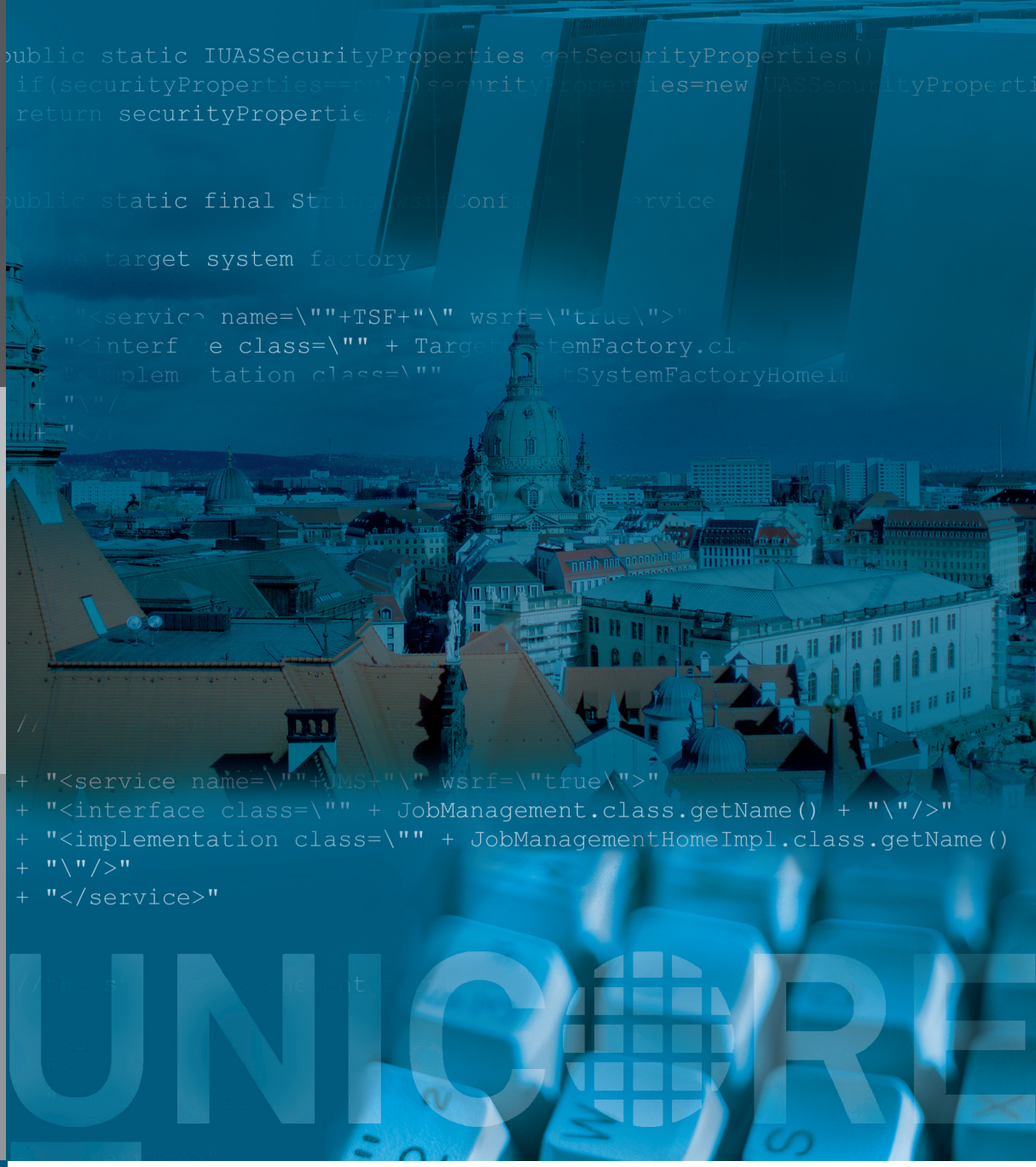

UNICORE Summit 2012

Proceedings, 30 - 31 May 2012 | Dresden, Germany

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

JÜLICH
FORSCHUNGSZENTRUM

Mitglied der Helmholtz-Gemeinschaft

Schriften des Forschungszentrums Jülich

IAS Series Volume 15

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

UNICORE Summit 2012

Proceedings, 30 - 31 May 2012 | Dresden, Germany

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

Schriften des Forschungszentrums Jülich

IAS Series Volume 15

ISSN 1868-8489 ISBN 978-3-89336-829-7

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 2012

Schriften des Forschungszentrums Jülich
IAS Series Volume 15

ISSN 1868-8489
ISBN 978-3-89336-829-7

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:0001-2012111202
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. The Summit has been held every year since it started in 2005.

The 2012 Summit took place at Dresden University of Technology, Germany, on May 30 and 31. Dresden is a modern cultural metropolis with lovely surroundings and important ties to the past.

The event was opened with an invited talk by Victor Gene Hazlewood from the University of Tennessee "UNICORE in XSEDE: The Journey Down the Road Less Traveled By", a presentation about the XSEDE project and the deployment of UNICORE on XSEDE resources. In the following, 14 technical contributions gave more insights to various UNICORE development and usage aspects. In addition, there were plenary discussions about various aspects of UNICORE. The main topics focused on the next generation of virtual organisations, data management, and the future ways UNICORE should go, as well as the subjects like "UNICORE and Portals".

We would like to thank all contributors for their presentations and papers as well as the organisers of the UNICORE Summit 2012 in Dresden for their excellent work. 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 2013!

