetto, B. Cowles, S. Wang, Y. Saeki, H. Sato, S. Matsuoka, P. Uthayopas, S. Sriprayoonsakul, O. Koeroo, M. Viljoen, L. Pearlman, S. Pickles, D. Wallom, G. Moloney, J. Lauret, J. Marsteller, P. Sheldon, S. Pathak, S. De Witt, J. Mencák, J. Jensen, M. Hodges, D. Ross, S. Phatanapherom, G. Netzer, A.R. Gregersen, M. Jones, S. Chen, P. Kacsuk, A. Streit, D. Mallmann, F. Wolf, T. Lippert, T. Delaitre, E. Huedo, and N. Geddes, *Interoperation of World-Wide Production e-Science Infrastructures*, Concurrency and Computation: Practice and Experience **21**, 961–990, 2009.
5. M. Riedel, F. Wolf, D. Kranzlmüller, A. Streit, T. Lippert, *Research Advances by Using Interoperable e-Science Infrastructures - The Infrastructure Interoperability Reference Model Applied in e-Science*, Cluster Computing **12**, 357–372, 2009.
6. B. Demeler, *UltraScan: A Comprehensive Data Analysis Software Package for Analytical Ultracentrifugation Experiments*, Modern Analytical Ultracentrifugation: Techniques and Methods **210–229**, 2005, .
7. Z. Zang, J. Choi, C. Herath, S. Marru, and S. Kim *Biovlab: Bioinformatics data analysis using cloud computing and graphical workflow composers*, Cloud Computing and Software Services **309**, 2010, .
8. R. Walko and R. Avissar, *The ocean-land-atmosphere model (OLAM) Part I: Shallow-water tests.*, Monthly Weather Review **136(11)**, 4033–4044, .

9. S. Marru, L. Gunathilake, C. Herath, P. Tangchaisin, M. Pierce, C. Mattmann, R. Singh, T. Gunarathne, E. Chinthaka, R. Gardler, A. Slominski, A. Douma, S. Perera and S. Weerawarana, *Apache airavata: a framework for distributed applications and computational workflows*, Proceedings of the 2011 ACM workshop on Gateway computing environments **21–28**, 2011, .
10. S. Memon, M. Riedel, F. Janetzko, N. Attig, Th. Lippert, B. Demeler, G. Gorbet, S. Marru, R. Singh, L. Gunathilake , A. Grimshaw *Improvements of the UltraScan Scientific Gateway to Enable Computational Jobs on Large-scale and Open-standards based Cyberinfrastructures*, Proceedings of XSEDE 2013, **San Diego**, 2013, .
11. M. Riedel, M. Memon, A. Memon, G. Fiameni, C. Cacciari, and Th. Lippert, *High Productivity Processing - Engaging in Big Data around Distributed Computing*, Proceedings of the 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) **Opatija, Croatia**, 2013, .
12. S. Jha, H. Kaiser, A. Merzky, and O. Weidner *Grid Interoperability at the Application Level Using SAGA*, Proceedings of the IGIIW Workshop at the third IEEE Int. Conference on e-Science, Bangalore, India **584–591**, 2007, .
13. J. Basney, T. Fleury, and V. Welch *Federated login to teragrid*, Proceedings of the 9th Symposium on Identity and Trust on the Internet **1–11**, 2010, .
14. M. Russell, P. Syiubecki, P. Grabowski, M. Krysinski, T. Kuczynski, D. Szjenfeld, D. Tarnawczyk, G. Wolniewicz, and J. Nabrzyski, *The Vine Toolkit: A Java Framework for Developing Grid Applications*, Proceedings of the 7th Int. Conference on Parallel Processing and Applied Mathematics **Gdanks, Poland**, 2007, .
15. D. Lopez, *eduGAIN: Federation Interoperation by Design*, Proceedings of TERENA Networking Conference , 2006, .
16. I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, and M. Theimer, *Open Grid Services Architecture - Basic Execution Service Version 1.0*, OGF Grid Final Document **29**, 2004, .
17. A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, *Job Submission Description Language (JSDL) Specification Version 1.0*, OGF Grid Final Document **136**, 2008, .
18. W. Allcock, *GridFTP: Protocol Extensions to FTP for the Grid*, OGF Grid Final Document **20**, 2006, .
19. S. Cantor, J. Kemp, R. Philpott, and E. Maler, *Assertions and Protocols for the OASIS Security Assertion Markup Language*, OASIS Specification , 2005, .
20. F. Bachmann, I. Foster, A. Grimshaw, D. Lifka, M. Riedel, S. Tuecke, XSEDE Architecture Level 3 Decomposition, 2012, Online: `https://www.xsede.org/documents/10157/281380/XSEDE_Architecture_Lev3_Decomposition-v0.9.pdf`

Experiences Running Data Extraction Applications using UNICORE

Lara Flörke and Mathilde Romberg

Jülich Supercomputing Centre,
Research Centre Jülich, 52425 Jülich, Germany
E-mail: {l.floerke, m.romberg}@fz-juelich.de

The huge, increasing amount of chemical patents makes it impossible for humans to analyse them manually. The automation and speedup of the analysis process is a subject of the German research project UIMA-HPC. It uses UNICORE workflow features and applications suited to run on HPC systems to reach its goal. The UIMA (Unstructured Information Management Architecture) framework is used in the applications to ensure interoperability between the different analysis tools.

This paper focuses on workflow modeling for the data extraction process. It discusses those features from UNICORE that support such kind of use case as well as the features that UNICORE lacks in order to achieve a minimal time to solution.

1 Introduction

Data extraction from chemical patents is a quite complex process in which different analysis steps are combined to annotate the documents. The steps identify different categories such as sentences, chemical names or chemical structures. Input to the process ranges from kilo- to hundreds of gigabytes with documents of type pdf, txt or others, while output is of increased size, because of the annotations, and can be for example input to a (tripe store) data base [1], [2].

In order to modeling the data extraction process as an UNICORE workflow the analysis steps have to be implemented as applications with standardised I/O format. This is achieved by using the UIMA framework [2], [3]. The various data extraction tasks are very different in their resource requirements and computing time, e.g. from days to a couple of minutes for the same amount of input data. In general they are all high-throughput applications processing the different input documents independently. The UIMA framework supports them in making use of the available cores of a node so that multiple documents are processed in parallel threads [3]. The workflow exploits the for loop and iterates over input files sending either a certain volume of data or a certain number of input files to one job. The work is done within the UIMA-HPC project (funded by the German Ministry of Education and Research (BMBF), grant id 01IH11012A-D) [1].

As one of the aims is to reach shortest time to solution, many different possibilities of workflow arrangements were tested. Hence, in this paper different workflow tests are presented which reveal benefits that UNICORE already provides, but also the limitations caused by UNICORE.

The paper is structured in the following way: section 2 explains the use case in detail and in section 3 different tests and results are described as well as the observed advantages UNICORE offers and the restrictions UNICORE imposes.

2 The Use Case

Currently, there exists one major use case. The analysis of chemical patents, which are available as text or pdf files, is the subject to this use case. The question it should answer is, for example, if the input documents contain relevant information on a certain topic like a chemical. The information extraction process consists of a sequence of different kinds of applications which annotate the input files. First of all, the input text and pdf files are converted into a common data structure, which enables the communication between the following analysis components so that the adjacent analysis steps can be performed. This data structure is called CAS (Common Analysis Structure). In the case of UIMA-HPC CAS is represented in XML so that the input files have to be transformed into XCAS [4], [5], [6].

After this conversion other applications are executed which, for example, identify sentences, words or the corresponding word type. Subsequently executed applications recognise, for instance, chemical terms or enzymes based on databases or dictionaries and annotate these [7], [8].

At the end, the results from the different annotation steps have to be converted into an output format so that a suitable presentation of the results is achievable. One currently existing possibility is the generation of a CSV (Comma Separated Values) file, which contains the output. Moreover, the outcome can be converted into RDF (Resource Description Framework) format, which is used to store the data into a triple store [9]. It is also possible to create an output that can be displayed with *brat*, a tool that shows the text with special marks in the web browser [10].

The information extraction process is modeled as an UNICORE workflow, which provides control structures such as for loops iterating over files or file sets including parallel submission of jobs and automatic data transfer between subsequent steps (figure 1). As one job uses the output files of the previous job, the output data of a workflow step is transferred to the workflow's working directory. From there the output files are transferred to the next job's working directory. With this procedure the workflow service ensures that all data needed in the workflow is available during the lifetime of the workflow. With the UIMA framework the analysis applications offer an unified interface and exploit multithreading [3], [6].

Altogether twelve different applications are executed within the workflow: In the first phase txt input files are preprocessed in parallel to the preprocessing of pdf input files. The preprocessing of pdf files consists of the identification of text regions and the optical character recognition and is executed in a for loop. After this another for loop containing the other applications follows. Here the preprocessed text and pdf files are joined and serve as an input for the subsequent applications, which perform the natural language processing, recognise the chemical terms and create the different output formats in parallel so that the output can be visualized with other tools.

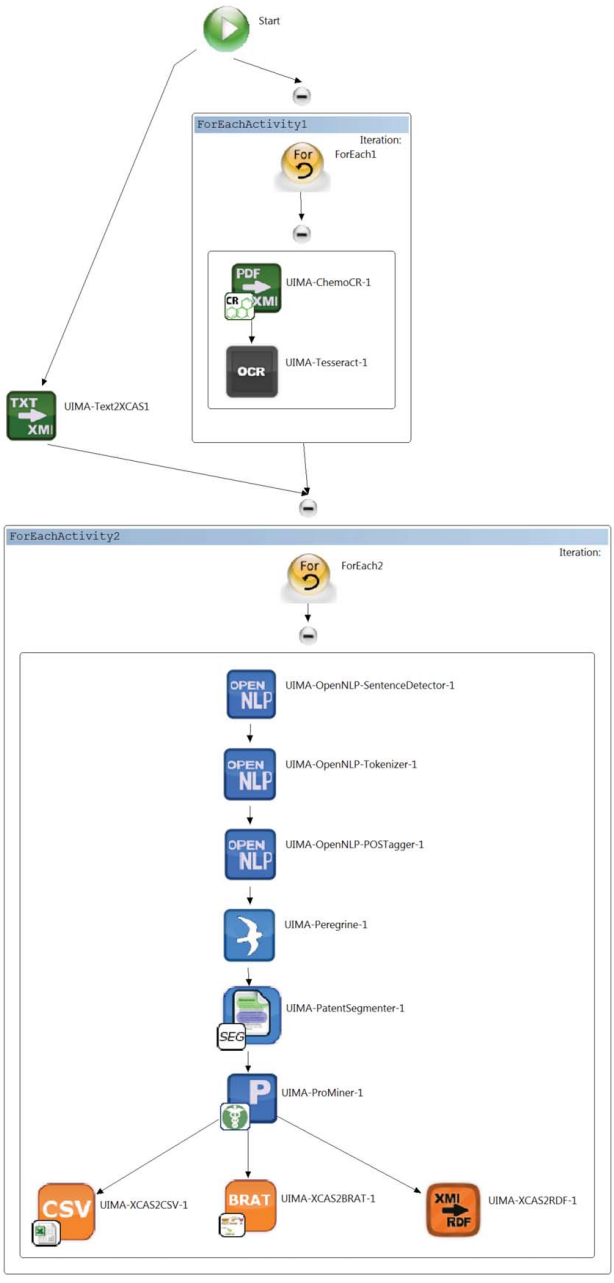


Figure 1: Workflow to analyse chemical patents.

3 Advantages and Restrictions

Different tests have been executed to examine how to reach the shortest time to solution of the workflows. During this process some advantages were discovered that UNICORE already offers, but also missing aspects were noticed that would be helpful to improve the performance.

The use case was tested with an input of 60 text files and 60 pdf files. At the beginning the size of the 60 text files is nearly 9 MB and the size of the pdf files is 170 MB. However, the data size of all documents together sums up to nearly 3 GB during the analysis process. There exist several workflows that all represent the same use case, but they differ from each other by treating unpacked or packed input data. Besides, there is one case in which the workflows have additional jobs to equally distribute the input data. Below all stated values are averages. The results were measured on a testbed consisting of a cluster with 54 nodes each with 24 cores and 2 workstations each with 1 node and 4 cores.

In the first execution scenario of the use case all input files were packed into one tar file so that the files have to be unpacked before running the applications and packed after the execution. Since the input files are all packed into one tar file, within these tests only one job is started for each application. This case was tested with the different data transfer protocols BFT (Baseline File Transfer) and UFTP (UNICORE File Transfer Protocol). With the use of the protocol BFT the stagein and stageout time of the input and output files amount to about 5 hours and 50 minutes (table 1). As opposed to this, the stagein and stageout time using UFTP amounts to about only 13 minutes and 28 seconds.

	first case (total transfer time for one packed file)	second case (total transfer time for several packed files)	third case (total transfer time for unpacked files)
BFT used	05:50:08	06:23:46	11:52:59
UFTP used	00:13:28	00:53:34	13:53:15

Table 1: Overview of the measured transfer times.

This shows that UNICORE with UFTP already provides fast transport of big files. Nevertheless, it is preferable to eliminate the transfer time and the number of transfers between different jobs by avoiding the transfer of data between two steps, when one job uses the output of the preceding job. Because increasing data input sizes leads to a higher transfer time also when using UFTP, the time to solution can only be minimized, when the amount of data to be transferred is reduced to a minimum.

In the second execution scenario of the use case two additional jobs were inserted into the workflow before the two for loops to pack the input data into several tar files with similar data size. The advantage is that some jobs can run in parallel, which reduces the time to solution. But due to the larger quantity of jobs, more file transfers and more packing and unpacking of input and output files have to be done. In this test case, executed with BFT, the transfer time in total is 6 hours and 23 minutes (table 1). When using UFTP the total stagein and stageout time is about 53 minutes. This example also shows that due to the file transfer a lot of time is lost during the analysis process.

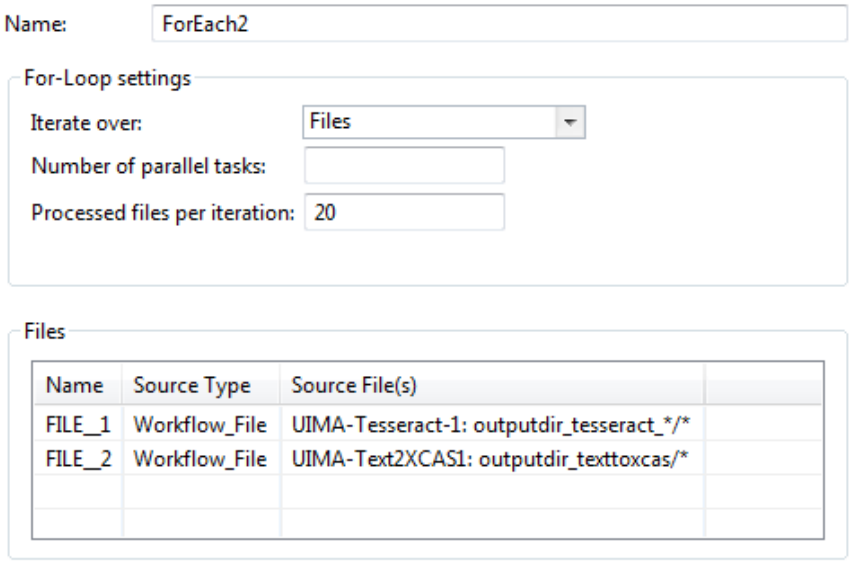


Figure 2: For loop with a specified file number as input for the jobs. A value greater than 1024 in the *Processed files per iteration* field is treated as a data size.

In addition, it has to be taken into account that these two test cases spend extra time for unpacking and packing of the files before and after an application is executed. This time is also wasted. In the test cases the total packing and unpacking time lies between 5 and 7 minutes. However the packing of the input files into different tar files of similar size offers the possibility to run jobs in parallel, which enables to reach a shorter time to solution. The similar data size entails that the jobs have almost the same runtime so that all job threads in the for loop finish at almost the same time.

Due to the implications of packing and unpacking the files, in the third execution scenario workflows with unpacked input files were executed. In the properties of the for loop the possibility is given to iterate over files (figure 2). When iterating over files there is an option to specify a number of files that should be processed during a job. Thus, only jobs whose particular number of input files is equal or smaller than the number of files mentioned in the for loop will be executed. Moreover, a data size for the input of each job can be specified. Then several jobs are executed whose particular input file data size is approximately the data size specified in the for loop.

Both the number of files and the input data size for one job can be specified in the *Processed files per iteration* field of the for loop. Currently a value greater than 1024 is treated as data size, not any longer as number of files.

In this execution scenario the data size for the input was specified in both for loops to divide the input in nearly similar parts. Within these tests an average transfer time of 11 hours and 52 minutes is measured by using BFT (table 1). With using UFTP the average transfer time is 13 hours and 53 minutes, which is due to high transfer setup effort.

A positive aspect of UNICORE is the possibility to specify a file number limit or a data size limit per iteration in the for loop. Thus, the definition of the data size allows for a better distribution of the input amount. In this context it could be useful to determine the total input data size of all jobs at runtime. If, for example, in the for loop the input data size for the proceeding application would be known, it would be possible to equally assign the input to the jobs.

4 Conclusion

During the execution of several tests different advantages of and restrictions imposed by UNICORE were detected. One advantage is that in the for loop it is possible to iterate over a stated number of files or data size. UFTP or BFT can be used for file transfer, where UFTP offers faster transfer of big files. However the file transfer is also a big restriction, for example, when transferring small files, because the long transfer time raises the time to solution.

Hence, a necessary feature is a more efficient file transfer of unpacked files, which would reduce the transfer time and thereby also the time to solution. It would also be very helpful to avoid unnecessary file transfers, for example, between jobs that run on the same target system and especially in case one job uses the output of the preceding job. Another requested feature is that in the for loop the whole data size of the input for all subsequent jobs should be known, which would enable the equal distribution of the data volume among the iterations.

References

1. About UIMA-HPC, see `http://www.uima-hpc.de/en/about-uima-hpc.html`.
2. S. Bergmann, M. Romberg, A. Klenner, C. Janßen, T. Bathelt, G. Lonsdale: *UIMA-HPC - Application Support and Speed-up of Data Extraction Workflows through UNICORE*. Published in: eChallenges e-2012 Conference Proceedings, Paul Cunningham and Miriam Cunningham (Eds), IIMC International Information Management Corporation Ltd 2012, ISBN 978-1-905824-35-9.
3. Apache UIMA, see `http://uima.apache.org/index.html`.
4. About UIMA-HPC Technical, see `http://www.uima-hpc.de/en/technical.html`.
5. IBM Terminology, see `http://www-01.ibm.com/software/globalization/terminology/x.html#x3566550`.
6. UIMA Tutorial and Developers' Guides, Chapter 1. Annotator and Analysis Engine Developer's Guide, see `http://uima.apache.org/downloads/releaseDocs/2.2.2-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html#ugr.tug.aae`.

7. J. Fluck, H. T. Mevissen, H. Dach, M. Oster, M. Hofmann-Apitius:
ProMiner: Recognition of Human Gene and Protein Names using regularly updated Dictionaries. Published in: Proceedings of the Second BioCreative Challenge Evaluation Workshop, Lynette Hirschmann, Martin Krallinger and Alfonso Valencia (Eds), Centro Nacional de Investigaciones Oncológicas, Madrid, Spain, 2007, pp. 149-251.
8. Welcome to the Peregrine data-mining project,
see `https://trac.nbic.nl/data-mining/`.
9. Resource Description Framework (RDF), see `http://www.w3.org/RDF/`.
10. Brat rapid annotation tool,
see `http://brat.nlplab.org/introduction.html`.

The UNICORE Portal

Mariya Petrova[1], Valentina Huber[1], Bastian Demuth[1], Krzysztof Benedyczak[2], and Bernd Schuller[1]

[a]Jülich Supercomputing Centre,
Research Centre Jülich, 52425 Jülich, Germany
[b]Interdisciplinary Centre for Mathematical and Computational Modelling,
Warsaw University, Poland
E-mail: m.petrova@fz-juelich.de, v.huber@fz-juelich.de, golbi@icm.edu.pl,
b.schuller@fz-juelich.de

A dedicated web portal has been requested by user communities as an alternative interface to UNICORE resources for a long time. Compared to the traditional end-user clients such as the UNICORE Rich Client (URC) and UNICORE Command-line Client (UCC), a portal, accessed by a standard web browser, promises several advantages which are to be discussed in this paper. We also provide an overview of the portal development work: starting from the architecture and software design, we then describe various authentication possibilities, and end with demonstrating an example use case. Towards the end of the paper we present the current status of the UNICORE portal and the development road-map up to and beyond the first production release.

1 Introduction

The World Wide Web is nowadays the main means to enable access to data from diverse places around the world. People use wired or mobile Internet connections daily to check the online schedule of trains, publish photos in social networks, blog their opinions, talk to their friends and relatives, and many more. The Web grants almost unrestricted possibilities to share, use, communicate. As the essence of a Grid infrastructure is sharing and coordinated use of resources, distributed at different locations[1], we have started the development of a UNICORE web portal to provide a web-based graphical user interface. Applying web technologies has multiple advantages, for instance availability at any time from every part of the world. Moreover, installing a desktop client on an end user machine, as well as managing it, can turn out to be cumbersome and requires additional knowledge and time. The end users would need to handle administration and updates of the client on their own.

A web portal on the other side allows for easy software updates and version conflicts between client and server are avoided. Furthermore, authentication can be simplified, attempting to remove the error-prone handling of X.509 certificates. Besides, compared to more heavyweight Grid clients, the portal's user interface (UI) was simplified, thus becoming more intuitive and user-friendly. Simplification and high development efficiency have been achieved by using Vaadin[4] as the underlying web framework. Last but not least, access from mobile devices becomes much easier after deploying the SOAP-based UNICORE client code on a web server and leaving only the lightweight UI on the client machine.

2 UNICORE Guidelines

UNICORE is a Grid middleware with a history dating back over a decade. Its design guidelines have been established at the very beginning of its development in 1996 and have been followed ever since. These are, among others, seamlessness, security and intuitive access to distributed computing and data resources[2]. Security plays an important role in High Performance and Grid computing, so UNICORE grants access to a remote system only after users have successfully authenticated themselves with electronic certificates. The employed certificates comply to the X.509 standard. The UNICORE middleware supports not only job execution, but also data access and transfer from and to remote storages. In addition, UNICORE enables the execution of workflows where multiple tasks can be organized and executed in a predefined order[3]. Different development communities have been welcomed to contribute to the software ever since 2004 when UNICORE became open source[2].

3 Vaadin Framework

After a survey on available Java-based web application frameworks, we made our choice in favour of Vaadin. "Vaadin is a server-side AJAX web application development framework that enables developers to build high-quality user interfaces with Java"[4]. As the UNICORE client code is entirely written in Java, the chosen framework had to be Java-based, too. Vaadin's main benefits are flexibility, ease of integration, well written documentation, constant support, good reliability, and frequent updates. It is also an open source framework. Since the client frontend of Vaadin is entirely written in Javascript, there is no need for additional browser plug-ins[5].

4 Architecture of the Portal

The portal is characterised by a layered architecture consisting of different interdependent modules (see Figure 1). It has a clear separation of core logic and user interface components. Such a partition simplifies the design and enhances future re-usability of the code.

The ***core*** module forms the foundation of the pyramidal architecture. It is completely independent of the other modules and is responsible for basic functions such as logging, event management, common services, background thread handling, global access to preferences and other vital features. It maintains session data and user settings and implements the basic mechanisms for providing extension points as discussed in section 5.

The next architectural layer is formed by the ***ui*** and ***grid.core*** modules. The UI is based entirely on Vaadin (section 3) but some of the layouts and component styles have been customized. The ***ui*** module implements commonly used UI elements, e.g. menu navigation, common views like home and profile page, re-usable panels and user dialogues. The module provides a foundation for the other UI-related modules of the portal (i.e. ***authentication***, ***grid.ui***, ***workflow.ui***). Both the basic ***core*** and ***ui*** modules are free of Grid and UNICORE specific code which makes them a re-usable fundamentals for any other web application.

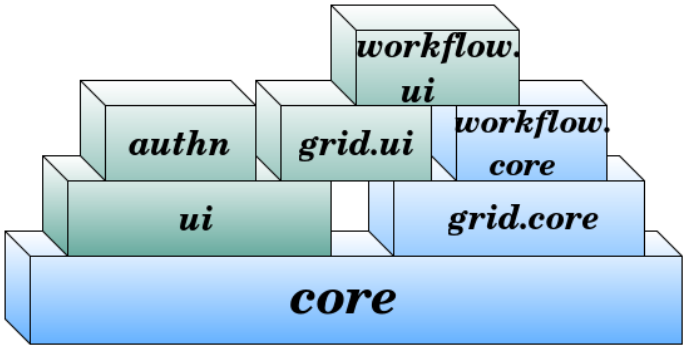


Figure 1: The picture illustrates the layered modular structure of the web portal as well as its interdependencies. The most basic and only independent module is the ***core***. Modules providing domain logic are marked in blue and UI modules are depicted in green. The authentication (***authn***) module combines both domain logic and UI code in a single lightweight code base.

The next module, the ***grid.core***, as the name suggests, extends the functionality of the ***core*** module into the Grid world and UNICORE. The module communicates with the UNICORE server and deals with job submission and monitoring, filtering and management of Grid resources, as well as Grid related extensions and actions. The overall architectural style is very similar to the one followed in the URC, but the separation of domain logic and UI code is improved.

Currently, there are no Grid application specific user interfaces, which are called "GridBeans" in the URC world, but the Generic GridBean has been ported from the URC to the portal[6]. Similar to the Generic GridBean in the URC, the input parameter panel for job preparation is built dynamically based on the application metadata. These metadata are published by the UNICORE server and include properties such as name, type, short description, and valid value ranges for each parameter[3]. Furthermore, the portal supports stage-in and stage-out of individual files or file sets. Each file staging definition can be created or edited on the fly through the file editor provided by the ***grid.ui*** module. Common and site-specific resources can also be requested for job execution. The status of each job is monitored after submission and forwarded to the ***grid.ui*** module which summarizes the results in a table. Thus, the user can conveniently view detailed information about any job or browse through its working directory. Moreover, he/she can check the availability of sites or the whole Grid infrastructure with registries, storages, jobs, workflows, etc. An additional powerful data manager assists the user in data-related tasks. It enables transfer, upload, or download of data within different locations in a user interface in the look & feel of the well-known Norton Commander[10] as shown on Figure 2. High throughput protocols such as UFTP are also supported.

The current architecture includes two workflow modules, ***workflow.core*** and ***workflow.ui*** further extending Grid functionality. Workflows are often a necessity for scientists to perform complex computations, as different software tools need to be chained to obtain a final result. Based on this requirement, the UNICORE web portal aims for providing

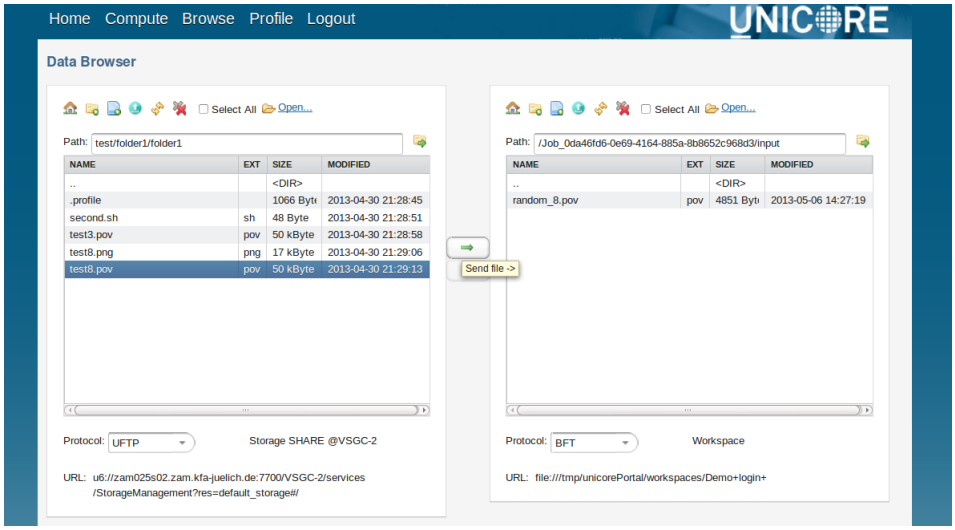


Figure 2: The data manager.

a full set of features including an easy-to-use workflow editor. The current development covers only basic features but further effort will be invested in this field. At the moment of writing, it is possible to create and execute basic workflows without complex relationships between the contained tasks. In future versions, more advanced possibilities such as the execution of workflows from pre-defined templates will be supported. It is interesting to note that the UI workflow editor has been developed with the help of the jsPlumb Javascript library. The library provides the possibility to establish visual connections between existing UI components[12].

Last but not least, let us look briefly into the authentication (***authn***) module. This module is rather lightweight and has therefore not been split up into UI and domain logic sub-modules. The authentication is performed by a separate servlet, currently providing three different options. More about authentication and security of the portal can be found in section 8.

5 Extension Points

For full flexibility the portal allows extensions through declarative XML-based interfaces, a mechanism somewhat similar to XML configuration in the Spring framework[9] or extension points in the Eclipse Rich Client[8] framework but with simpler and more intuitive syntax. Extension points for each module can be configured in a portalPlugin.xml file. Developers can add their own components and customize the main menu toolbar, the short-cut buttons, the context menus of the application, the logos in the footer, etc. Just like in the URC, in the portal it is possible to create new Grid services or modify the client's security settings without changing any existing code[3]. With minimal effort one can successfully add the desired new functionality.

```
<commandExtension>
    <id>eu.unicore.portal.grid.ui.commands.DeleteFileNodeCommand</id>
    <commandId>eu.unicore.portal.grid.ui.commands.DeleteFileNodeCommand</commandId>
    <commandClass>eu.unicore.portal.grid.ui.commands.DeleteFileNodeCommand</commandClass>
</commandExtension>

<menuExtension>
    <id>eu.unicore.portal.grid.ui.deleteFileNode</id>
    <name>Delete</name>
    <description>Delete</description>
    <icon>vfsaction/delete</icon>
    <!-- This menu item should be available in any menu -->
    <menuPath>.*</menuPath>
    <category>/delete</category>
    <visibleWhen>
        <selection>
            <instanceof>eu.unicore.portal.grid.core.nodes.u6.AbstractFileNode</instanceof>
        </selection>
    </visibleWhen>
    <commandDescriptor>
        <commandId>eu.unicore.portal.grid.ui.commands.DeleteFileNodeCommand</commandId>
    </commandDescriptor>
</menuExtension>
```

Figure 3: Delete extension.

For example (see Figure 3), to add a new delete command in a context menu, one first needs to implement a Java class that defines the behaviour of the command. After that, a commandExtension element must be added to the portalPlugin.xml file specifying the fully qualified class name and an internal command identifier for the new class named DeleteFileNodeCommand on the Figure. Next, a menuExtension element must be added to the portalPlugin.xml in order to add the delete command to the context menu of a certain Grid resource type (the AbstractFileNode). The menuExtension element fully defines the behaviour of the new menu extension. After having added these two XML elements to the configuration file, the delete menu item will be available in the context menu, whenever one or more AbstractFileNodes have been selected and it will act as desired. The menu extension point is quite flexible, the developer can declaratively restrict which menus should contain the new delete item (by entering a menuPath), set a customized icon, or set a visibleWhen condition. In the example, deletion is available to all menus but it will only be visible and available if the object is of type AbstractFileNode. This means that the given command is only bound to Grid file resources, it will not attempt to remove Grid jobs, storages or services registries from the UI views.

6 User Workspace

Another typical characteristic of the portal is the so called „user workspace“ which allows users to store data directly in the portal. With the help of the data manager described above, users are presented with a comfortable way to transfer from and to, upload into, or download files from their workspace. The data from the submitted jobs are being preserved in the workspace to be accessible from anywhere on the web.

7 Internationalization

Last but not least, the portal supports internationalization. Currently we have translated the user interface into English, German, Russian, and Polish. Through its extensible design the portal enables the translation of the whole UI into any other language without any programming effort. Developers would only have to translate the messages from the so-called message properties files of the application (this is the default mechanism for localizing Java applications). The portal code will take care of the rest.

8 Security and Authentication

One of the main added values provided by the portal is a better end-user experience with respect to getting access to Grid resources. Therefore, a main requirement is a simple user registration and authentication process. On the other hand, the portal security architecture needs to be flexible, in order to be able to accommodate many different options for authentication and general security. Similar to any other UNICORE client, the portal has its own X.509 credential. To allow the portal to act on behalf of the user, a mechanism commonly referred to as *explicit trust delegation* in the UNICORE world, the portal needs a SAML trust delegation assertion. In the security model used in UNICORE 6[7], the delegation assertion must be signed by the user certificate. This functionality is provided by a Java WebStart application that can be launched from the portal. We have decided to split the two functions *authentication* and *user registration*, and provide pluggable implementations for both, which can be configured flexibly in the portal. In the following, various authentication and registration options are described.

8.1 X.509 authentication

In the standard Grid security model the user possesses his/her own X.509 credential, which can be imported into the web browser. This X.509 credential can be used to authenticate the user. From the security point of view this can be considered the strongest form of authentication. To allow the portal to act on behalf of the user, the portal needs a SAML *explicit trust delegation* assertion signed by the user certificate. This function is provided by a Java WebStart application that can be launched from the portal.

8.2 Username and password based account

Another option is to authenticate the user with a password. In this case, the user's certificate is not required to be available in the web browser, allowing to log-in from other devices and locations such as an internet café. Still, the user must use his X.509 certificate at least once to generate the required trust delegation assertion before subsequently logging in via username/password.

8.3 Federated account

In the upcoming UNICORE 7 release, it will be possible to have users without end user certificates. They will only be authenticated by SAML assertions issued by a new service

called Unity. The portal can delegate the user login to this new service. Unity again offers many authentication options, which cannot be fully described here. We merely note that apart from the new no-certificate case, also other scenarios will be supported by Unity.

8.4 Other options

Depending on user community and deployment needs, other authentication options may be implemented, such as logging into Kerberos, LDAP, or even MyProxy.

9 Use Case

For better presentation and understanding of the portal, we would like to guide the user through a simple job submission.