November 2012

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)

Contents

Experience with UNICORE Services for Multiscale Materials Modelling

Michele Carpené[1], Stefan Bozic[2], Ivan Kondov[2], Andrew Emerson[1]

[1] CINECA
40033 Casalecchio di Reno, Bologna, Italy
E-mail: {m.carpen, a.emerson}@cineca.it

[2] Steinbuch Centre for Computing,
Karlsruhe Institute of Technology (KIT)
Hermann-von-Helmholtz-Platz 1,
76344 Eggenstein-Leopoldshafen, Germany
E-mail: {stefan.bozic, ivan.kondov}@kit.edu

The MMM@HPC project uses UNICORE to create a distributed High Performance/High Throughput Computing environment for multiscale materials modelling. The project has partners from three different expertise areas: computing resource providers, program code developers and material scientists. Code developers provide GridBeans and Application Wrappers for the applications used in the material simulations and install the applications and wrappers on the computing resources of the resource providers. The process of installation requires a close collaboration with the site system administrators. Despite the widespread applicability of UNICORE, sometimes system administrators are still forced to customize UNICORE code to meet specific needs, for example because they want to use a new scheduler version or they wish to exploit fully all the features of the scheduler. Prompted by the necessity to install UNICORE on CINECAs GPU cluster for the MMM@HPC project, we analysed its characteristics of portability and flexibility and this paper describes our experience. We give a brief overview of the activities that have been carried out in CINECA to help researchers to use UNICORE by describing how we constructed our site configuration, both to accommodate the characteristics of our batch scheduler and the user environment on the cluster.

1 Introduction

UNICORE[1] has been in use for a long time in CINECA[2] since it provides one of the most advanced workflow services for designing complex scientific applications. Communities currently utilizing UNICORE at the centre include researchers of the MMM@HPC[4] project who use the services for performing material science simulations. They chose this software because in computational materials science researchers often employ many program codes since simulations are based on different physical models defined by experts working in different research fields. This means that simulations have to integrate multiple applications and in order to achieve this an infrastructure is needed which allows users to combine multiple steps into a single workflow and submit this workflow for computation. This can be accomplished by middleware tools which allow the CINECA High Performance Computing (HPC) infrastructure to be accessed in a transparent way, hiding the details of the computer architecture and the implementations of the application codes from the users.

UNICORE provides an easy-to-use graphical interface to CINECA's computational resources, which in the case of MMM@HPC is principally the IBM GPU cluster (PLX),

while CINECA UNICORE Services are used via the UNICORE Rich Client (URC), extended with additional graphical plugins (GridBeans). The experience gained collaborating with MMM@HPC researchers has been useful for improving our knowledge of UNICORE, since by working with them we have been able to evaluate objectively the potential strengths and weaknesses of this tool. In this paper we will not explore the scientific models, but instead give a simple overview of the main issues that a system administrator has to solve in order to configure a new UNICORE server instance, trying to match the infrastructure characteristics with the user requirements. Further considerations about the scientific models and applications used in the project are described elsewhere[5,6].

In the next sections we present results of our experience: first we will outline the approach used by MMM@HPC scientists to develop their own applications; then we describe which resources have been provided from CINECA and how the CINECA UNICORE Site has been configured. Following this we will describe the work done in order to adapt UNICORE services to the CINECA infrastructure. Finally we will summarize our efforts and try to make some suggestions for further improvements.

2 The MMM@HPC Project and UNICORE

In order to make state-of-the-art multiscale materials modelling feasible for end-users, several approaches have been adopted. First, the multiscale model is stripped down to multiple physical sub-models that are loosely coupled in a directed graph called a workflow. The individual sub-models, called steps, can be solved using different standard simulation codes depending on the physical scale. For example, to describe electrons and related properties on the atomic scale Quantum Mechanical models (QM) are used. A workflow is created by the workflow editor of the URC into which the simulation codes are integrated via application interfaces, the so-called GridBeans, providing a Graphical User Interface (GUI) to collect user input as well as functionality to generate a JSDL job description. The workflow is submitted to the UNICORE workflow engine and monitored through the Grid Browser of the URC, which also collects the results of the simulation.

In the MMM@HPC project several GridBeans have been developed in order to implement a multiscale model for Organic Light Emitting Diodes (OLED)[5]. The development of the workflow and the GridBeans is described in a paper of the current issue[6], while the GridBeans have been made available on the MMM@HPC Update Site[7] where also the workflow description has been made available for download. These components can be installed very easily into the URC by the user. Figure 1 shows a screenshot of the current OLED workflow inside the URC composed of several GridBeans from the MMM@HPC Update Site.

In order to make it possible to execute parts of a workflow on different UNICORE sites (other partner sites are CSC, KIT and KIST) a trust assertion had to be made on the sites in order to allow cross-site processing. For each simulation code supported by the workflow, the UNICORE site administrators create an entry in the incarnation database (IDB) of the UNICORE/X server. Once this is done the associated application GridBean can then submit jobs to the corresponding site. The IDB also defines the local execution environment of the computing system (e.g. SP6 and PLX, in the case of CINECA), and particularly the executable search paths (or alternatively environment modules), which have to be configured. On the HPC system, the corresponding application and the application wrapper

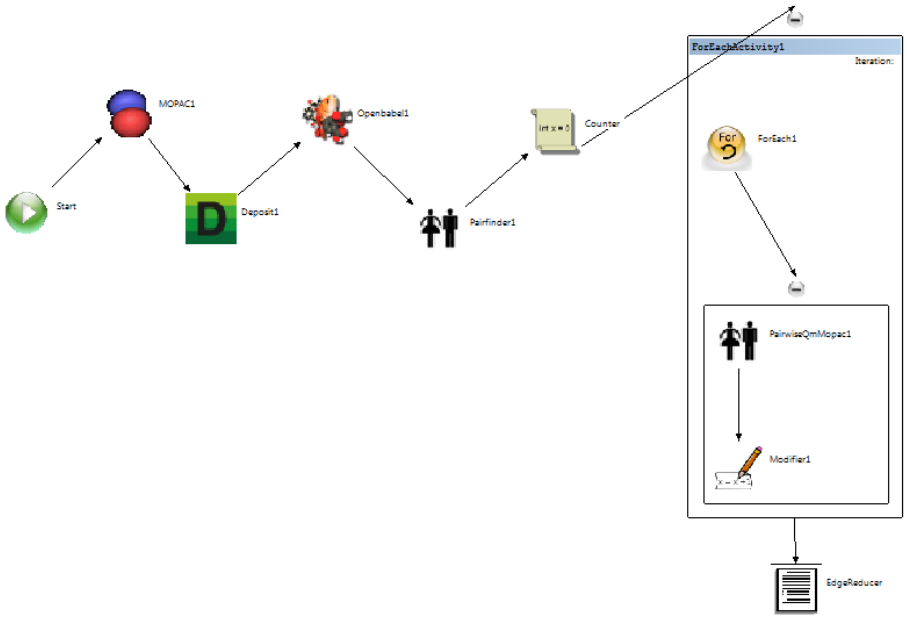

Figure 1. Screenshot of the current OLED workflow taken from the workflow editor of the UNICORE Rich Client.