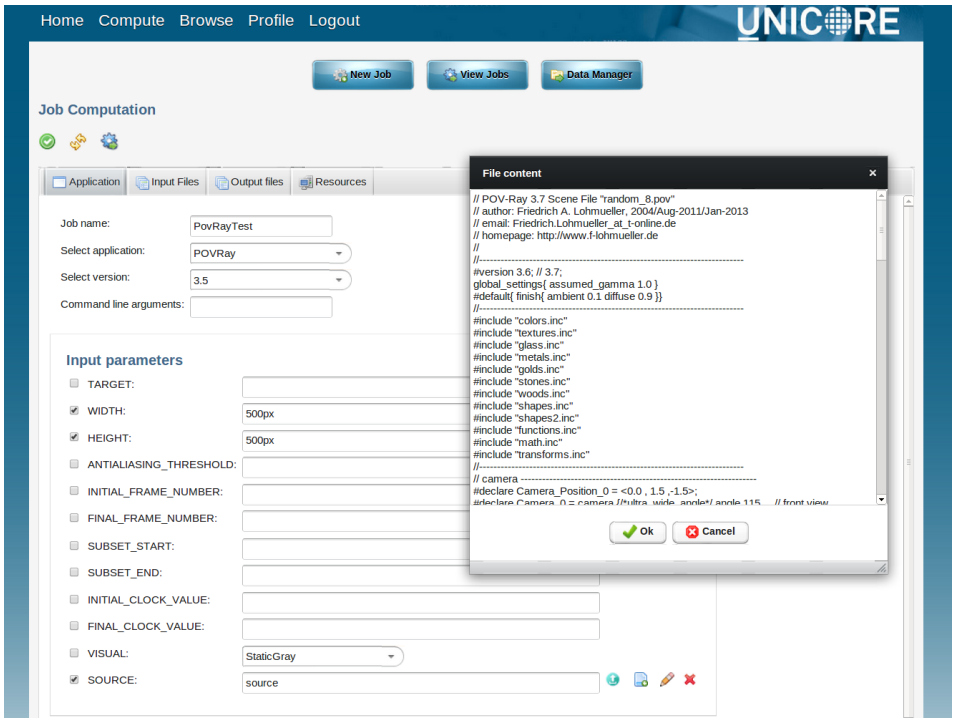
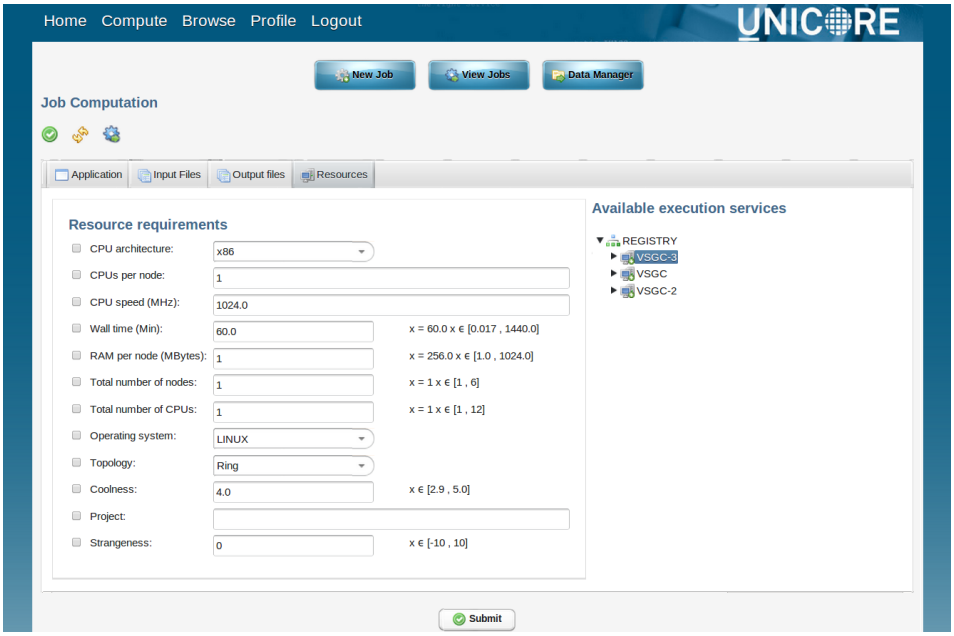
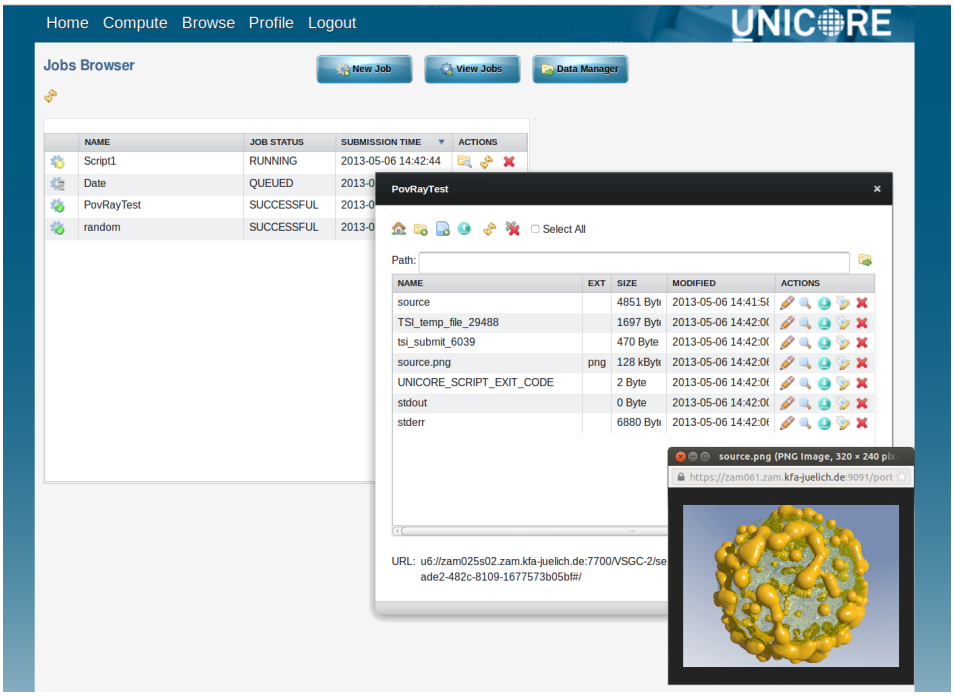


Figure 4: Definition of a new job.

We can either navigate through the menu options of the website in Compute → New Job (please refer to Figure 4) or directly use the New Job button to reach the job preparation view. For our example we have chosen the POVRay application, a free ray-tracing program that renders high-quality three-dimensional images through a set of commands[11].



(a) Choice of common or site-specific resources to be requested for the job.



(b) View of the job results.

Figure 5: Job submission.

We named our job PovRayTest, set the preferred size for the resulting image, and uploaded a source file to generate the result. Figure 4 illustrates the preparation and shows the source file that has been opened in the file editor.

Next, we can switch to the Input Files tab, where we can import or create additional files if necessary (e.g. for providing textures for 3D objects in the PovRay scene), or we can continue to the Output Files tab, where stagings of output files to any available UNICORE storage can be declared.

At the Resources tab we can select and define various resources required for our job. We can also point a site to which we wish to submit the job. In the case of Figure 5a we have chosen submission on our test-site VSGC-3. The last four resources from the example are site-specific to VSGC-3. It should be noted that these steps are optional for the job being submitted.

The job can be submitted through the green submit button which is available twice: once at the bottom and once as a short-cut on the up-leftmost corner.

Job monitoring starts immediately after submission. After job completion we will receive a notification. In order to view the result we can either refer to Browse $\rightarrow$ Jobs from the menu or use the short-cut View Jobs. In the table shown in Figure 5b we can observe the status of the job and check more detailed information by left-clicking the job with the mouse. Browsing the job working directory is achieved by clicking the button depicting a folder and a spying glass. As shown in the example, we can see all files in the working directory, e.g. the output and error streams, a file containing the process exit code, as well as the resulting image called source.png generated by the POVRay application. A double-click or a click on the spying glass button will open a new window displaying the image.

10 Conclusion and Future Plans

The UNICORE portal is a web application that aims for presenting a dedicated and user friendly graphical UI for working with a UNICORE Grid. It has several advantages compared to the UNICORE desktop clients, mainly it being easily accessed through all known internet browsers. Its modular architecture promises an extensible and flexible framework that can be customized for different Grid infrastructures with minimal effort. The internationalization of the user interface offers an additional convenience.

The development of the portal is still ongoing. The first production release is expected in the autumn of 2013. It will fully support the preparation, submission, and monitoring of generic jobs and custom executables; basic creation, editing and submission of simple workflows; Grid, jobs, and site browsers; a powerful data manager for working with local and remote data. The portal will also provide different ways of authentication including certificate and certificate-less options as well as hierarchy and administration of user accounts in Unity. We also plan to provide basic social features like message boxes for users belonging to the same group and event lists depending on user settings. Throwing a look beyond the first release, we are planning to expand the events list into a fully-fledged history component, which will allow filtration and restriction of entries in accordance with users' preferences or security concerns.

In the future we will also include resource reservation and we will improve the existing components by supporting advanced UNICORE functions like metadata management and resource sharing.

To conclude, we encourage the users to share their requirements, suggestions and needs as our main target is to introduce a simple, secure, and usable web application for communication with a UNICORE Grid.

References

1. Foster, I., Kesselman, C., Nick, J. M., and Tuecke, S., "The physiology of the grid", Grid computing: making the global infrastructure a reality, pp. 217-249, 2013
2. Streit, A., Bała, P., Beck-Ratzka, A., Benedyczak, K., Bergmann, S., Breu, R., Ziegler, W., et al., "UNICORE 6 - recent and future advancements", Annals of telecommunications-annales des télécommunications, vol. 65, no. 11-12, pp. 757-762, 2010
3. Demuth, B., Schuller, B., Holl, S., Daivandy, J., Giesler, A., Huber, V., and Sild, S., "The UNICORE Rich Client: Facilitating the automated execution of scientific workflows", IEEE Sixth International Conference on e-Science, pp. 238-245, 2010
4. Gronross, M., "Book of Vaadin: Vaadin 6.4", Turku, Finland: Vadin Ltd., 2010
5. Holan, J. and Kvasnovský;, O., "Vaadin 7 Cookbook", Packt Publishing, 2013
6. Ratering, R., Lukichev, A., Riedel, M., Mallmann, D., Vanni, A., Cacciari, C., Ohme, G., et al., "Gridbeans: Support e-science and grid applications.", IEEE Second International Conference on e-Science, pp. 45-45, 2006
7. Benedyczak, K., Bała, P., van den Berghe, S., Menday, R., and Schuller, B., "Key aspects of the UNICORE 6 security model", Future Generation Computer Systems, vol. 27, no. 2, pp. 195-201, 2011
8. McAffer, J., Lemieux, J. M., and Aniszczyk, C., "Eclipse Rich Client Platform", Addison-Wesley Professional, 2010
9. Spring Framework, `http://www.springsource.org/spring-frameworks`, accessed: 27.08.2013
10. Norton Commander, `http://www.softpanorama.org/OFM/Paradigm/Ch03/norton_commander.shtml`, accessed: 27.08.2013
11. POVRay, `http://www.povray.org/`, accessed: 27.08.2013
12. jsPlumb, `http://jsplumbtoolkit.com/`, accessed: 27.08.2013

Providing a Web Portal for Development and Utilization of Distributed Virtual Test Beds on the basis of UNICORE Grid infrastructure

Gleb Radchenko[1], **Elena Hudyakova**[1], **and Evgeniy Zakharov**[1]

Supercomputer Simulation Laboratory,
South Ural State University, Chelyabinsk, Russia
E-mail: radchenko@acm.org

The Distributed Virtual Test Bed (DiVTB) technology provides a solution for the integration of problem-oriented systems in distributed computing environments. DiVTB provides a problem-oriented user interface to distributed computing resources within the grid, online launch of virtual experiments, automated search, monitoring and allocation of computing resources for carrying out the virtual experiments. The most important task of the DiVTB system is to provide a user-friendly interface to create and use DiVTB.

This paper describes the architecture of Web systems that provide a direct interaction of the DiVTB platform with end users: DiVTB Developer and DiVTB Portal. DiVTB Developer is a Web-based integrated development environment of DiVTB. DiVTB Portal provides a problem-oriented user interface for management and execution of virtual experiments in a distributed computing environment.

1 Introduction

Web-portals increasingly become a major way to provide interfaces to distributed computing systems. Using the Web-based applications one can significantly simplify access to a distributed computing environment for end users. First, the user is not required to install and configure desktop applications to access remote systems. Second, the use of Web technologies such as HTML5 can significantly extend the range of devices from which you can access the grid systems.

Let us consider most widespread Web-portal systems. The Vine Toolkit[1] was developed to provide a Web platform for different scientific portals. The main purpose of the Vine Toolkit is to provide a convenient way to gain access to remote supercomputer resources and grid systems. P-GRADE Portal[2] is a Web portal focused on management of workflow jobs in grid environments based on Globus Toolkit platform[3]. It includes the tools necessary to create, execute and monitor workflows. For requirements of EGEE (Enabling Grids for E-Science in Europe)[4] and the WLCG (The Worldwide LHC Computing Grid)[5] projects the CIC Portal[6] (now known as Central Operations Portal) was developed. This portal includes features such as computing environment load monitoring, user management and virtual organizations support, allocation of computing resources among tasks, etc.

After analysing the existing solutions as well as customer feedback, we identified the following requirements for Web clients for the Distributed Virtual Test Bed System[7]:

- a Web-client should be developed using pure Web technologies (such as HTML5), without the need for additional plug-ins such as Java, Flash, Silverlight etc.;
- an interface for development (IDE) and for using of distributed applications should be separated, providing end users with transparent problem-oriented interface to the grid environment;
- a convenient visual mechanism for workflow descriptions should be provided;
- a convenient mechanism for parameterization of the source files will allow the most seamless integration of classical batch applications in the grid environment;
- automatic generation of the user interface based on the description of the parameters of distributed application should be provided.

The paper is organized as follows: the next section presents an overview of the Distributed Virtual Test Bed (DiVTB) project, including an architecture of DiVTB System software solution. Section 3 describes architecture and features of the DiVTB Developer Web-IDE. In Section 4 the DiVTB Portal Web-application is presented. And finally, in Section 5 possible future extensions and enhancements are discussed.

2 DiVTB System

The principal objective of the Distributed Virtual Test Bed project - is to develop a technology allowing to take into account the specifics of the problem-oriented subject areas while providing the resources of distributed computing environments. This technology aims to create "intelligent" middleware providing users with easy, transparent and secure access to distributed computing resources and allowing them to solve specific classes of applied problems.

The *Distributed Virtual Test Bed (DiVTB)* technology provides a solution for the integration of problem-oriented systems in distributed computing environments[8]. DiVTB provides a problem-oriented user interface to distributed computing resources within the grid, online launch of virtual experiments, automated search, monitoring and allocation of computing resources for carrying out the virtual experiments.

To implement the DiVTB Technology we created a *DiVTB System* - a UNICORE-based software solution for DiVTB development and implementation. We selected the UNICORE 6 grid computing middleware[9] as a platform for implementation of grid services. The GridBeans approach included in UNICORE 6 supports the transparent integration of legacy standalone applications as grid services in the grid environment.

Within the DiVTB System (Fig. 1) we distinguish two levels of the components: "driver" level and "client" level. The main representatives of the driver level are *DiVTB Server* and *DiVTB Broker*[10] components. They provide execution of virtual experiments in distributed computing environment and usage of UNICORE grid resources.

The "client" layer provides an interaction of developers and end-users with the DiVTB System. We decided to separate the process of development and use of a virtual test bed.

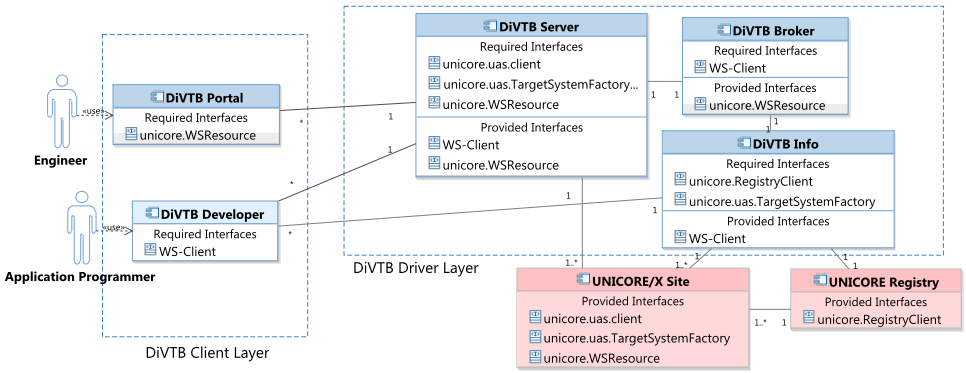


Figure 1: Components of the DiVTB System

Accordingly, we developed two Web applications that provide development and utilization of Distributed Virtual Test Beds:

- *DiVTB Developer* - a Web IDE that provides the development process of distributed virtual test beds.
- *DiVTB Portal* - a Web application that provides a problem-oriented user interface for management and execution of virtual experiments in a distributed computing environment.

3 DiVTB Developer

The DiVTB Developer Web IDE provides the development of DiVTB, including visual design of workflows, parameterization of the source files, definition and description of parameters of the virtual experiment.

The application programmer designs the virtual test bed by describing the workflow. A workflow consists of actions, which define stages of virtual experiment. Each action is implemented by computing services, located in the UNICORE grid environment. DiVTB Developer receives information about the available computing services via DiVTB Info component, which periodically polls the available grid nodes and receives information about the interfaces of UNICORE Applications installed on them.

The developer can import and export the DiVTB Project as an archive which consists of XML-based project files and a set of additional source files. In addition, the developer can export the DiVTB Project to DiVTB Server component, which provides storage and execution of DiVTB.

3.1 DiVTB Developer Client

The DiVTB Developer IDE is implemented as a client-server Web-based application (Fig. 2). The server part provides storage and handling of DiVTB projects. The client part provides a visual interface for workflow creation and customization.

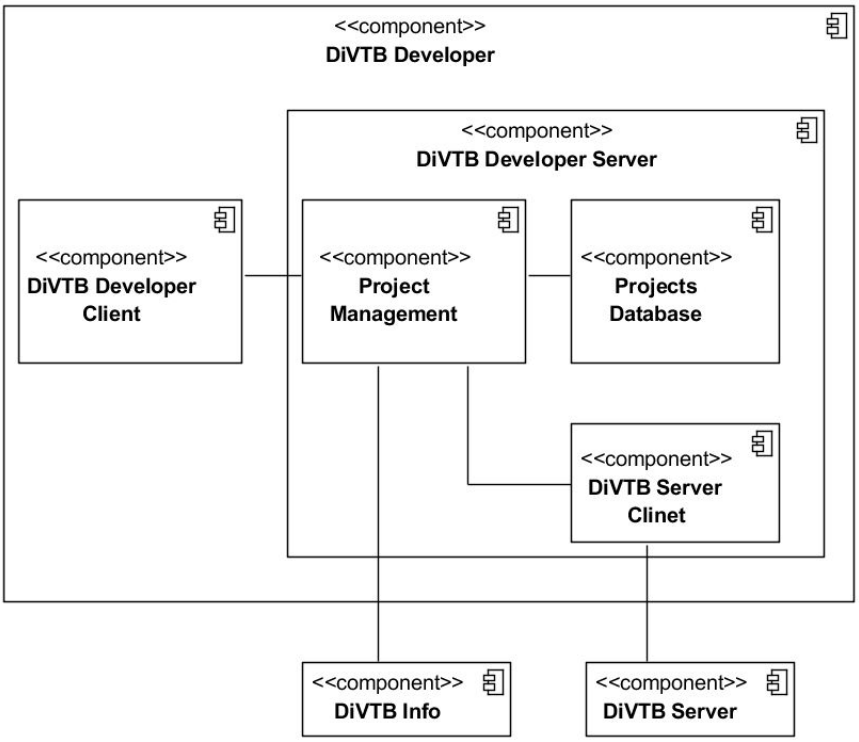


Figure 2: Architecture of the DiVTB Developer system

The project management component processes and stores DiVTB projects in the project database. It interacts also with the DiVTB Info service to get the information about services available in the UNICORE Grid, including parameters of each service. The DiVTB Server interaction component allows the developer to upload the DiVTB project to the DiVTB Server, which provides execution of the virtual experiment in the UNICORE grid environment.

3.2 DiVTB Developer Interface

The user interface of the DiVTB Developer IDE can be divided into three logical parts: workflow description module, action description module, source file editing and parameterization module.

The *workflow description module* provides the DiVTB workflow design (Fig. 3). The developer can build a workflow using several node types (action nodes and control nodes), connected by edges which describe the control flow. In table 1 you can find available types of workflow nodes, including their graphical representation and semantics.

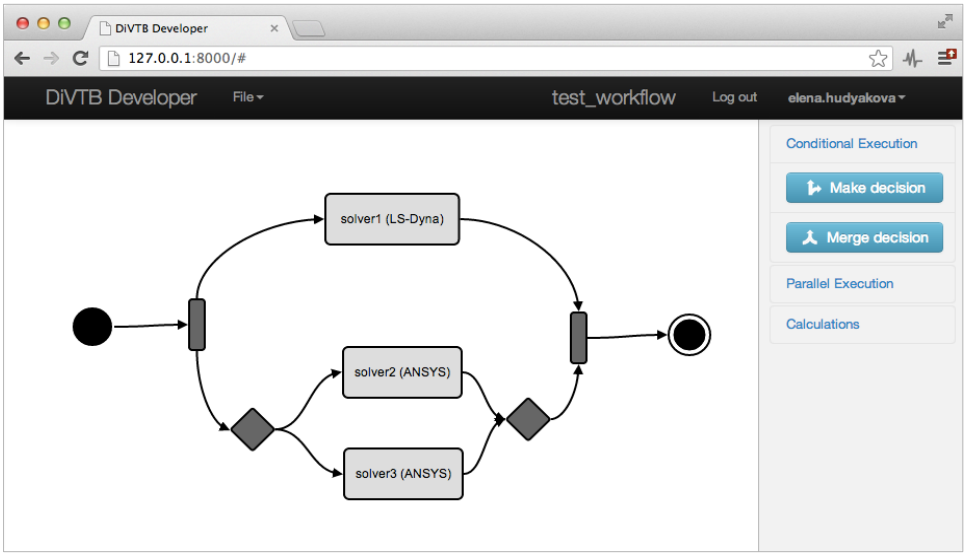


Figure 3: The workflow description module

In the *action description module* developer can specify the following parameters of the node:

- a name and a description of the semantics of action node;
- a basic service of an action (developer can choose a service from the list of existing services);
- values of system parameters (start-up options for a service);
- a list of problem parameters in terms of the relevant problem domain;
- names or name masks of input source files that are to be uploaded/downloaded from the grid service before/after the calculations execution;
- a list of input files for a service.

The *source file editing and parameterization module* provides an interface for editing DiVTB input files (Fig. 4). These files describe the simulated problem in a format that is appropriate to the end computing services. The developer can edit any source file and insert a special template code into the source code using problem or system parameters of the DiVTB as variables. The parameterized source files together with values of DiVTB parameters would be sent to the DiVTB Server during virtual experiment initialization. There, the parameterized templates would be transformed to the source files for the corresponding UNICORE applications.

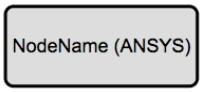
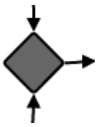
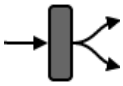
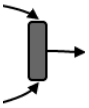
Name	Graphical representation	Semantics
Initial node		A control node that starts the workflow.
Final node		A control node that indicates the end of the workflow
Action node		The Action node provides a specific action of a virtual experiment on the basis of particular service, available in the UNICORE Grid.
Decision node		A control node that provides a choice of a route of a control flow depending on the value of a Boolean expression (the `WHEN` field of a control flow branch).
Merge node		A control node, which merges two or more alternative control flow branches.
Fork node		A control node that initiates the execution of multiple concurrent control flows.
Join node		A control node that synchronizes several parallel control flows.

Table 1: Workflow node types

We use Java Minimal Template Engine (JMTE)[11] as a template engine for DiVTB. JMTE is a template system that allows developer to embed variables in the template text, using a simple syntax that supports basic operations with variables (like loops, conditions etc.). Thus, creating a problem-oriented file once for a specific task, the developer can describe a DiVTB for a whole class of similar problems.

4 DiVTB Portal

4.1 DiVTB Portal Architecture

The main purpose of the DiVTB Portal is to provide a problem-oriented user interface for management and execution of virtual experiments in a distributed computing environment. The engineer interacts with a Web-interface of the DiVTB Portal. It provides a list of test beds available to the user and allows the user to setup and run the virtual experiment. It also responsible for authentication and management of DiVTB System users.

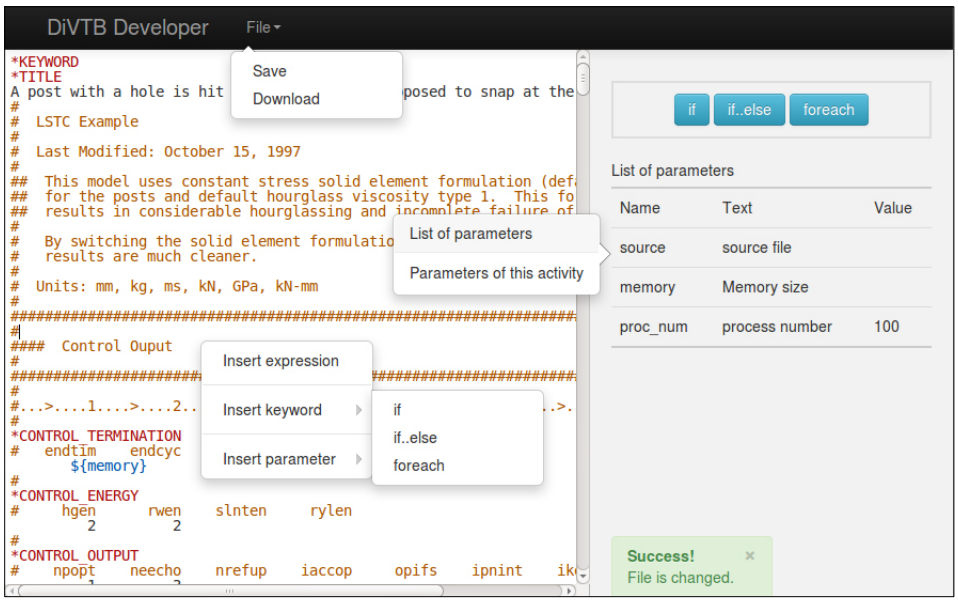


Figure 4: The source file parameterization module

The DiVTB Portal consists of five components (Fig. 5):

- Test Beds Manager
- Web Forms Generator
- Client for DiVTB Server
- Virtual Experiments Manager
- Local Storage

Test Beds Manager provides management of test beds, including import of test beds from the DiVTB Server and distribution of test beds between the users of the DiVTB System.

Web Forms Generator provides automatic generation of DiVTB Web-interface on the basis of parameters of DiVTB. Using this interface, a user can specify values of parameters of DiVTB and submit the virtual experiment into the UNICORE grid environment.

Client for DiVTB Server provides interaction with DiVTB Server service, including download of available DiVTB, virtual experiments submission and download of simulation results.

Virtual Experiments Manager provides functions for virtual experiments execution, including updates of status of workflow execution and retrieval of results.

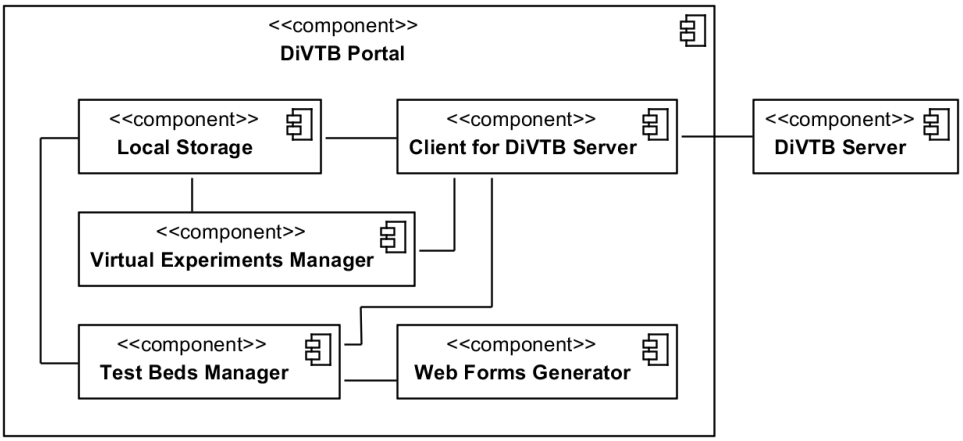


Figure 5: Architecture of the DiVTB Portal system

4.2 Management of DiVTB

All the available test beds are stored in the project storage of the DiVTB Server. Each test bed has a unique identifier (the `UID`). Based on the `UID`, DiVTB Portal can determine presence or absence of the certain test bed in its local database. To maintain the database of distributed virtual test beds, DiVTB Portal uses Virtual Experiments Manager which provides import of test beds from the DiVTB Server storage.

During the test beds import process, DiVTB Portal requests from the DiVTB Server a list of the UIDs of all available DiVTB using the `getProjectIDs` method. Next, DiVTB portal checks presence or absence for each identifier in its own database.

If the DiVTB is absent in the DiVTB Portal's database, it requests the description of DiVTB interface and parameters of the virtual experiment by calling `getProblemCaebean` method from the DiVTB Server. After retrieval, the DiVTB can be provided for the users of DiVTB Portal.

In case if the test bed is presented in the DiVTB Portal database but unavailable in DiVTB Server, its status changes to "Not Active". The user of the DiVTB Portal cannot create a virtual experiment based on the test bed with such status. However, he can view parameters and download results of previous experiments based on such test bed.

All test beds in the DiVTB Portal are divided into two groups: test beds, imported into DiVTB Portal, and test beds that are available to the user of the portal. Each user can ask from an administrator of a DiVTB System for a subscription to any available test bed.

A user can create a virtual experiment based on any DiVTB on which he signed. A user also can unsubscribe from a DiVTB or view a list of his/her virtual experiments.

4.3 Generation and Execution of Virtual Experiment

Each DiVTB consists of groups of parameters. Each group is a union of semantically related parameters of the experiment. Parameter names should be unique within the group. For each group, the Web Forms Generator creates a container.

For each parameter inside the group, the Web Forms Generator creates an HTML-form, corresponding to the type of the parameter (table 2). Using DiVTB Developer, an application programmer can limit the variety of definitions of any parameter (except the "File" parameter) by listing the possible values of the parameter. In this case Web Forms Generator will provide to the user a drop down list with the possible values of the parameter.

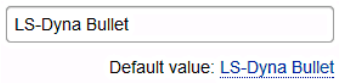
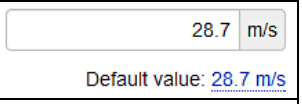
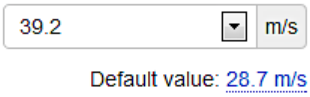
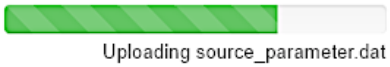
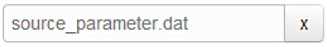
Parameter type	Interface item
String	
Integer, Float, Double, …	
Integer, Float and Double with constraints (enum)	
Input file (select)	
Input file (uploading)	
Input file (uploaded)	

Table 2: Mapping DiVTB parameters and corresponding Web forms

Web Forms Generator of the DiVTB Portal allows users to set parameters of the test beds, by specifying their values in a simple HTML-form. The parameters in this case may have a different type (e.g. String, Integer, Float, File and etc.). If the parameter type is "File", the user needs to select the desired file through the file selection dialog. Selected files are loaded asynchronously into the DiVTB Portal and temporarily stored before the job execution. Figure 6 shows an example of the generated interface for specifying parameter values.

Before submission, the values of all parameters (including files) are validated by the DiVTB Portal. If the value of a parameter does not match the domain of its possible values, the user will be prompted to fix the value of the corresponding parameter. After successful validation, DiVTB Portal calls a `CreateInstance` method of the DiVTB Server and creates an instance of an experiment. Next, all source files are uploaded from DiVTB Portal to DiVTB Server by the `uploadSourceFiles` method. Finally, the `submitJob` method provides a transfer of the parameters of the virtual experiment to the DiVTB Server in a context of a previously created instance. Figure 7 shows the process of setting up and successful completion of the virtual experiment.

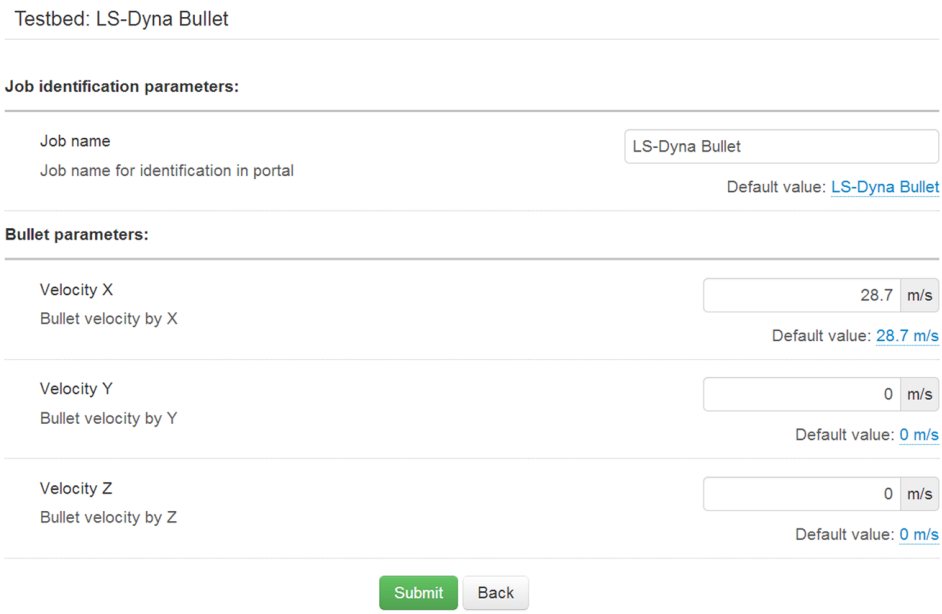


Figure 6: Example of interface for virtual experiment definition for the "LS-Dyna Bullet" virtual test bed

There are five possible states of virtual experiments:

- `NOT_STARTED` - the initial state. The corresponding job is not running.
- `RUNNING` - intermediate state. The virtual experiment was started by the `submitJob` method.
- `HELD` - intermediate state. The virtual experiment is suspended for any reason.
- `SUCCESSFULL` - final state. Successful completion of the virtual experiment.
- `FAILED` - final state. The virtual experiment ended with an error.

When a user requests the results of a virtual experiment being in the intermediate state, the DiVTB Portal uses the `getStatus` method to get currently available intermediate results. When the experiments status change from an initial or intermediate to the final state, DiVTB Portal obtains the final results by means of `getStatus` method and calls the `getExecTime` method to get the duration of the conducted virtual experiment.

When the user deletes a virtual experiment in `SUCCESSFULL` state, the DiVTB Portal sends a `removeResults` request to the DiVTB Server. This query deletes all the results of the virtual experiment stored on the DiVTB Server. If the user wants to stop and delete a virtual experiment in `RUNNING`, `HELD` or `FAILED` state, the DiVTB Portal calls `removeWorkDir` method, that stops the execution of all active actions of the workflow and deletes all intermediate results from the DiVTB Server.

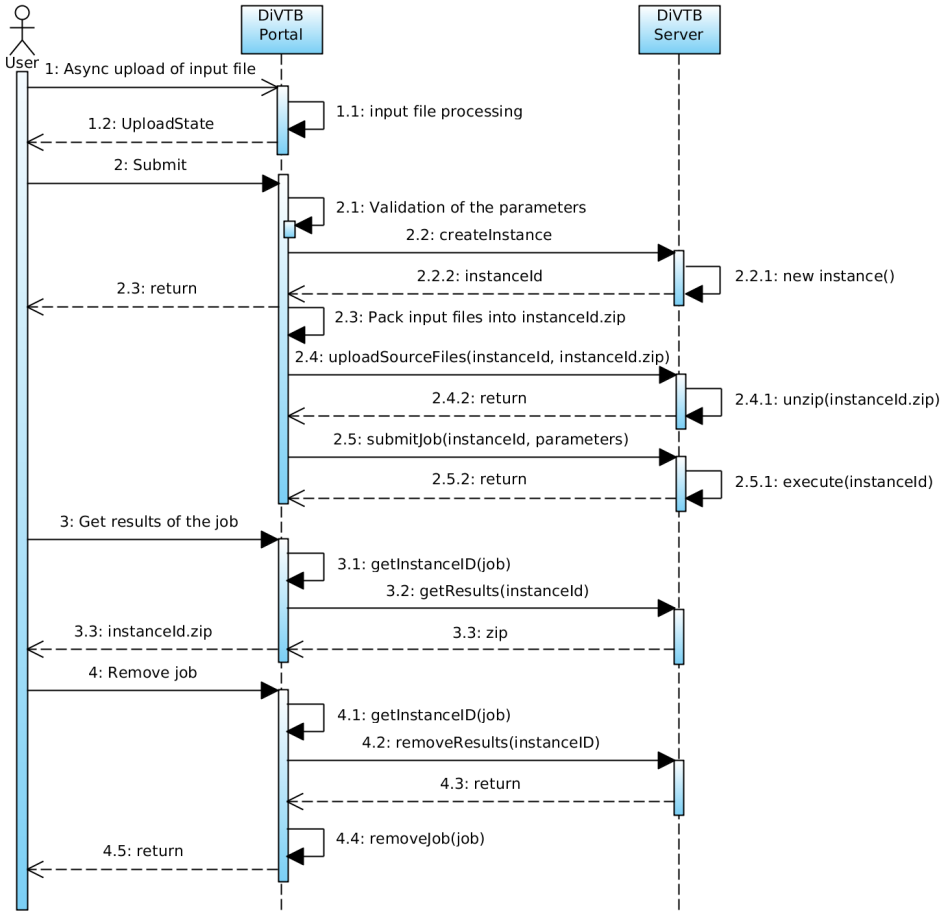


Figure 7: A Sequence diagram of virtual experiment execution

4.4 DiVTB Portal File Transfer Tests

To test DiVTB Portal file transfer speed and compare it with alternative ways of files transfer (such as SSH and UNICORE Rich Client file transfer) we deployed DiVTB System on computing resources of Supercomputing Simulation Laboratory of South-Ural State University. For testing, we used the client PC from the external network. We have prepared three test files, size of 1, 2, and 5 GB respectively. Each file has been downloaded and uploaded from the node, where the DiVTB Portal was located by means of SSH, UNICORE Rich Client and DiVTB Portal systems. To take into account temporal anomalies in the network, each download and upload procedure was carried out for 5 times. Average values of measurements made are shown in Tables 3 and 4. As can be seen from the test results, the rate of data transfer to/from the end user through the DiVTB Portal corresponds to the existing common methods of data transmission.