(based on OpenMolGRID) had to be installed and made available to the user.

Now we proceed by explaining in more detail the key features of the CINECA infrastructure and how it has been configured to provide MMM@HPC users with proper execution environments and access to resources.

3 UNICORE CINECA Site Overview

CINECA is a partner in PRACE[3], the pan-European Research Infrastructure for High Performance Computing, where UNICORE has been adopted as part of the middleware stack. For this reason CINECA hosts a PRACE UNICORE server which runs on an IBM SP6 cluster. The server is constantly monitored and the URI is published in the CINECA UNICORE PRACE Registry. Recently also an additional UNICORE server instance has been deployed on our hybrid Intel Westmere/GPU cluster (called PLX), which will substitute the SP6 instance when the SP6 cluster is decommissioned in the second quarter of 2012. However, it should be emphasised that work on configuring the PLX instance was initiated in early 2012, i.e. well before the planned final shutdown of the SP6, since it was discovered that some crucial components of the MMM@HPC software stack (e.g. the OpenBabel file converter) could not be installed on the SP6 due to the non-compatibility of the IBM compilers on this system. An additional motive was to have the possibility of exploiting the GPUs present on PLX, which for some appropriately written applications can

Cluster	Service
Front End Cluster, OS: Debian GNU/Linux 6.0	Gateway, Registry, XUUDB, Workflow engine, Service orchestrator
IBM SP6 Cluster, OS: AIX version 6.1, LoadLeveler update: 4.1.1.5	UNICORE XNJS + TSI
PLX DataPlex Cluster, OS: redHat EL 5.6, PBS version: PBSpro 10.4	UNICORE XNJS + TSI

Table 1. The UNICORE components as they are deployed on the CINECA infrastructure.

increase the performance by a factor of ten or even more in favourable cases. (It is worth noting that the peak performance of the whole cluster is about 30 Tflops if only the Intel Westmere cores are used, but this rises to above 300 Tflops if the GPUs can be exploited as well.) Thus, the needs of the MMM@HPC project have accelerated the installation of UNICORE on PLX, as well as providing a useful testbed for the configuration. Figure 2 shows the CINECA UNICORE Services as they are deployed on our infrastructure. The Front End Cluster (FEC) hosts the UNICORE Gateway, the Service Registry and the XUUDB as PRACE standard services, in addition to the Workflow engine and the Service orchestrator which have also been installed here. The Workflow service is published in the same UNICORE Registry as the other services and it is listed in the *connection.properties* file of the UNICORE Gateway. The UNICORE XNJS and the perl TSI daemons run on the SP6 and PLX login nodes. Table 1 shows a summary of the main site components.

UNICORE version 6.4.2 was installed for all UNICORE components. Since most of the workflows were run with the PLX UNICORE server, the considerations reported below are related solely to the PLX UNICORE instance.
In the current storage configuration job output files are stored in the GPFS directory: "/plx/prod/unicore/FILESPACE". This has been specified in the UNICORE *xnjs.xml* configuration file and from this directory users are able to retrieve output of their jobs using the URC interface, by simply clicking on their working directory.
In order to execute their applications MMM@HPC researchers first moved their software tools inside different GPFS scratch directories, for example:

```
/gpfs/scratch/userexternal/.../mopac2009
```

Then each executable path was configured, as specified before, in the UNICORE IDB configuration file in order to allow the URC to select the available applications. In the next section we show how the PLX UNICORE server of CINECA was configured to satisfy essential user and infrastructure requirements.

4 The PLX UNICORE server configuration

Once it has been installed, the UNICORE server was configured to match the underlying infrastructure and user requirements. Some changes to the UNICORE XNJS IDB file were needed and are described as follows:

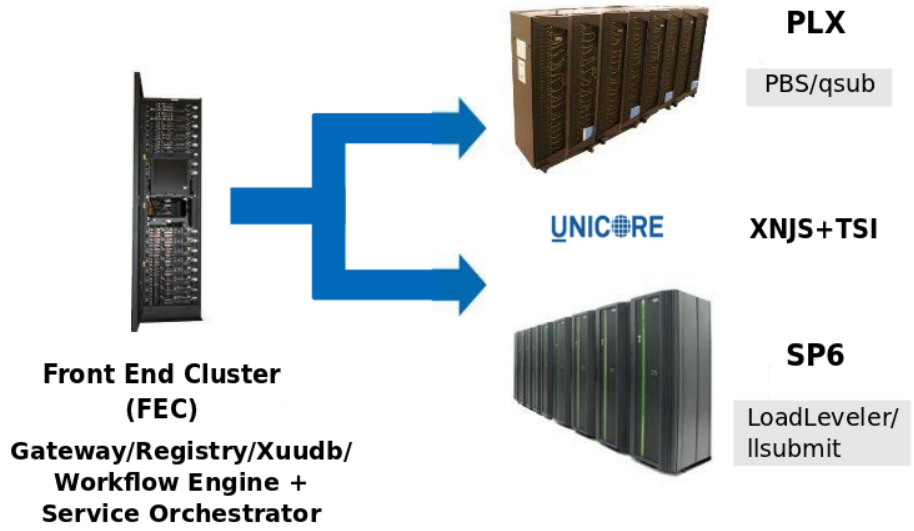

Figure 2. The CINECA UNICORE site showing the Front End Cluster Services (Gateway, Registry, XUUDB, Workflow Engine, Service Orchestrator) and the UNICORE XNJS server.