	SSH		DiVTB Portal		UNICORE Rich Client	
File Size, MB	Time, s.	Speed, Mbit/s	Time, s.	Speed, Mbit/s	Time, s.	Speed, Mbit/s
1024	459	17.85	456	17.96	490	16.71
2048	918	17.85	949	17.25	1080	15.17
5120	2293	17.86	2323	17.63	2637	15.53

Table 3: Comparison of download time and speed (mean values)

	SSH		DiVTB Portal		UNICORE Rich Client	
File Size, MB	Time, s.	Speed, Mbit/s	Time, s.	Speed, Mbit/s	Time, s.	Speed, Mbit/s
1024	156	52.51	300	27.31	438	18.70
2048	306	53.54	684	23.95	810	20.23
5120	720	56.89	1512	27.09	1644	24.91

Table 4: Comparison of upload time and speed (mean values)

5 Conclusion and Future Work

We presented an architecture and peculiarities of the DiVTB System's user-oriented Web-applications: DiVTB Developer and DiVTB Portal. The DiVTB Developer provides a Web-based IDE for the development of distributed virtual test beds, including design of a workflow of a DiVTB, description of problem-oriented parameters of simulation and parameterization of source files by means of templating engine.

The DiVTB Portal provides a problem-oriented user interface for management and execution of virtual experiments in a distributed computing environment, supporting automatic generation of Web-forms for virtual experiments definition on the basis of DiVTB parameters description and transfer of input and output files of virtual experiment from the DiVTB Server.

As future work, there are several areas in which we would like to continue the development of the DiVTB Web-applications. First, the experience of using the DiVTB Developer and DiVTB Portal systems has identified the need of workflows validation and visualization of workflows execution.

Second, users asked us to provide a possibility to visualize the results of their virtual experiments without the need of downloading them. A possible solution to this is to provide a remote interactive visualization system for the DiVTB Portal.

In addition, we would like to provide an integration of our authorization and authentication services with UNICORE services. This will significantly enhance the security of the DiVTB system and greatly expand its applicability to other UNICORE-based grids.

Acknowledgements

The reported study was partially supported by RFBR, research project No. 11-07-00478-a and by the Ministry of Education and Science of Russia, task No. 8.3786.2011

References

1. P. Dziubecki, et al.: *Online Web-based science gateway for nanotechnology research*, JournalLecture Notes in Computer Science7136205-2162012.
2. Cs. Nemeth, *The P-GRADE grid portal*, in: Computational Science and Its Applications (ICCSA 2004) International Conference, pp. 10-19, 2004.
3. `http://www.globus.org/toolkit/`
4. `http://www.egi.eu/`
5. `http://wlcg.web.cern.ch/`
6. O. Aidel et al.: *CIC Portal: a collaborative and scalable integration platform for high availability grid operations*, in Proc. of the 8th IEEE/ACM Int. Conf. on Grid Computing (GRID'07), pp. 121-128, 2007.
7. G. Radchenko, E. Hudyakova, *Distributed Virtual Test Bed: an Approach to Integration of CAE Systems in UNICORE Grid Environment*, in: MIPRO 2013 Proc. of the 36th International Convention, pp. 183-188, 2013.
8. G. Radchenko, E. Hudyakova, *A Service-Oriented Approach of Integration of Computer-Aided Engineering Systems in Distributed Computing Environments*, in: UNICORE Summit 2012 Proceedings, pp. 57-66, 2012.
9. A. Streit et al.: *UNICORE 6 - Recent and Future Advancements*, Annals of Telecommunications **65(11)**, 757–762, 2010.
10. A. Shamakina, *Brokering Service for Supporting Problem-Oriented Grid Environments*, in: UNICORE Summit 2012 Proceedings, pp. 67-75, 2012.
11. `http://code.google.com/p/jmte/`

The DiVTB Platform: Some Experience Gained in the Application of UNICORE as Middleware in the "Mobility of Young Scientists" project in Russia

**Anastasia Shamakina[1], Gleb Radchenko[1], Elena Khudyakova[1]
Evgeny Zakharov[1], Dmitry Savchenko[1], Kamila Koledina[2]
Dmitry Maryin[2], Igor Kulikov[3], Igor Chernykh[3], Michael Bakhterev[4]
Pavel Vasev[4], Alexey Mishchenko[5], Andrey Poluyanov[5]
Andrey Sozykin[6], Yury Kirienko[7], and Vladislav Shchapov[8]**

[1] South Ural State University, Russia

[2] Bashkir State University, Russia

[3] Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Russia

[4] Institute of Mathematics and Mechanics UB RAS, Russia

[5] Omsk Branch of Sobolev Institute of Mathematics SB RAS, Russia

[6] Ural Federal University named after the first President of Russia B.N.Yeltsin, Russia

[7] Far Eastern Federal University, Russia

[8] Perm National Research Polytechnic University, Russia

In this paper we discuss a number of results that were gained with the application of the UNICORE platform in a DiVTB system for creating virtual test beds. The DiVTB system provides a problem-oriented user interface to distributed computing resources within the grid, online launch of Computer-Aided Engineering (CAE) simulation, automated search, monitoring and allocation of computing resources for carrying out the virtual experiments. The studies were presented within the framework of the "Mobility of Young Scientists" project of The Federal Target Programme entitled as "Scientific and Scientific-Pedagogical Personnel of the Innovative Russia for the Years 2009-2013" in 2012.

1 Introduction

Nowadays a very promising direction is associated with the use of grid technologies [1] for solutions of demanding scientific problems in different fields of knowledge. The modern solution tools of applied tasks are feature-rich software systems composed of many individual software sub-systems with a complex user interface. Quite often solving one problem requires using two or more different software packages simultaneously and arranging components on the user interface, which demands special deep knowleadge. In addition, the execution of actual tasks using remote supercomputer resources can be difficult for an average user.

We can think of approaching a solution of the problems mentioned above through an integration of software packages into the distributed computing grid environment based on the concept of a distributed virtual test bed (DiVTB) [2]. Such a test bed is a software system that provides a solution for a specific class of applications in a distributed computing environment. Accordingly, grid environments like these are called *problem-oriented*.

The principal feature of the construction of problem-oriented environments is an ability to integrate software packages from different areas, based on a unified technology. The CAEBeans [3, 4] is an example of such a technology which serves for creating problem-oriented shells. It was developed in South Ural State University (Chelyabinsk, Russia). The software DiVTB system was created on the basis of this technology. The DiVTB system provides virtual application services that ensure start-up, simulation, visualisation, and analysis of remote modelling in grid environments.

The present article discusses the results of studies carried out on the basis of the DiVTB technology in the "Mobility of Young Scientists" project of the Federal Target Programme "Research and Scientific-Pedagogical Personnel of the Innovative Russia in the Years 2009-2013" in 2012. The subject of the research is "The Technology of Creating Problem-Oriented Services in Distributed Computing Environments".

2 Problem-oriented DiVTB Complex

The DiVTB (Distributed Virtual Test Bed) is a software system that ensures the development and operation of distributed virtual test beds. DiVTB provides a task-oriented approach for solving specific classes of problems in engineering design through resources supplied by grid computing environments [5].

The DiVTB system provides the following functionality:

- development of test beds;
- search for computing resources in grid environments;
- launch of test beds;
- monitoring and execution of test beds;
- transfer of results to a user;
- visualisation of results.

The DiVTB system includes the following components (Fig. 1):

1. *A DiVTB Developer*, which is a web-application that provides the development of DiVTBs for a grid environment. This gives an opportunity to an application programmer to visually design the workflow for simulation, to create a file template of a task statement, to generate a set of parameters of a simulated task. The system also enables a function to export DiVTBs to a DiVTB Server where they can be launched for task calculation.
2. *A DiVTB Portal*, which is a web-application for an engineer that provides starting-up and retrieving simulation results of virtual experiments. The DiVTB Portal ensures authentication, user account management and a user interface for virtual experiments through the DiVTB system. The DiVTB Portal includes a built-in web-form generator which automatically generates a user interface on the basis of a relevant DiVTB's parameter description.

3. *A DiVTB Server*, which is an environment for storing DiVTBs and managing virtual experiments. The DiVTB Server ensures the execution of a DiVTB workflow, including the formatting of a file of a task statement for each individual computing service in a grid environment on the basis of experimental parameters defined by an engineer, as well as the obtaining of simulation results and transfer of results to an engineer.

4. *A DiVTB Broker*, which implements automated registration, search and allocation of grid resources to perform the actions of engineering simulation. The DiVTB Broker maintains a permanent record of grid resources, which are available in the computing environment, and provides sets of computing resources on demand of the DiVTB Server for carrying out computational experiments [6].

5. *A DiVTB Viewer*, which is an auxiliary service for interactive visualisation of distributed virtual test beds. The DiVTB Viewer is designed to check for a correct display of generated problem-oriented shells of a DiVTB.

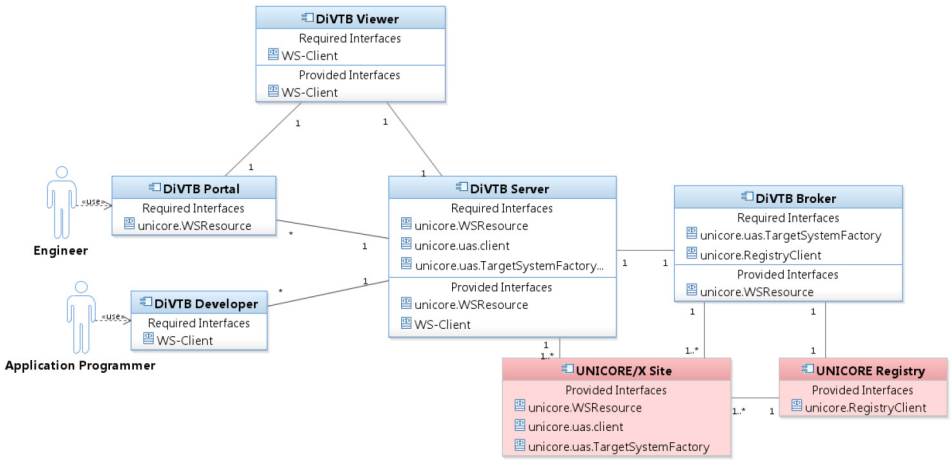


Figure 1: The architecture of the DiVTB system.

The DiVTB system is implemented on the UNICORE software platform [7] ensuring the access to services of the organisation in a distributed computing grid. In the DiVTB project the UNICORE platform is used as middleware which allows to provide an access to application packages in a grid environment. The UNICORE architecture consists of three layers: a client, a service, and a system layer.

The DiVTB Server, which executes tasks, is a client of the UNICORE environment and is located at the client level like other standard UNICORE clients: UCC (UNICORE Commandline Client) and URC (UNICORE Rich Client).

The service level is represented by various components of the UNICORE among which the UNICORE/X is important. The UNICORE/X is a component responsible for providing engineering resources to a grid environment. The DiVTB Broker component obtains in-

formation about existing applications of a grid computing environment and resources from the Registry and UNICORE/X components of the UNICORE.

The system level is presented by a target system interface (TSI). The TSI submits commands and executes applications on the grid environment, and performs a direct launch of an application package.

3 Integration of External Applications into the DiVTB System

The aim of the "Mobility of Young Scientists" project, carried out in 2012 with the participating of eleven young visiting scientists (teachers, graduate students and researchers), was the development of the technology for creating problem-oriented services in a distributed computing environment. The duration of the stay of the participants in Chelyabinsk, who carried out the work on the project, was very short, from 1 to 3 days. The time was too insuficient to learn, install and configure the DiVTB and UNICORE systems, and to develop a methodology for integrating the participants' own applications into the DiVTB system.

In order to save time, a development team of the DiVTB system prepared the following set of materials for the project participants:

- a quick guide for the DiVTB system;
- a virtual machine;
- a guide for the virtual machine;
- examples of project integration;
- a visualisator software for test beds.

All the materials were made available on a web page before the "Mobility of Young Scientists" project started. The quick guide for the DiVTB system contains some general information about the DiVTB software system including a description of features, software and tools that support the operation of the DiVTB system; a description of the architecture of the DiVTB system and its interaction with the UNICORE platform; as well as an integration methodology of external applications explained with the help of an example using POV-Ray and LS-Dyna.

The virtual machine is the most important item on the list of materials prepared for the participants. The virtual DiVTB machine was created using the Oracle VM VirtualBox application [8]. The OpenSUSE and the UNICORE platform, including the UNICORE Commandline Client, were installed.

Additionally, two user accounts were created on the virtual machine of the DiVTB system: user and root. The first account is recommended to be used for configuring necessary applications and the second one serves to configure an operating system (for example, to install required system libraries, applications, etc.).

In addition, the user's home directory contains an example of a test task named "unicore_test", which raises number X to the power of Y and shows how to use the basic functions (transmission of parameters, downloading and uploading of files) in the DiVTB system.

The methodology of integrating external applications into the DiVTB system comprises two steps:

1. A participant describes his or her own application in a simpleidb file.

2. A participant creates his or her problem-oriented shell for the DiVTB Portal component to trigger an external application as a grid service remotely.

Step 1. A description of an application in the simpleidb file has to be written into the simpleidb file on the UNICORE system in a similar way. Tab. 1 presents a Pow application description for raising X to a Y power. The Pow application takes two input parameters: an original file with a pair of numbers for raising to a power and an output file for recording the results.

```
<idb:IDBApplication>
<idb:ApplicationName>Pow</idb:ApplicationName>
   <idb:ApplicationVersion>1.0</idb:ApplicationVersion>
   <jsdl:POSIXApplicationxmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix">
      <jsdl:Executable>/bin/pow</jsdl:Executable>
      <jsdl:Argument>in=$INPUT_FILE</jsdl:Argument>
      <jsdl:Argument>out=$OUTPUT_FILE?</jsdl:Argument>
   </jsdl:POSIXApplication>
</idb:IDBApplication>
```

Table 1: A description of an application in a simpleidb file.

Step 2. A problem-oriented shell for the DiVTB Portal has to be created, which produces an XML-document containing identification parameters of the shell as well as a description of parameter groups and their values. In this case, the root element is a tag <problemCaebean> containing the name of a task, the name of an author, as well as a version and an internal identifier of the shell. The internal identifier (uid) is to be assigned in the DiVTB Server, when the shell is integrated into the DiVTB system.

The root node should always contain two elements:

- a tag <categories>, which describes parameters of a problem;
- a tag <resources>, which contains a description of a problem-oriented shell interface.

Each group of parameters has a specific name within the task performed by the DiVTB Portal, as well as a link to an appropriate resource, containing a text representation of a group name. The parameters listed in the group include: a name, a type and an availability parameter to be changed by a user, a text description, a unit of measure, a comment of a developer of a problem-oriented shell, and a default value. An example of setting parameters and groups for the Pow application is shown in Tab. 2.

4 The DiVTB Viewer

The participants of the project did not have to perform any final integration of their applications into the DiVTB Portal. They were able to display the problem-oriented shell

```
<?xml version="1.0" encoding="utf-8"?>
<problemCaebean xmlns="http://caeserver.caebeans.org" name="Concept_IH"
  author="Zakharov" version="1.0" caebeanId="pow-test">
  <categories>
    <category name="first_pair" data="first_pair_data">
      <parameter name="first_parameter" type="Float" visible="true">
        <text data="first_parameter_T" />
        <units data="first_parameter_U" />
        <comment data="first_parameter_C" />
        <enums />
        <default>2</default>
        <value> </value>
      </parameter>
      <parameter name="second_parameter" type="Float" visible="true">
      ...
      </parameter>
    </category>
  <resources>
    <language xml:lang="eng">
      <data name="first_pair_data">First pair</data>
      <data name="first_parameter_T">X</data>
      <data name="first_parameter_U" />
      <data name="first_parameter_C">The X number</data>
      <data name="second_parameter_T">Y</data>
      <data name="second_parameter_U" />
      <data name="second_parameter_C">The Y number</data>
    </language>
  </resources>
</problemCaebean>
```

Table 2: Creating a problem-oriented shell.

by using the DiVTB Viewer. The testbed.xsl file was attached to their problem-oriented shell in order to transform the XML format into the HTML format. The result was a visual representation of the problem-oriented shell.

The final integration of applications into the DiVTB system was perfomed as follows:

- An engineer passes a tuned virtual machine and a problem-oriented shell to a developer of the DiVTB system.
- A developer of the DiVTB system generates a project file for the problem-oriented shell for the engineer.
- The developer integrates the virtual machine for the engineer on the UNICORE site.
- The developer integrates the resulting test bed into the DiVTB Server component.