4.1 The applications

First of all, the applications needed for computation were configured inside the idb file:

```
1. MOPAC
2. Deposit
3. Pairfinder
4. OpenBabel
```

This was done by adding entries in the IDB file as xml elements and setting the appropriate executable paths, for example for MOPAC:

```
<!-- MOPAC -->
<idb:IDBApplication>
  <idb:ApplicationName>MOPAC2009</idb:ApplicationName>
  <idb:ApplicationVersion>1.0</idb:ApplicationVersion>
     ...
    <jsdl:Executable>/gpfs/scratch/userexternal/.../mopac2009
     ...
  </jsdl:POSIXApplication>
</idb:IDBApplication>
```

Now each application can be selected by users from a list box in the URC.

4.2 The Execution Environment

In the current implementation for MMM@HPC only serial jobs are being submitted but since in the future also parallel jobs will be considered, it will be necessary to specify additional parameters such as the number of cores required. It should be emphasised however that the parallel environment of PLX is very different to that of the SP6. For example, when an application compiled with MPI is executed on the SP6, the parallel launcher program, *poe*, is automatically invoked to launch the program on the compute nodes regardless of whether the user included it or not in the job script. For PLX this is not the case and the *mpirun* launcher program must be explicitly supplied in order to run the job correctly. A further complication is that currently there are two MPI implementations on PLX, OpenMPI and IntelMPI, each with its own mpirun command whereas on the SP6 there is only one (namely the IBM implementation).

In order to solve this problem we've created an ExecutionEnvironment inside the IDB file, so that the mpirun command is invoked if the environment is specified from the URC.

```
<ee:ExecutionEnvironment xmlns:ee="http://www.unicore.eu...">
 <ee:Name>MPIRUN-intel-1.4.2</ee:Name>
 <ee:Version>1.0</ee:Version>
 <ee:Description>Runs the user's command...</ee:Description>
 <ee:ExecutableName>path-to-mpirun/mpirun</ee:ExecutableName>
 <ee:CommandlineTemplate>
    #EXECUTABLE #ARGS #USERARGS #USERCOMMAND\
 ...
```

We note that the ExecutionEnvironment has been called MPIRUN-intel-1.4.2 since this corresponds to the most commonly used MPI environment on PLX.

4.3 Physical cores

A new resource has been added into the IDB file as an xml element. The name of the resource is *nCores* and it represents the number of physical cores for each CPU (processor). This was needed since the URC allows users only to specify the number of CPUs/processors per node while we need to select also the number of cores (max 12) per node.

4.4 GPUs

As mentioned above, the availability of GPUs (two per node) is a key feature of PLX, since a growing number of applications are able to exploit them and demonstrate significant performance increases when compared to the non-accelerated versions. In order to use the GPUs within a PBS job it is strongly recommended to request them explicitly within the resource line of the batch script, for example:

```
#PBS -l select=1:ncpus=12:ngpus=2
```

If the *ngpus* parameter is not specified then for PLX, since all nodes are equipped with GPUs, the job may still run correctly but this is not guaranteed and in any case for

clusters where only a subset of the nodes has GPUs, it would mean that only conventional core resources would be allocated. For that reason we configured the IDB file to add the resource nGPUs:

```
<idb:Resource xmlns:idb="http://www.fz-juelich.de/.../idb">
  <idb:Name>nGPUs</idb:Name>
  <idb:Type>int</idb:Type>
  <idb:Description>Number of GPUs per node</idb:Description>
  <idb:Min>0</idb:Min>
  <idb:Max>2</idb:Max>
  <idb:Default>0</idb:Default>
</idb:Resource>
```

In this way users are now able to select the GPUs number directly from the URC.

4.5 The batch queues

Finally a choice was introduced to specify a selection of batch queues:

```
<idb:Resource xmlns:idb="http://www.fz-juelich.de/.../idb">
  <idb:Name>Queue</idb:Name>
  <idb:Type>choice</idb:Type>
  <idb:Description>The batch queue to use</idb:Description>
  <idb:Default>parallel</idb:Default>
  <idb:AllowedValue>parallel</idb:AllowedValue>
  <idb:AllowedValue>longpar</idb:AllowedValue>
  <idb:AllowedValue>debug</idb:AllowedValue>
</idb:Resource>
```

In the following section we now show in detail the changes made to the UNICORE Target System Interface (UNICORE TSI) to make it work correctly on our PLX cluster.

5 The UNICORE TSI

The UNICORE TSI version initially installed on the PLX did not match perfectly the underlying infrastructure characteristics and there was no TSI compatible with the scheduler version on the PLX (PBSpro 10.4). In order to start and run the UNICORE server on the PLX correctly we had to modify the TSI perl code creating a new version of the PBS TSI and these changes are described in the next subsection.

5.1 Code Fixes

The *Submit.pm* module was the only file modified and these are the main changes we have done:

1. All the user parameters are now collected inside an executable bash script, instead of adding them directly to a single PBS command. The executable produced represents the final job description and once it has been created the TSI submits it on the batch scheduler to run the job. This was done for convenience and because it is not always easy to send all the parameters to PBS as a single command.

2. The 'memory' and 'job time' requirements have been modified accordingly to the specific PBS version syntax. For example the job time now is specified in this way:

```
# Job time requirement. Wallclock time in seconds.
#$time = "-lT $time -lt $time";
...
$time = "#PBS -l walltime=$hours:$mins:$secs";
```

3. An entry has been added to set the *ncpus* parameter which defines the number of physical cores:

```
"#PBS -l select=1:ncpus=$ncores";
```

   This code is not sufficient though because with *select=1* we have specified only one "chunk", i.e. one allocation, which on PLX cannot be more than 12 physical cores. In addition, for MPI jobs we need to set the *mpiprocs* parameter which is normally set to *ncpus*. To allow an arbitrary number of tasks to be specified by the user we have decided to adopt an algorithm which assigns as many cores as possible to one chunk and puts the rest into a second chunk, as described by the following pseudo code:

```
# pseudo code
if $ncores <=12
   $resource="#PBS -l select=1:ncpus=$ncores:\
   mpiprocs=$ncores"
if $ncores>12 {
   $nchunks=int($ncores/12);
   $rem=$ncores % 12;
## if we have an exact multiple of 12 cores\
   use chunks of the same size
   if $rem=0
   $resource="#PBS -l select=$nchunks:ncpus=12:mpiprocs=12";
##  add any remainder to a second chunk
   if $rem >0
   $resource .= "+ncpus=$rem:mpiprocs=$rem";
   }
```

   The code added here is specific for our scheduler and is needed because of the particular syntax required in our PBS version.

4. We have introduced the *ngpus* parameter inside the perl code, in order to set correctly the number of GPUs requested. This can be added to the resource string created in the previous step. For example:

```
"#PBS -l select=1:ncpus=$ncores:ngpus=$ngpus";
```

5. Added a piece of code to invoke a local utility (the 'saldo' script), which is needed to identify the project budget for that user and was done to resolve an issue related to CINECA's accounting system. In CINECA the *account_no* parameter is used to specify to PBS which budget is being used so that PBS can verify if the user can submit jobs on that machine. Since it is not possible to specify the *account_no* parameter in the job description from the URC, we changed the TSI code in order to invoke the 'saldo' script which returns the project account for that user. In this way we have been able to allow MMM@HPC users to submit jobs on PLX.
   Using this script is convenient, since it reveals automatically the user's budget identifier without the need to specify it manually.

```
my $saldo = `/cineca/bin/saldo -b`;
if ($saldo ne "username not existing")
{
     my @lines=split(/\n/m,$saldo);
     chomp @lines;
     my @fields=split(" ",$lines[5]);
     $account_number=$fields[0];
     if ($account_number ne "")
     {
       $account_no = "#PBS -A $account_number";
       debug_report("account number: $account_no\n");
     }
}
```

5.2 Demonstration and Verification of Installation

In this section we describe the latest working installation, with Figure 3 showing the PLX UNICORE XNJS server as it is published in the PRACE Registry. The job properties window contains the parameters for the job application setup: in particular the new parameters introduced are: *Execution_Environments*, *Queue* and *nGPUs*. Before submitting a job the user can select the correct Execution Environment simply clicking on the property value: when the property has been clicked an external window appears as shown in Figure 4. Here the user is able to set Precommands and Postcommands and to overwrite the default IDB file settings. Finally the user can run his job selecting the PLX UNICORE server from the URC. After deployment and configuration we have tested our infrastructure by submitting a simple mpi test job, selecting the executable and providing the arguments:

```
/gpfs/scratch/userinternal/aem0/mpi_test -np 4
```

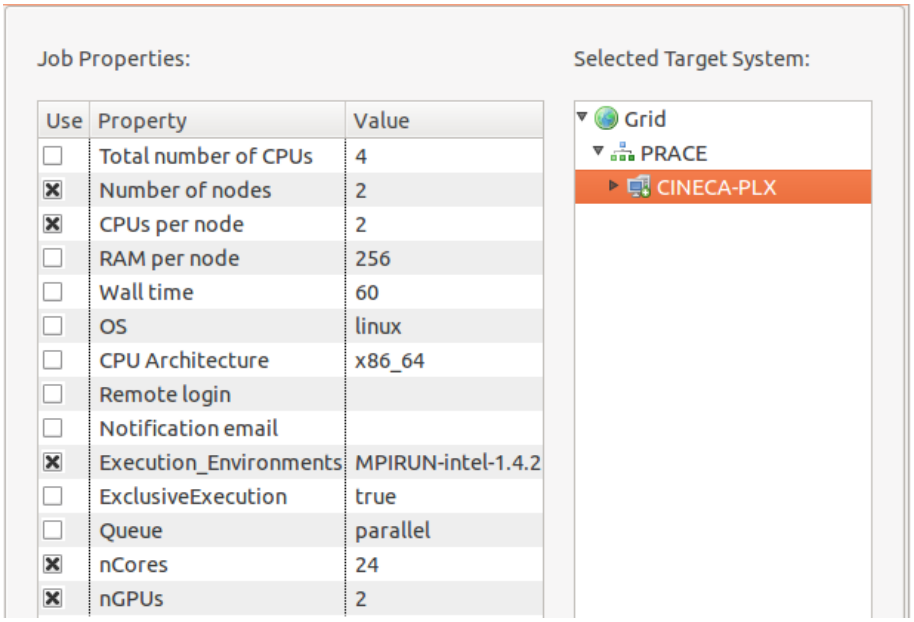

Figure 3. The UNICORE PLX Service as it appears in the URC graphical interface.

This job runs an mpi test example application and returns correct output (in this case simply the number of tasks available to the MPI program).

In a talk on multiscale simulations of complex materials at the ISGC 2012[10] in Taipei we performed a live demonstration of the OLED workflow. This demonstration included several applications (MOPAC2009, Deposit, OpenBabel, Pairfinder) which had been installed on the PLX site and configured in the IDB of the UNICORE server. We demonstrated the submission of an OLED workflow (Figure 1) to the PLX site which calculated the charge transport in an Alq3 layer[5] consisting of 25 molecules. All relevant parameters and input files had to be staged from Taiwan to the CINECA workflow service in Italy. During the demo we monitored the workflow progress and the individual job status and fetched the output files from the working directories. The demo showed a successful interoperation between the OLED workflow components in the URC and the UNICORE infrastructure provided by CINECA over a distance of more than 10,000 km.