5 Integration of External Applications

Within the research of the project "The Technology of Creating Problem-Oriented Services in Distributed Computing Environments", a mechanism for integrating application packages into the DiVTB system was developed and a set of distributed virtual test beds for dealing with new classes of tasks in various problem fields was implemented.

5.1 Simulation of Chemical Processes

In this project, Ph.D. Kamila Koledina developed a distributed virtual test bed for the study of chemical processes. The main objective of the test bed is to enable the construction of a kinetic model of a hydroalumination reaction [9] and to simulate the reaction induction period in order to predict the latter's behaviour under different source data.

The developed information system includes: features of field and computational experiments, mathematical modelling of reactions, chemical processes, and an information computer complex with an ever-growing database of kinetic studies. It can reduce the time for developing kinetic models of complex reactions of metal catalysis, thus accelerating the research and exploration of new processes.

From a practical point of view it is important to monitor the process of the induction period flow. This allows to:

- prevent an explosion in the reactor where the reaction takes place;
- obtain the maximum output of the useful product;
- get the minimum amount of the by-product, etc.

The results of such an application are calculated values of the reactant concentration which then were stored in a database. The verification of the conservation law was written to the output file.

5.2 Problem Solving for Production of Nanosystems and Nanomaterials

Yury Kiriyenko, a fellow worker of Far Eastern Federal University, has developed, using efficient algorithms, an application for obtaining data on magnetic properties of nanomaterials based on the simulation of their magnetic properties [10].

Currently, the production of nanosystems and nanomaterials is experiencing a rapid growth coupled with an enormous volume of data about objects of a nanometer scale with unique properties, and with an opened-up possibility of mass production of such objects. Ultrathin films occupy a special place among these new facilities. On the one hand, they are characterised by a relative simplicity of obtaining. On the other hand, they have all the inherent properties of nanoobjects. The magnetic properties of these films are of considerable interest for science and technology [11].

The designed test bed for the calculation of magnetic properties of ultrathin films is implemented as a set of programmes written in the PYTHON programming language. Additional libraries for solving systems of nonlinear transcendental equations with many unknowns are also used.

As an example, we will consider the performance of the test bed with the following input data: the number of film monolayers, the precision of calculations of Curie temperature, and the type of solver. The result of the test bed's performance is a magnetisation curve of a two-layer model shown in Fig. 2.

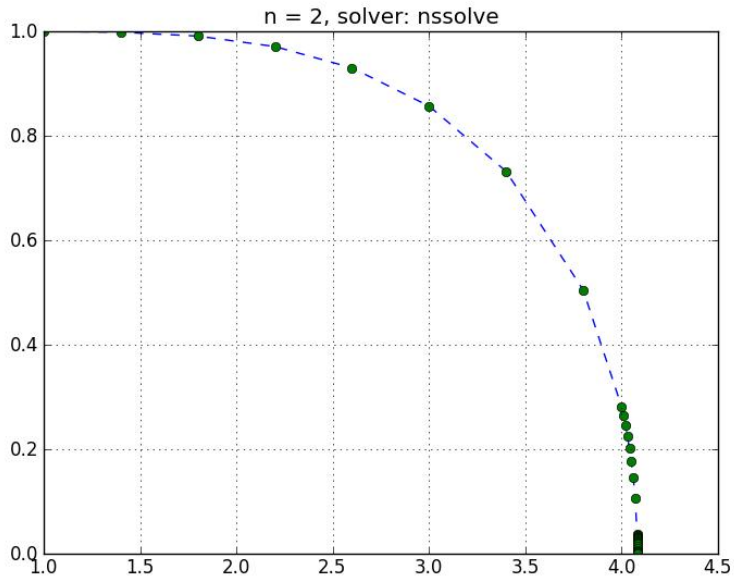


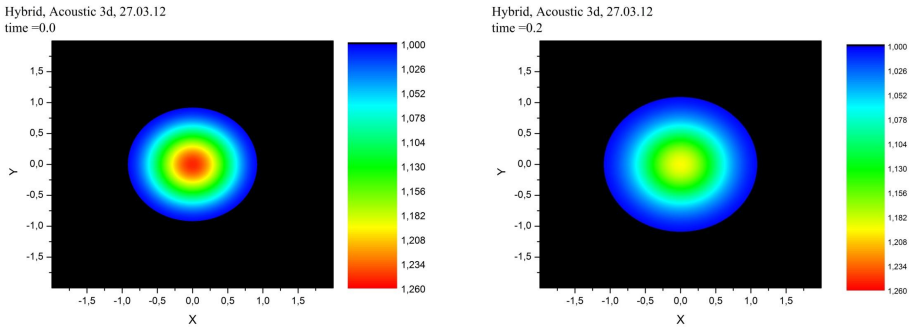
Figure 2: A magnetisation curve of a two-layer modelling.

5.3 Simulation of Mechanics of Continuous Media

Another example of the test bed is an application used for the simulation of acoustic processes developed by Ph.D. Igor Kulikov, a fellow worker of the Institute of Computational Mathematics and Mathematical Geophysics, the Siberian Branch of the Russian Academy of Sciences.

A problem statement consists in solving equations of acoustics in an anisotropic medium. Solving such equations is connected with a correct description of strong discontinuities: shock waves in an anisotropic medium [12]. The problem of discontinuous solutions in all areas of continuous media mechanics is acute. As a basic approach, a linear acoustic model of continuous media mechanics was chosen and Godunov's method was used. In the context of the project, a software complex for solving problems in acoustics using a hybrid supercomputer has been created. This complex can be used to study acoustic problems and to build more complex models of continuum mechanics on its base.

As a basic solver for tasks of acoustics, a two-dimensional formulation of linear acoustic theory equations was chosen. The field of our task was characterised by a parametrisable



(A) (B)

Figure 3: (A) An initial pressure distribution; (B) a final pressure distributions.

constant density, a parametrisable constant speed of sound, and by an initially-specified constant wave-exciting pressure source. The input parameters for the test bed are presented in a file including a dimensionless speed of sound, a density of the material and an initial additional pressure.

The output data of this test bed were written to a file with the finally calculated acoustic parameters recorded in five columns, namely: the X-coordinate (1), the Y-coordinate (2), the point pressure (3), the longitudinal velocity of perturbations (4), and the transverse velocity of perturbations (5). Fig. 3 shows initial and final pressure distributions.

5.4 Remote Visualisation of Results

Remote visualisation is a technique in which a visualisation system generates a graphical image on a separate computer and transmits it via a network to a user's computer. Thus, there is no need for installing visualisation software there. This significantly reduces requirements imposed on the user's computer, as well as eliminates potential delays associated with the transmission of data.

The staff of the Institute of Mathematics and Mechanics UB RAS was engaged in the study of remote visualisation [13]. The problem of remote visualisation was solved by refining the existing SharpEye system of scientific visualisation (Fig. 4). The main idea was to duplicate selection components at an infrastructure level.

In addition, a REST-server for the SharpEye [14] was developed, which allows the DiVTB system to operate as a Web service for performing the visualisation of a job based on a query. The latter includes data for the visualisation and an optional script to modify load data and scene display. As a result, a query service returns a response in the form of an image (Fig. 5).

The developed web service protocol is compatible with a JAVA client for the UNICORE platform. Thus, a component is designed for carrying out a remote visualisation on demand within the framework of distributed virtual test beds. The program is written in Ruby language and installed on the web-server based on the Sinatra platform.

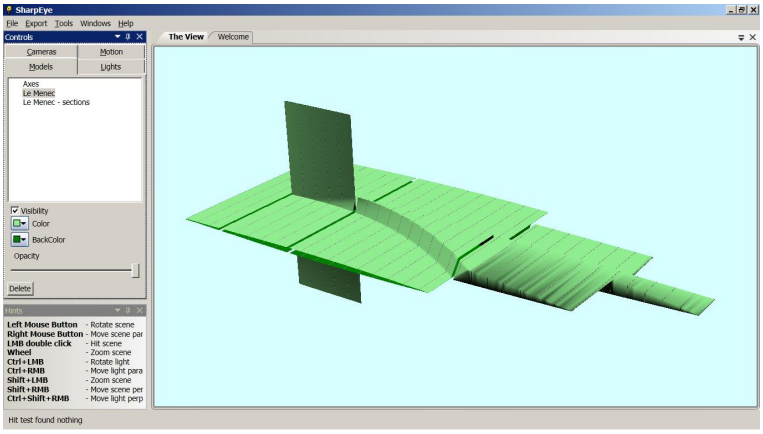


Figure 4: A screenshot of the main window of the SharpEye visualisation programmer.

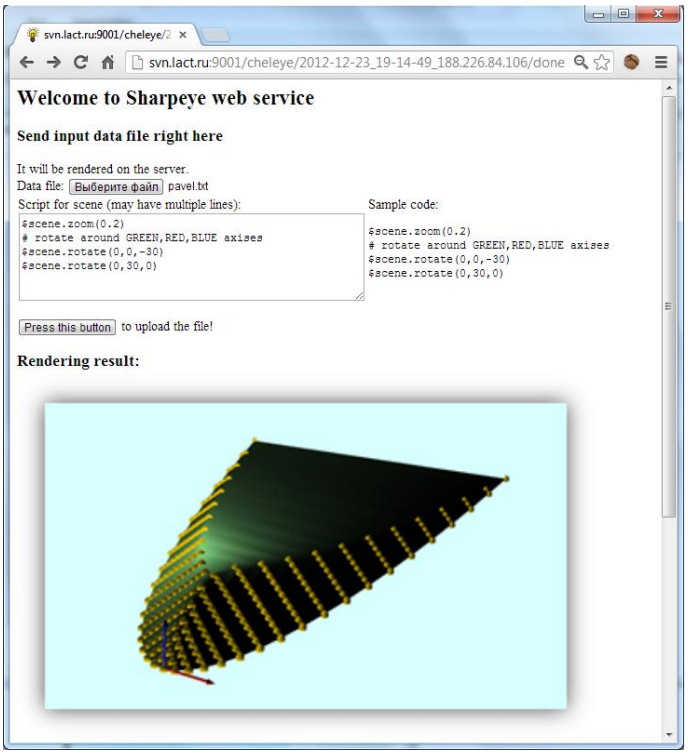


Figure 5: A performance task with a circular varying of speeds in the plane and a Dirichlet problem associated with it (for a differential Isaacs-Bellman equation).

6 Conclusions

In a context of the "Mobility of Young Scientists" project carried out in 2012, the goal to integrate external applications into the DiVTB system was achieved with the participation of the invited young specialists. Particularly, the studies in the area of simulation of chemical processes were carried out by specialists of Bashkir State University and the Institute of Computational Mathematics and Mathematical Geophysics SB RAS. The integration of applications for solving tasks for the production of nanosystems and nanomaterials was performed by the staff of Far Eastern Federal University.

Furthermore, the developments in the area of remote visualisation were undertaken by specialists of the Institute of Mathematics and Mechanics, the Ural Branch of the Russian Academy of Sciences. In addition, the solution of application tasks for processing large volumes of images was due to the activity of specialists of Ural Federal University; whereas the introduction of mathematical and numerical analysis was done by specialists of the Omsk Branch of the Sobolev Institute of Mathematics. And finally, the new methods of data flow management were developed in Perm National Research Polytechnic University.

During the implementation phase of the "Mobility of Young Scientists“ project, the following results were gained:

- development of a technology for creating problem-oriented services in a distributed computing environment;
- creation of a methodology for the integration of problem-oriented applications;
- elaboration of a set of distributed virtual test beds for the solution of problems in various fields of knowledge.

Overall, the results of the project have demonstrated an ease of the integration of external packages into the DiVTB system and a demand for a technology for creating problem-oriented services in distributed computing environments in various areas of knowledge. They also have shown that the UNICORE platform as a middleware provides a higher level of scientific and technical research. The results achieved comply with the standards of research in the development of problem-oriented grid environments.

Acknowledgements

The present work was supported by the Federal Target Programme "Scientific and Scientific-Pedagogical Personnel of the Innovative Russia for the Years 2009-2013" (No. 14.37.21.2088). The performed research was partially supported by RFBR, Research Project No. 11-07-00478-a, and by the Ministry of Education and Science of Russia, Task No. 8.3786.2011.

References

1. I. Foster and C. Kesselman, *The Grid 2, Second Edition: Blueprint for a New Computing Infrastructure* (Morgan Kaufman, San Francisco, 2003).
2. G. Radchenko and E. Khudyakova, *Distributed Virtual Test Bed: an Approach to Integration of CAE Systems in UNICORE Grid Environment* Proceedings of the 36th International Convention MIPRO 2013, pp. 183-188, 2013.
3. G. Radchenko *Methods of organising grid shell of a system layer in the CAEBeans technology*, J. Bulletin of SUSU. Series "Mathematical Modelling, Programming and Computer Software" **15(115)**, 69–78, 2008.
4. G. Radchenko *The CAEBeans grid system: integration of engineering packages resources in distributed computing environments*, J. Bulletin of the Nizhny Novgorod University N.I. Lobachevsky **6(1)**, 192–202, 2009.
5. G. Radchenko and E. Khudyakova, *A Service-Oriented Approach of Integration of Computer-Aided Engineering Systems in Distributed Computing Environments* Proceedings of the 2012 UNICORE Summit, IAS Series Volume 15, pp. 57-66, 2012.
6. A. Shamakina, *Brokering Service for Supporting Problem-Oriented Grid Environments*, Proceedings of the 2012 UNICORE Summit, IAS Series Volume 15, pp. 67-75, 2012.
7. K. Benedyczak, P. Bała, S. van den Berghe, R. Menday, and B. Schuller, *Key aspects of the UNICORE 6 security model*, J. Future Generation Comp. Syst. **27(2)**, 195–201, 2011.
8. Oracle VM VirtualBox: a virtualization software package, see `https://www.virtualbox.org/`
9. E. Y. Pankratyev, T. V. Tyumkina, L. V. Parfenova, L. M. Khalilov, S. L. Khursan, and U. M. Dzhemilev, *DFT Study on Mechanism of Olefin Hydroalumination by $XAlBu^i_2$ in the Presence of Cp_2ZrCl_2 Catalyst. I. Simulation of Intermediate Formation in Reaction of $HAlBu^i_2$ with Cp_2ZrCl_2*, J. Organometallics **28(4)**, 968–977, 2009.
10. L. Afremov and Y. Kirienko, *Magnetic phase transitions in ultrathin films of different crystal structures*, J. Advanced Materials Research **378**, 589–592, 2012.
11. C. Liu and S. D. Bader, *Magnetic properties of ultrathin epitaxial films of iron*, J. Magnetism and Magnetic Materials **93**, 307–314, 1991.
12. D. Folini and R. Walder, *Supersonic turbulence in shock-bound interaction zones. I. Symmetric settings* J. Astronomy and Astrophysics **459(1)**, 1–19, 2006.
13. V. L. Averbukh, S. S. Kumkov, V. S. Patsko, O. A. Pykhteev, and D. A. Yurtaev, *Specialized Visualization Systems for Differential Games*, Progress in Simulation, Modeling, Analysis and Synthesis of Modern Electrical and Electronic Devices and Systems, N.E.Mastorakis (Ed.), WSES Press, pp.301-306, 1999.
14. SharpEye: a system of scientific visualisation, see `http://www.sharpeye.lact.ru/`

Advancements in UNICORE Accounting

Piotr Bała[1,2], **Krzysztof Benedyczak**[1], **Rafał Kluszczyński**[1], **and Grzegorz Marczak**[1,2]

[1] Interdisciplinary Center for Mathematical and Computational Modelling, University of Warsaw, Warsaw, Poland
[2] Nicolaus Copernicus University Toruń, Poland
E-mail: {golbi, bala, klusi, grzemar}@icm.edu.pl

In this paper we describe the advancements in the UNICORE accounting system developed at ICM. The system is used in production since 2010 and has undergone many developments since that time. The most significant improvements were applied to the persistence layer, in order to provide better scalability, with the ability to effectively handle (process, query) millions of job records. The initial proof-of-concept accounting data presentation web application was redesigned and rewritten to offer production-grade features and speed. Finally the data gathering modules were enhanced to collect more information. The collected data is now available to the system users (administrators and managers) as well as for export to external systems such as EGI APEL or NGI-PL BAT using the latest accounting data formats.

1 Introduction

The initial version of the UNICORE accounting system [1] developed at ICM, was released back in 2010. Subsequently it has been deployed in the PL-Grid production infrastructure. Since this date the system has processed several millions of jobs. Each of them was assembled from several (typically around 6) partial job records produced by different instances of sensors collecting resource usage data. The number of partial job records is highly dependent on batch subsystem used.

During this time, the integration of the system with other accounting solutions started to play an important role. On one hand, the propagation of the UNICORE records to the central, middleware agnostic, Polish Grid accounting system was a constant necessity. On the other hand, reporting of UNICORE data to the central accounting system of the European Grid Initiative has been requested, as many national UNICORE deployments are incorporated in EGI. Finally operational requirements grew significantly: the possibility to support both UNICORE and local jobs as well as advanced querying and reporting capabilities were needed not only to give an overall picture of resource usage, but also to perform more advanced tasks such as the detection of anomalies in Grid usage.

In the following text we describe solutions for the challenges presented above. The next section describes the general system design. Subsequent sections cover the new persistence layer, enhanced accounting data gathering, the new accounting portal, and finally the data export to the APEL system. The paper is summarised with a brief overview of the planned developments.