6 Remarks and Ongoing Work

During the configuration procedure of UNICORE on PLX a number of limitations of the current implementation were identified, some of which may be resolved with a further programming effort but also some of which are inherent limitations of UNICORE. We shall now look at some of these:

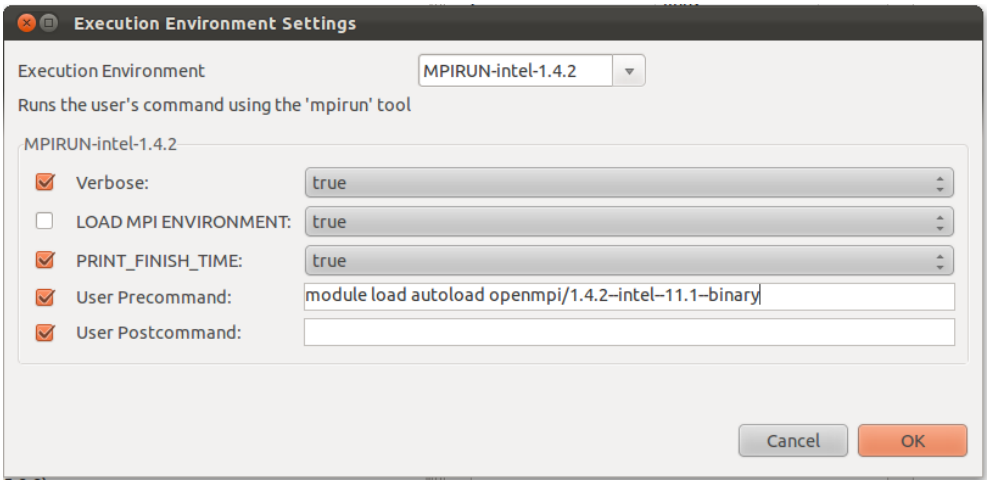

Figure 4. The Execution Environment Settings panel where users can set their own Precommand and Postcommand.

- There is no "simple way" to change the TSI - system administrators are forced to modify the perl code to allow for particular customisations for different schedulers or local configurations.

- With UNICORE we cannot use the full flexibility of PBS with respect to resource specification. For example, it is not possible to specify arbitrary "chunk" sizes as in the following PBS example:

  ```
  #PBS -l select=2:ncpus=8:mpiprocs=8+1:ncpus=5:mpiprocs=5
  ```

  Here we have requested 2 chunks of 8 cores and 1 chunk of 5 cores. It is usually preferable to allocate entire nodes of PLX, i.e. chunks of 12 cores, but there might be cases where having many chunks might be advantageous (e.g. to maximise the number of GPUs available). However, this resource allocation cannot be specified with the URC.

- It should be noted that in addition to the *Submit.pm* module we also found it necessary to modify another TSI program, *GetStatusListing.pm*, which monitors the jobs in the batch system and reports the status for each one. It does this by periodically parsing the output of the appropriate scheduler command, which in the case of PBS is qstat. The precise output of this command, however, is very sensitive to the arguments given to it and with the original version of the code often the parsing failed to give the correct job status. In these situations the UNICORE/X server assumed the jobs had completed even if they were still executing. This has now been corrected at CINECA but clearly each site will have to check their own installations.

7 Conclusions

Once again UNICORE middleware has been proved useful in providing users with a user-friendly graphical interface for accessing CINECA supercomputing resources. UNICORE has been used also to extend our computing infrastructure, helping MMM@HPC users to perform their simulations. The limits we encountered do not affect UNICORE basic mechanisms, but rather we have highlighted some problems which arose from the necessity to fully exploit the features of our machine, the IBM PLX cluster, in terms of hardware capabilities (the PLX is a GPU-based system) and because the test applications we ran made it necessary to exploit the characteristics of the particular scheduler version (PBSPro 10.4). These limitations have essentially been overcome with some configuration changes and some modifications of the TSI perl code. However, a path towards a general solution to the problems exposed in this paper, as for example fast scheduler customization or multiple chunk allocation, is not yet completely clear. The present work anyway has highlighted those user needs we have to take into account when we plan to submit this kind of applications through the UNICORE middleware. The collaboration with the MMM@HPC project has been extremely useful since it has allowed us to test UNICORE with a broad range of real user requirements and applications and the experience gained will be essential for providing workflow services to other research communities.

8 Acknowledgements

This work has been funded by the 7th Framework Programme of the European Commission within the Research Infrastructures with grant agreement number RI-261594, project MMM@HPC.

References

1. Streit, A.; Bala, P.; Beck-Ratzka, A.; Benedyczak, K.; Bergmann, S.; Breu, R.; Daivandy, J.; Demuth, B.; Eifer, A.; Giesler, A. et al., *UNICORE 6 — Recent and Future Advancements*, Ann. Telecommun. 65, 757–762 (2010), `http://www.unicore.eu`.
2. CINECA `http://www.cineca.it/`.
3. PRACE `http://www.prace-ri.eu/`.
4. Multiscale Material Modelling on High Performance Computer Architectures (MMM@HPC) `http://www.multiscale-modelling.eu/`.
5. Kondov, I.; Maul, R.; Bozic, S.; Meded, V., and Wenzel, W.; *UNICORE-Based Integrated Application Services for Multiscale Materials Modelling*, In: Romberg, M.; Bala, P.; Müller-Pfefferkorn, R., and Mallmann, D. (Eds.), *UNICORE Summit 2011 Proceedings, 7–8 July 2011, Torun, Poland, 2011*, IAS Series, Vol. 9, pp. 1-10, Jülich 2011.
6. Bozic S.; Kondov I.; Meded, V.; Wenzel, W.; *UNICORE based Workflows for the Simulation of Organic Light-Emitting Diodes* (ibid.).
7. MMM Update Site: `www.multiscale-modelling.eu/update-site`.
8. GPUs `http://en.wikipedia.org/wiki/GPGPU`.

9. OpenMP http://en.wikipedia.org/wiki/OpenMP.
10. Bozic S.; Kondov I.; Meded, V.; Wenzel, W.; *Talk:Multiscale simulations of complex materials basing on UNICORE workflows*, ISGC, 29 February 2012, Academia Sinica Taipei.

UNICORE Based Workflows for the Simulation of Organic Light-Emitting Diodes

Stefan Bozic[1], **Ivan Kondov**[1], **Velimir Meded**[1], **and Wolfgang Wenzel**[2]

[1] Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT)
76128 Karlsruhe, Germany
E-mail: {stefan.bozic, ivan.kondov, velimir.meded}@kit.edu

[2] The Institute of Nanotechnology (INT)
Karlsruhe Institute of Technology (KIT)
76128 Karlsruhe, Germany
E-mail: wolfgang.wenzel@kit.edu

Integration of different program codes into automatic workflows is essential to perform whole-device simulations based on multiscale models. We report on the successful implementation of an automatic workflow for simulation of organic light-emitting diodes using the UNICORE middleware and GridBeans technologies. We discuss the aspects of scalability and data exchange in the created workflow and suggest efficient solutions in different use cases. In addition, we describe the provided infrastructure and toolkits for development of GridBeans, wrappers and workflows. The realization of the workflows provides a proof of concept and, due to the low-cost high-efficiency approach, enables deployments in industrial research.

1 Introduction

Multiscale modeling of materials plays an increasingly important role in development of novel materials, for example of energy conversion and storage devices. It is a highly scalable approach which combines established physical models on several length and time scales (see Refs. 8, 10, 12 for recent reviews). Nevertheless, to perform computer simulations of real devices using multiscale models is still very difficult due to the lack of a platform integrating diverse program codes performing the individual model steps, and providing seamless access to computing resources and flexible interfaces for modelers and end-users.

An organic light-emitting diode (OLED) is a nano-structured device, in which the emissive electroluminescent layer is a thin film of organic material, emitting light in response to applied voltage. This type of diodes can be used in novel manufacturing processes based on inkjet printer. The usage of OLEDs on light weight and flexible substrates allows the production of flexible displays. Furthermore OLEDs provide good power efficiency and a fast response time.

Beside these advantages current OLEDs have many characteristics that need to be improved. Manufacturing is still very expensive because of novel process steps. Also the operation lifetime, the colour balance and the power consumption are not optimal and need to be improved. On the other hand, the research on novel nano-structured materials, such as OLEDs, in laboratory experiments is very costly and time consuming. Thus these experiments are suited to be carried out in silico modern high performance computing

(HPC) and high throughput computing (HTC) resources using complex multiscale models. The target end-users of the models are experimentalists and industrial researchers who have insufficient experience with modeling and computer simulation. A service-oriented platform connecting the end-users with existing HPC/HTC resources and integrated ready-to-use models is still lacking. The development and enabling of such computer simulation services is the main objective of the MMM@HPC project[1].

In this paper we report our progress in developing software tools, including GridBeans, pre- and post-processors, wrappers and workflows, in particular to implement and perform proof-of-principle simulation of OLEDs. Further aspects such as experiences with operating the developed infrastructure will be provided in another paper[5].

2 Methodology

Performing automatic multiscale material modeling simulations on HPC resources is not a trivial task and multiple challenges, regarding security, reusability, data complexity, licensing issues, and high computational demands, have to be solved. The UNICORE middleware provides a multi-layered service-oriented architecture which addresses all of these challenges. A major task of UNICORE is to provide easy access to HPC resources. With x.509 certificates and the Security Assertion Markup Language (SAML) it uses state-of-the-art techniques to build up a secure environment for accessing computing and storage resources, and handling simulation data. One of the principle reasons for MMM@HPC to use UNICORE as key technology is the workflow support[6] allowing users to set up a coherent multi-step model for which a simulation is executed automatically. A workflow is developed once by a workflow architect/designer and can then be forwarded to material scientists who use the workflow in their applied research.

A UNICORE workflow[20] consists logically of a control flow that defines the order of execution of the individual simulation steps and a dataflow describing the data exchange between the steps. The UNICORE Rich Client (URC) provides a workflow editor to specify these two flows. The constituents of a UNICORE workflow are GridBeans[14] providing a graphical user interface for simulation codes. These applications are installed on distributed computer resources and can be accessed via UNICORE services.

In a recent paper[11] we have described the principles of our approach for the model of charge transport in thin layers and drawn a roadmap for full implementation of the OLED workflow. The simulation protocol, as shown in Figure 1, consists of several modeling steps on different time and length scales (QM= Quantum Mechanics; MM=Molecular Mechanics; CG=Coarse Grained; FEA= Finite Element Analysis). Each step requires a specialized simulation code that can fulfil the particular needs of the physical model. The right side of Figure 1 shows a concrete implementation as UNICORE workflow. Each workflow step is assigned to an application GridBean that has been developed in the project. For instance the geometry optimization step in the UNICORE Workflow is realized with the MOPAC GridBean. The constructed workflow also includes the dataflow between the individual GridBeans. Detailed manuals on the development of GridBeans and UNICORE Workflows can be found in Refs. 3, 19.

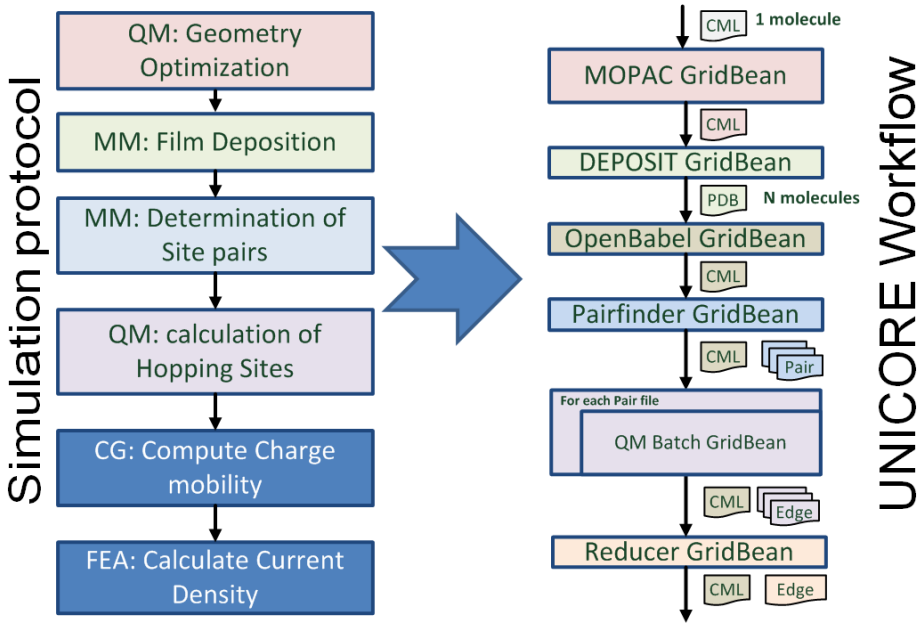

Figure 1. Mapping of the OLED simulation protocol based on a multiscale model to a UNICORE workflow