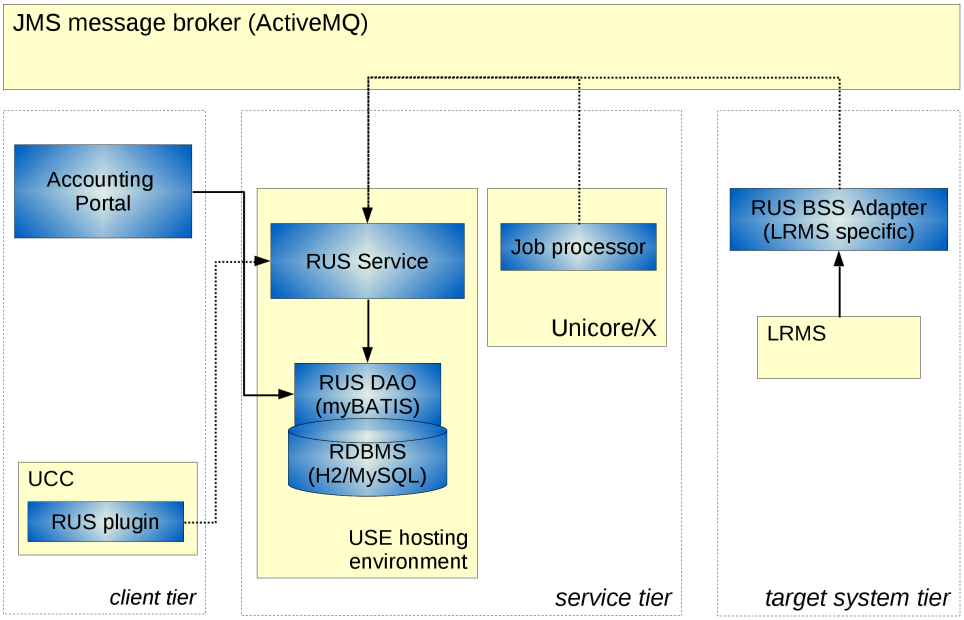


Figure 1: The architecture of the UNICORE accounting system.

2 Overview of the System Design

The UNICORE accounting system is a highly distributed system, composed of several elements, cooperating together. The three main building blocks are Unicore/X RUS job processor, BSS-Adapter and RUS-Service[a]. The first two parts are the sensors, collecting jobs' data from a Grid service and from a Batch Subsystem (BSS) respectively. The RUS-Service is the heart of the whole system. It is responsible for merging the job data coming from the all sensors (usually there are many instances of BSS-Adapters and Unicore/X job processors installed) and storing the records in a database. RUS-Service offers also an optional export feature, i.e. it can use plugins to translate the job records into arbitrary format and to send them to external systems.

The communication is based on the Java Message Service interface and its ActiveMQ implementation. Decision to use message oriented middleware was motivated by a need of robust transport, which can handle large traffic, with a guaranteed delivery even if some of the actors are temporarily down. Properly configured ActiveMQ messaging system fulfils those goals and is proved to be the right choice: during several years of system operation we have not experienced any data loss on the transport level and the performance overhead of the messaging subsystem is negligible.

The internal format of job record is compliant with the Open Grid Forum Usage Record

[a]The RUS acronym is used for the historical reasons referring to the Open Grid Forum work on an interface for retrieving accounting data, called Resource Usage Service. The unfinished work was abandoned, but UNICORE accounting system implemented an early draft of the interface. Currently this interface is not available and is replaced by a more complete and simpler version.

specification[2], however many extensions are defined to store more advanced pieces of information. The details are presented in section 4.

Accounting data can be consumed using one of the two available clients. The most versatile one is a web application called RUS-WebUI. RUS-WebUI provides an advanced visualization and is described more deeply in the section 5. There is also a more low-level solution, a plugin to the UNICORE Command Line Client (UCC). The plugin offers features useful for system administrators: it is possible to get a list of records matching provided criteria and to force execution of export plugins of the RUS-Service. The latter feature is useful to replay data which was lost on an external service side.

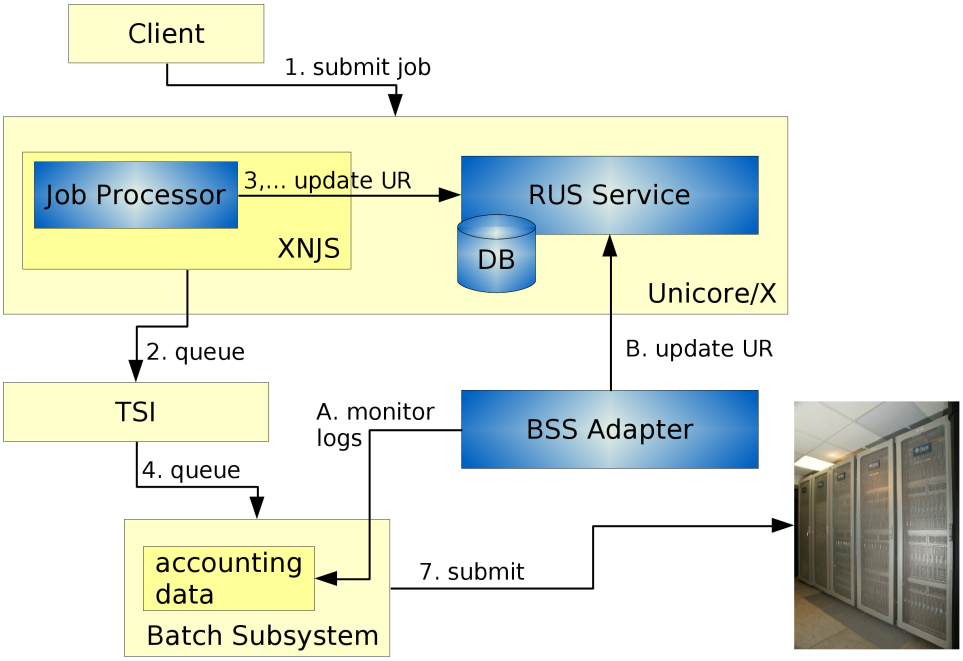


Figure 2: The typical data flow in the accounting system. Numbers and letters show the order of operations. It can be noted that BSS-Adapter works without any coordination with the RUS job processor and the actual order of partial records at the RUS-Service is random.

Figure 1 depicts the architecture of the complete system. Subsequently figure 2 presents the processing of a single job. On the latter figure one of the fundamental features of the system can be noted: the system collects the accounting information of all jobs, also those running or even just submitted and queued. Therefore the accounting information allows for a simplified monitoring of the Grid infrastructure, e.g. showing the number of queued jobs at a particular point in time.

3 The New Database Layer

The original accounting system used a relational database to store job records. Then a hybrid storage model was employed: raw XML data was stored together with some of its fields duplicated in separate columns. The duplicated data was used to achieve fast queries of records. In principle a single table was used.

Over time it turned out that number of records grows quickly and in the case of the PL-Grid deployment it reached the value of 1,000,000 in less then one year. More or less from this database size, the database became the bottleneck of the whole system. Complicated queries took several seconds, what was a result of heavy load of the database caused by a constant flow of incoming job records being added or merged with existing ones. For the Torque BSS the system has to process six partial records per job (sent when job's status is changed from two probes). For the SLURM BSS this number is even larger. Deployment of a more powerful hardware would be only a short term solution.

Therefore the storage layer was redesigned and the new version is using a quite different persistence model. It is based on the following observations:

- The managers using the system are interested in summarised data. The daily granularity of usage is perfectly enough for them, assuming it is possible to get all job details (owners, queues, final status etc). In other words they are not interested in at what time a particular job was started or finished, rather what was the cumulative usage of resources per day, per each user, queue, site, etc.

- The system administrators are sometimes interested in more detailed accounting data. However this interest is highly time limited, as the information is merely used to track problems in the infrastructure. Very detailed data for the jobs finished half a year ago is nearly never requested.

- The amount of individual jobs is much higher then their daily summaries, even when grouped by all possible distinguishing parameters: site, BSS host, global user, job status, queue, project, VO, local user and submit host. Table 1 provides a more concrete evidence for this claim. This regularity is caused by the fact that on each day only a limited number of users is submitting jobs, and typically a user submits a lot of jobs during a single workday. At the same time the amount of distinct values of other parameters is small (there are few queues per BSS, several sites, etc). It is also worth to note that the HTC jobs (e.g. LHC Grid jobs) bear the same characteristic, with even larger reduction.

The above observations allowed us to design a universal solution. The individual job records are stored as in the original version, but at the same time another table with daily summary records is used. The summaries in this table contain only usage from completed jobs, where we can be sure that values are not going to be modified further. The resource usage of jobs spanning several days is divided proportionally. Additionally a *records rolling* mechanism is implemented. This mechanisms moves all old individual job records to a historical jobs table and all old summaries to the historical summaries table. Both rolling functions are configured with a separate time threshold after which data is assumed to be old. In the case of individual job records this threshold should be small, e.g. 6 months as the database grows quickly. For the summaries, the threshold can be much

larger (e.g. several years) - even more then 10 times as the daily summaries use about 20 times less records to store the same data as the individual jobs. After fine tuning the parameters based on the available hardware and operational requirements, the system can run for many years unattended.

The database layer offers a rich query interface. The vast majority of queries is performed using the daily summaries. As the data amount in the summaries table is small, the performance is high. Still, in the unlikely case that the historical data is needed, it is safely stored in separate tables.

4 Enhanced Data Gathering and SLURM Support

As it was mentioned in section 2, the RUS-Service component is responsible for merging complementary records coming from different sources. Records are matched together either by a global job identifier or by a local job identifier and BSS machine name pair. The merging of other elements is a complicated process and is subject to several algorithms. The initial approach using quite simple merging was replaced with a formalized one, described below.

This development was especially important when support for the SLURM BSS was added. The semantics of accounting information turned out to be slightly different from what was reported by the previously supported Torque BSS and a well defined data merging process was a necessity to guarantee accounting data coherence.

The table 2 shows all supported record elements along with components which can produce them and the merging algorithm applied. Note that in many cases a particular piece of data is provided only in records coming from a single source.

The merging algorithms used in the table are:

- *first-win* the first non-empty value is used, the subsequent ones are ignored.
- *bss-win-warn* the value from BSS overwrites the older ones. When a different value is received a warning is produced, regardless of the actual change after merge (i.e. the value should be constant in the system).
- *bss-win* the value from BSS overwrites the older ones.
- *ux-win-warn* the value from Unicore/X overwrites the older ones. When a different value is received a warning is produced, regardless of the actual change after merge (i.e. the value should be constant in the system).

	June 2013 week 1	June 2013 week 2	June 2013 week 3
Individual jobs	1,028	1,774	1,954
Daily summaries	56	78	89
Reduction	18.36	22.74	21.96

Table 1: Example of records number reduction between individual job records and daily summaries for the ICM Hydra cluster, taking into account only the UNICORE jobs in the first three weeks of June 2013.

- *fail-on-change* if a different value is received, then the new record is ignored with a warning (i.e. the value must be the same in the system)
- *ignore-new-warn* if a different value is received, then the new value is ignored with a warning (i.e. the value should be the same in the system)
- *overwrite-status* used to merge job status: the new status overwrite the old one, but only if the old one was not in one of the terminal states.

As it can be seen from the table 2, the UNICORE accounting system collects a large amount of accounting data about each job. The collected data makes the use of the system practical also in cases when only a single probe is available. For instance when there is no BSS-Adapter installed, the data coming from Unicore/X can be used to assess job's approximate start and end time.

The rich collected data can be also used to monitor system anomalies. For instance the maxWalltime information which is reported on job start, allows for detecting jobs which hung in the cluster, even though should be killed/removed by BSS.

The SLURM support itself turned out to be the most difficult module of the BSS-Adapter. SLURM supports different storage backends for accounting data. In order to make the RUS BSS-Adapter independent from a SLURM backend, we developed a solution accessing the SLURM accounting data via its standard reporting "sacct" command. Unfortunately because of this the records of unfinished jobs are send multiple times, even if not changed they reappear in the "sacct" output on each invocation. A special caching of sent records was developed to minimize such situations. Finally it was noticed that SLURM allows itself to change the state of jobs even after those are finished (for instance CPU time is sometimes slightly adjusted). Therefore we still work on the SLURM module to get fully reliable data.

5 Accounting Portal

The first release of the UNICORE accounting solution featured a simple web interface presenting accounting database contents. This initial version maintained a separate copy of accounting data in a local database. With the new release of the base components, the portal was rewritten to use the new, unified database layer and become a first class citizen in the portfolio of accounting components.

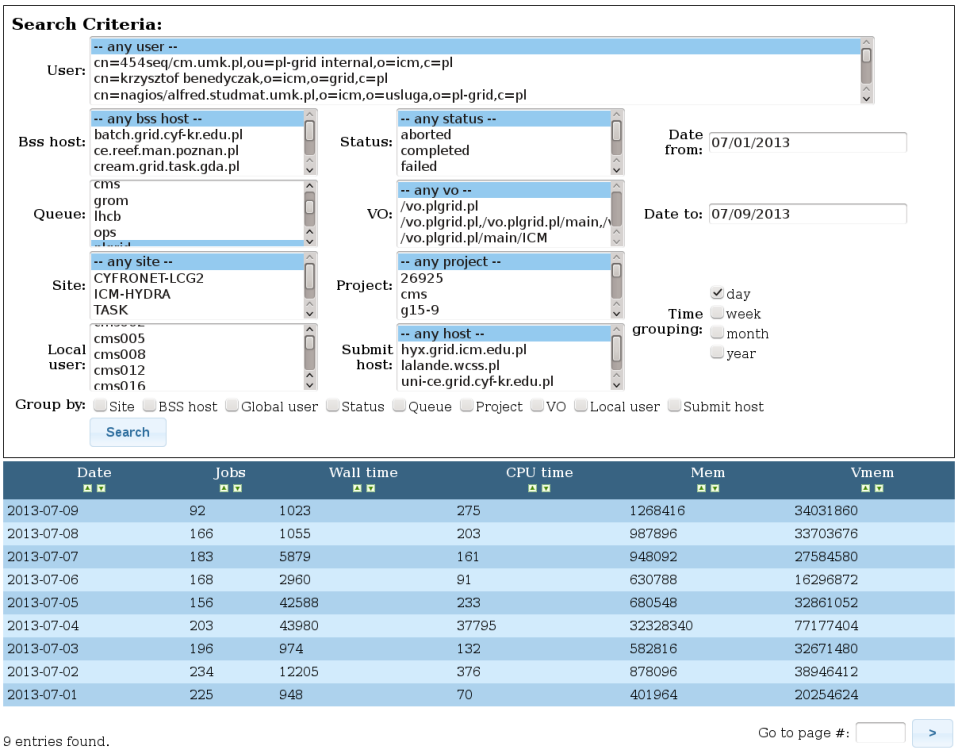
This evolution was possible thanks to the update of the RUS-Service database interface API, which made it much more flexible, and sufficient for the rich query interface of the web interface. Theoretically it might be tempting to have a separate copy of the same database for the portal, to split the database load. However, the optimization described in the section 4 eliminate the performance problems, so we did not provide such an option.

The base query panel of the web interface (see figure 3) is presenting the data from the daily job summaries database table. The top part of the interface allows for specifying advanced record selection criteria and grouping options from each of the two sets described below. First of all it is possible to select time based grouping: daily, weekly, monthly or yearly. The aggregated data of records is added for the grouping period. For instance with weekly grouping, jobs amount, wall time, CPU time, and memory consumption is summed for each week which falls into the time range from the records selection criteria.

Additionally the portal user can group the results with other fields. For example it is possible to get monthly statistics of resources usage grouped by queues and sites together, or by Grid users.

XML element name	Source	Merging alg.	Description
RecordIdentity@createTime	BSS & U/X	The earliest value is used	Initial record creation time
RecordIdentity@recordId	U/X	first-win	Unique record identity
JobIdentity/GlobalJobId	U/X	fail-on-change	UNICORE Job id
JobIdentity/LocalJobId	BSS & U/X	fail-on-change	BSS Job Id
UserIdentity/LocalUserId	BSS & U/X	bss-win-warn	Owner's uid
UserIdentity/GlobalUserName	U/X	fail-on-change	Owner's DN
JobName	BSS	ignore-new-warn	Job's name
ProjectName	BSS	ignore-new-warn	Job's project
Status	BSS & U/X	overwrite-status	Job's status
MachineName	BSS & U/X	fail-on-change	BSS server name
SubmitHost	BSS & U/X	ux-win-warn	Machine from which the job is submitted to BSS
Queue	BSS	bss-win	BSS queue name
Processors	BSS	ignore-new-warn	Total number of CPUs
NodeCount	BSS	ignore-new-warn	Total number of nodes
Hosts	BSS	ignore-new-warn	Used nodes list
StartTime	BSS	ignore-new-warn	Job's execution start time
EndTime	BSS	ignore-new-warn	Job's execution end time
WallDuration	BSS	ignore-new-warn	Job's wall time
CpuDuration	BSS	ignore-new-warn	Job's CPU time
TI@etime	BSS	ignore-new-warn	BSS enqueud time
TI@qtime	BSS	ignore-new-warn	BSS ready time
TI@ctime	BSS	ignore-new-warn	Time BSS receives the job
TI@maxWalltime	BSS	ignore-new-warn	StartTime+walltime
TI@uxToBssSubmitTime	U/X	ignore-new-warn	U/X to BSS submit time
TI@uxStartTime	U/X	ignore-new-warn	Time U/X detects job start
TI@uxEndTime	U/X	ignore-new-warn	Time U/X detects job end
Memory@shared	BSS	ignore-new-warn	Job's virtual memory
Memory2physical	BSS	ignore-new-warn	Job's physical memory
Resource@infrastructure	U/X	ignore-new-warn	'unicore'
Resource@exit_status	BSS	ignore-new-warn	Exit status of the job
Resource@group	BSS	ignore-new-warn	User's GID
Resource@vo	U/X	ignore-new-warn	User's effective VO
Resource@sitename	U/X (BSS)	ignore-new-warn	Site name
Resource@attributes	U/X	ignore-new-warn	User's authz. attributes
Resource@recordOrigin	BSS & U/X	-	bss, unicorex or merged

Table 2: Data collected by UNICORE accounting sensors with information in the source and merging algorithm used by the RUS-Service. "TI" = TimeInstant. "@" means that the data is encoded as XML attribute or that it is distinguished by XML attribute name, when there are many elements with the same name as in case of TimeInstant, Memory and Resource elements.



Date	Jobs	Wall time	CPU time	Mem	Vmem
2013-07-09	92	1023	275	1268416	34031860
2013-07-08	166	1055	203	987896	33703676
2013-07-07	183	5879	161	948092	27584580
2013-07-06	168	2960	91	630788	16296872
2013-07-05	156	42588	233	680548	32861052
2013-07-04	203	43980	37795	32328340	77177404
2013-07-03	196	974	132	582816	32671480
2013-07-02	234	12205	376	878096	38946412
2013-07-01	225	948	70	401964	20254624

9 entries found.

Go to page #: >

Figure 3: Selection criteria panel and example results of the daily job summaries data. An information on jobs submitted to a selected queue is presented, with a daily grouping for the first week of July 2013.

RUS-WebUI offers also the presentation of the aggregated data in graphical form. Using the same selection criteria as in the aforementioned statistics panel, the user can generate graphs, showing the total number of processed jobs, total consumed CPU time, wall time and memory (see figure 4) and an overall summary of job statuses (see figure 5).

Currently the graphs panel doesn't allow to select records grouping as the statistics panel does: always daily summaries are used (what is the best option for visualization as it provides the most accurate graphs). Any additional groupings can be considered, however it could be much more problematic as multiple lines would need to be drawn, blurring the final image.

Last but not least, the RUS-WebUI allows for viewing the individual job records (see figure 6). It is possible to select records with the same selection criteria as in the case of aggregated records. It is not possible to sum the data or group it, however records can be displayed even in a raw XML format, providing a full access to data stored in database. As it was previously noted, this tool is especially useful for system administrators checking the system for anomalies.

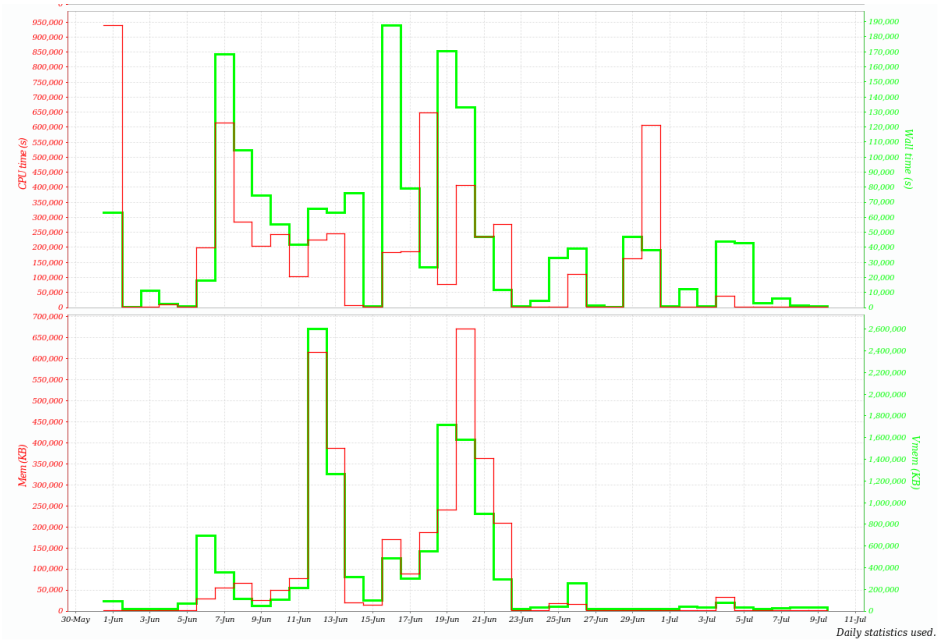


Figure 4: Aggregated CPU time, wall time, physical and virtual memory consumption chart. As in other cases, such a graph can be generated for any record selection criteria.

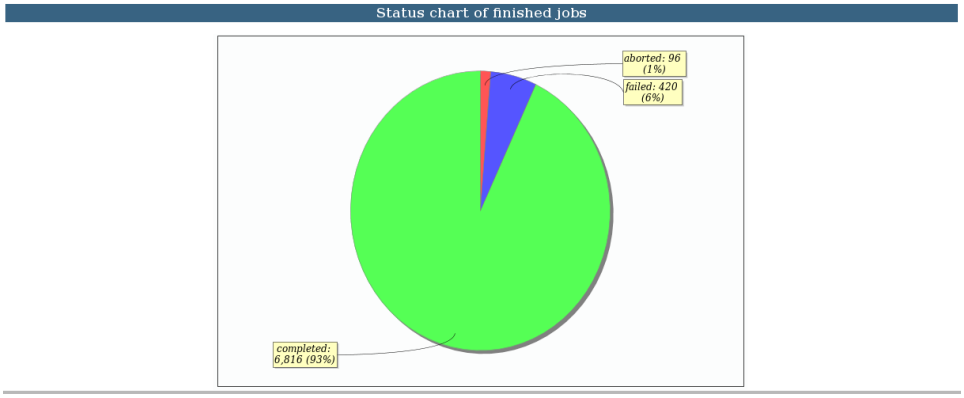


Figure 5: Graphical overview of job statuses is a simple but useful tool for quick assessment of the system's reliability. Such a pie chart can be generated for any selection criteria.

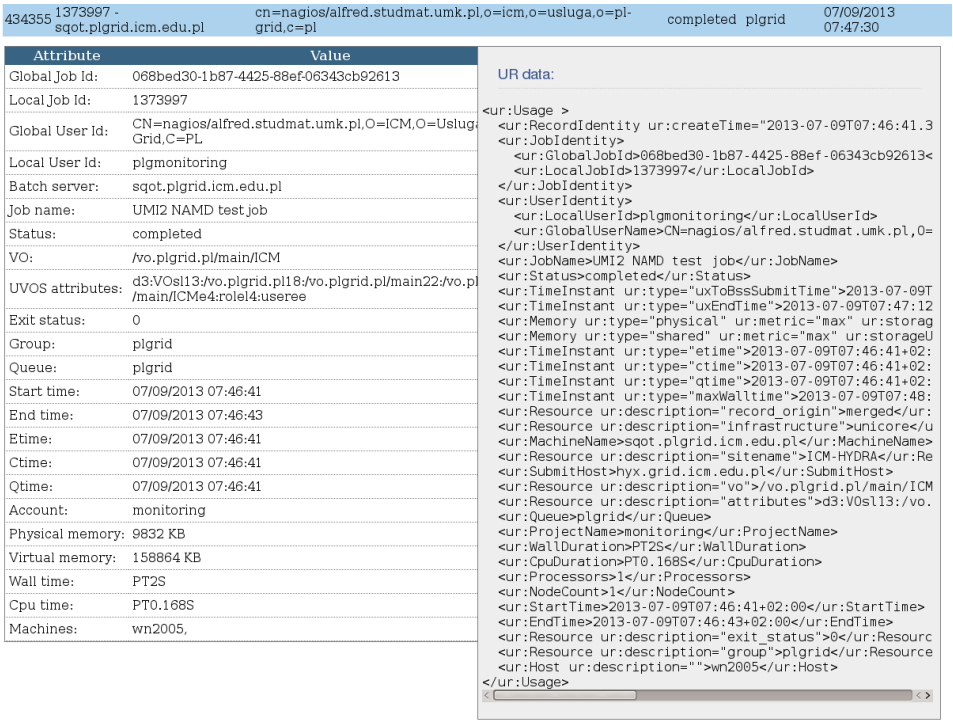


Figure 6: Detailed view of a record. Such an information is only available for the individual job records, which were not yet rolled to the historical database table.

6 EMPA CAR Support

During the last year, under the auspices of EMI project [3] a new usage record format was defined. The format called Common Accounting Record (CAR) is defined in [4]. It is a result of the collaboration between developers of the APEL [5], SGAS [6], DGAS [7] and UNICORE [8].

The principle goal of the CAR design was to provide standard definitions for many of the optional accounting record elements, which are often used in production. The CAR format is based on the OGF Usage Record format [2]. In the OGF UR, the optional elements are not defined, instead can be encoded as extensions. As different vendors used different extensions, the whole solution turned out to be not interoperable. The UNICORE team provided a significant input to the specification, based on the experience with the RUS accounting system.

Additionally, the EMI project has produced an accounting record exchange protocol, called EMPA [9]. The EMPA protocol is in fact a minimalistic document defining the best practices for usage of the standard messaging protocol STOMP [10] with the additional definition of two accounting specific header parameters.

The usage of the EMPA protocol and the ability to produce CAR records can be considered the most interoperable solution for Grid accounting nowadays. Therefore the RUS-Service was enhanced by a CAR export plugin, available in the base system distribution. The plugin allows to translate the internal OGF UR records to the CAR format and to send it over the EMPA/STOMP protocol. Both, aggregated and individual job records, are supported.

7 Summary and Outlook

In this paper we have presented the advancements in UNICORE accounting that were designed and implemented in the last years. The biggest challenge was related to the intensive database load, caused by the continuous stream of incoming job records and random user queries. The use of daily summaries as the base for the queries, together with rolling of old records allowed to solve this problem. At the same time an advanced and user friendly accounting portal was implemented, support for the interoperable EMI CAR format and the EMPA protocol as well as the more and more popular SLURM BSS were added.

A continuous work is needed to keep the system up to date and fix problems which occur in the software relatively often due to interactions with many external, components as the UNICORE middleware or BSS systems. Especially the SLURM support is problematic and still requires an effort to make it more reliable.

Additionally, we are planning to provide support for the commercial LoadLeveler BSS and to extend the portal capabilities. However the biggest challenge is the accounting of cluster node reservations. This feature is not supported by any system known to us, there are no standard formats for expressing reservations and the reservation lifecycle is more complicated then the simple job's lifecycle, as the reservation may be cancelled. At the same time reservations consume resources to a greater degree then the jobs, as nodes need to be drained for creating a reservation. This effort can be considered as a major challenge for the future.

8 Acknowledgements

The work described in this paper was partially funded by the PL-Grid (POIG.02.03.00-00-007/08-00) and PL-Grid Plus (POIG.02.03.00-00-096/10) projects as well as by the EMI project (RI-261611).

References

1. P. Bała, K. Benedyczak, M. Lewandowski, Resource Usage Accounting for UNICORE, *UNICORE Summit 2010 Proceedings*, 18 – 19 May 2010 Jülich, Germany. IAS Series 05, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 2010.
2. L. McGinnis (ed.) et al. Usage Record – Format Recommendation, `https://www.ogf.org/documents/GFD.98.pdf`.
3. European Middleware Initiative project, `https://twiki.cern.ch/twiki/pub/EMI/EmiOfficialDocuments/EMI_Overview_v2_1.pdf`.

4. EMI Definition of the Compute Accounting Record, https://twiki.cern.ch/twiki/pub/EMI/ComputeAccounting/CAR-EMI-tech-doc-1.2.doc.
5. APEL accounting system, https://wiki.egi.eu/wiki/APEL.
6. SGAS accounting system, http://www.sgas.se.
7. Distributed Grid Accounting System, http://www.to.infn.it/dgas/.
8. UNICORE project website, http://unicore.eu
9. EMI Messaging Protocol for Accounting (EMPA), https://wiki.egi.eu/wiki/File:EMI_Accounting_Messaging_Protocol_v1.1.pdf.
10. The Simple Text Oriented Messaging Protocol (STOMP), http://stomp.github.io/stomp-specification-1.2.html.

Schriften des Forschungszentrums Jülich
IAS Series

---

1. **Three-dimensional modelling of soil-plant interactions: Consistent coupling of soil and plant root systems**
   by T. Schröder (2009), VIII, 72 pages
   ISBN: 978-3-89336-576-0
   URN: urn:nbn:de:0001-00505

2. **Large-Scale Simulations of Error-Prone Quantum Computation Devices**
   by D. B. Trieu (2009), VI, 173 pages
   ISBN: 978-3-89336-601-9
   URN: urn:nbn:de:0001-00552

3. **NIC Symposium 2010**
   Proceedings, 24 – 25 February 2010 | Jülich, Germany
   edited by G. Münster, D. Wolf, M. Kremer (2010), V, 395 pages
   ISBN: 978-3-89336-606-4
   URN: urn:nbn:de:0001-2010020108

4. **Timestamp Synchronization of Concurrent Events**
   by D. Becker (2010), XVIII, 116 pages
   ISBN: 978-3-89336-625-5
   URN: urn:nbn:de:0001-2010051916

5. **UNICORE Summit 2010**
   Proceedings, 18 – 19 May 2010 | Jülich, Germany
   edited by A. Streit, M. Romberg, D. Mallmann (2010), iv, 123 pages
   ISBN: 978-3-89336-661-3
   URN: urn:nbn:de:0001-2010082304

6. **Fast Methods for Long-Range Interactions in Complex Systems**
   Lecture Notes, Summer School, 6 – 10 September 2010, Jülich, Germany
   edited by P. Gibbon, T. Lippert, G. Sutmann (2011), ii, 167 pages
   ISBN: 978-3-89336-714-6
   URN: urn:nbn:de:0001-2011051907

7. **Generalized Algebraic Kernels and Multipole Expansions for Massively Parallel Vortex Particle Methods**
   by R. Speck (2011), iv, 125 pages
   ISBN: 978-3-89336-733-7
   URN: urn:nbn:de:0001-2011083003

8. **From Computational Biophysics to Systems Biology (CBSB11)**
   Proceedings, 20 - 22 July 2011 | Jülich, Germany
   edited by P. Carloni, U. H. E. Hansmann, T. Lippert, J. H. Meinke, S. Mohanty, W. Nadler, O. Zimmermann (2011), v, 255 pages
   ISBN: 978-3-89336-748-1
   URN: urn:nbn:de:0001-2011112819

Schriften des Forschungszentrums Jülich
IAS Series

---

9. **UNICORE Summit 2011**
Proceedings, 7 - 8 July 2011 | Toruń, Poland
edited by M. Romberg, P. Bała, R. Müller-Pfefferkorn, D. Mallmann (2011), iv, 150 pages
ISBN: 978-3-89336-750-4
URN: urn:nbn:de:0001-2011120103

10. **Hierarchical Methods for Dynamics in Complex Molecular Systems**
Lecture Notes, IAS Winter School, 5 – 9 March 2012, Jülich, Germany
edited by J. Grotendorst, G. Sutmann, G. Gompper, D. Marx (2012), vi, 540 pages
ISBN: 978-3-89336-768-9
URN: urn:nbn:de:0001-2012020208

11. **Periodic Boundary Conditions and the Error-Controlled Fast Multipole Method**
by I. Kabadshow (2012), v, 126 pages
ISBN: 978-3-89336-770-2
URN: urn:nbn:de:0001-2012020810

12. **Capturing Parallel Performance Dynamics**
by Z. P. Szebenyi (2012), xxi, 192 pages
ISBN: 978-3-89336-798-6
URN: urn:nbn:de:0001-2012062204

13. **Validated force-based modeling of pedestrian dynamics**
by M. Chraibi (2012), xiv, 112 pages
ISBN: 978-3-89336-799-3
URN: urn:nbn:de:0001-2012062608

14. **Pedestrian fundamental diagrams: Comparative analysis of experiments in different geometries**
by J. Zhang (2012), xiii, 103 pages
ISBN: 978-3-89336-825-9
URN: urn:nbn:de:0001-2012102405

15. **UNICORE Summit 2012**
Proceedings, 30 - 31 May 2012 | Dresden, Germany
edited by V. Huber, R. Müller-Pfefferkorn, M. Romberg (2012), iv, 143 pages
ISBN: 978-3-89336-829-7
URN: urn:nbn:de:0001-2012111202

16. **Design and Applications of an Interoperability Reference Model for Production e-Science Infrastructures**
by M. Riedel (2013), x, 270 pages
ISBN: 978-3-89336-861-7
URN: urn:nbn:de:0001-2013031903

Schriften des Forschungszentrums Jülich
IAS Series

17. **Route Choice Modelling and Runtime Optimisation for Simulation of Building Evacuation**
by A. U. Kemloh Wagoum (2013), xviii, 122 pages
ISBN: 978-3-89336-865-5
URN: urn:nbn:de:0001-2013032608

18. **Dynamik von Personenströmen in Sportstadien**
by S. Burghardt (2013), xi, 115 pages
ISBN: 978-3-89336-879-2
URN: urn:nbn:de:0001-2013060504

19. **Multiscale Modelling Methods for Applications in Materials Science**
by I. Kondov, G. Sutmann (2013), 326 pages
ISBN: 978-3-89336-899-0
URN: urn:nbn:de:0001-20130902

20. **High-resolution Simulations of Strongly Coupled Coulomb Systems with a Parallel Tree Code**
by M. Winkel (2013), xvii, 196 pages
ISBN: 978-3-89336-901-0
URN: urn:nbn:de:0001-2013091802

21. **UNICORE Summit 2013**
Proceedings, 18th June 2013 | Leipzig, Germany
edited by V. Huber, R. Müller-Pfefferkorn, M. Romberg (2013), iii, 94 pages
ISBN: 978-3-89336-910-2
URN: urn:nbn:de:0001-2013102109

The UNICORE Grid technology provides a seamless, secure, and intuitive access to distributed Grid resources. UNICORE is a full-grown and well-tested Grid middleware system, which today is used in daily production worldwide. Beyond this production usage, the UNICORE technology serves as a solid basis in many European and International projects. In order to foster these ongoing developments, UNICORE is available as open source under BSD licence at http://www.unicore.eu.

The UNICORE Summit is a unique opportunity for Grid users, developers, administrators, researchers, and service providers to meet and share experiences, present past and future developments, and get new ideas for prosperous future work and collaborations. The UNICORE Summit 2013, the nineth in its series, has been held as a satellite event at the ISC Conference in Leipzig, Germany, on 18 June 2013.

The proceedings at hand include a selection of 9 papers that show the spectrum of where and how UNICORE is used and further extended, especially with respect to data management and application support.

This publication was edited at the Jülich Supercomputing Centre (JSC) which is an integral part of the Institute for Advanced Simulation (IAS). The IAS combines the Jülich simulation sciences and the supercomputer facility in one organizational unit. It includes those parts of the scientific institutes at Forschungszentrum Jülich which use simulation on supercomputers as their main research methodology.

IAS Series
Volume 21
ISBN 978-3-89336-910-2

JÜLICH
FORSCHUNGSZENTRUM