3 Calculation of Hopping Sites

For description of long-range charge transport in the amorphous molecular layer we use a model for hopping transport. In this section we will discuss in more detail the calculation of the hopping sites (see Figure 1) which are part of this model. Each hopping site (also referred to as site pair) contains two molecules called monomers. Additionally the two molecules are aggregated to an entity called dimer. Because the number of molecules in an amorphous layer can go up to one million the Pairfinder produces a very large amount of site pairs. Moreover, we need three quantum mechanical (QM) calculations for each hopping site. One QM calculation can require several CPU hours which varies depending on the precision of the result. This is why, the QM calculations of site pairs is a very resource-demanding step.

Let us assume that the Pairfinder step has found one million sites. Then three million QM calculations have to be carried out in the next step. Running each QM calculation in a single job will result in huge network traffic. In addition, the job submission/management becomes a bottleneck because every job must be processed by the UNICORE services and finally by the target systems which increases the overall time of a simulation.

In the following we shall discuss the pros and contras of two alternative approaches, sketched in Figure 2, to tackle this challenge. Another approach to alleviate this bottleneck is the XML space mechanism in UNICORE to handle high job throughput[16] but it is not used in this work.

3.1 Batch of QM jobs

One approach is based on so-called pair files. The Pairfinder application produces simple text files which include a list of IDs referencing the pairs of molecules forming hopping sites, for instance (2-4), (5-9), (2-12). The IDs are references to molecules in the CML file (see next section) containing the structure of the whole amorphous layer. The maximal number of pairs in a single pair file can be defined as parameter for the Pairfinder application and a change of this value will normally result in a variable number of pair files. In the workflow (cf. Figure 2) a *foreach* loop is iterating over the list of pair files. In the loop each pair file is redirected as input for a QM Batch GridBean. This GridBean submits a single job per pair file to a resource where a special application wraps the quantum mechanical calculation. In one limiting case, each batch of QM jobs has to perform as many QM calculations as three times the number of pairs, and in the other case with one pair per pair file, only three QM calculations. The job also processes the QM results and instantly creates edge files. The latter contain parameters used in the Coarse Grained method to compute charge mobilities in the next workflow step. The only function of the intermediate Reducer step is to collect all edge files and merge them to a single one. This batch approach is efficient for short QM calculations whose execution time is up to a few minutes. By bundling several QM calculations in one job we can minimize the network traffic and the management effort on side of the middleware and the batch systems. It will also reduce the overall overhead in time consumption that comes with a less amount of jobs.

One disadvantage in using pair files is the need of special GridBeans and applications for each quantum mechanical code used in the workflow. This increases the development effort and decreases the usability of existing GridBeans. For example we developed a special QM Batch GridBean for MOPAC2009 that supports a pair file as input. On the server side we need a special MOPAC batch application wrapper that processes the pair file, invokes the MOPAC2009 calculations and evaluates the results producing a single edge file.

3.2 Single QM jobs

In the second approach every QM calculation is mapped to a single job. For this purpose the Pairfinder application creates CML files for the monomers and dimers of the pertinent hopping sites. A *foreach* loop iterates over these CML files and submits for each file a separate QM job. In this case it is not important which QM application (e.g. MOPAC or TURBOMOLE) is used. The only requirement is that this GridBeans supports CML format for input.

This approach is advantageous if a single QM calculation takes more than a few minutes. In this case the overhead in managing a job by the grid middleware and the batch system stands in an arguable ratio with the running time of the QM calculation. A significant drawback of this approach can be the large amount of files created by Pairfinder. The number can grow up to one million files and more. These files must be transferred over the network to the workflow storage and then again must be transferred to the working directories for the individual QM calculations. Each QM calculation also creates one or more output files which has to be transferred and processed by an application-specific QM Reducer GridBean.

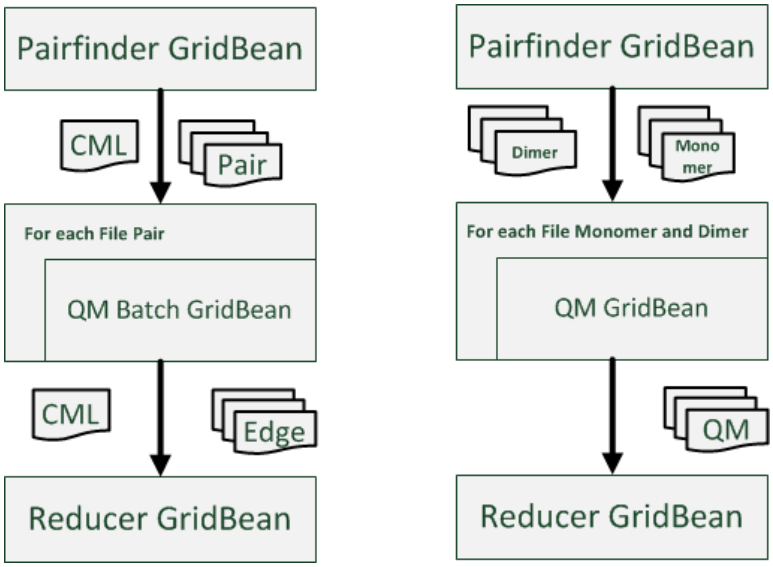

Figure 2. Two approaches, batch of QM jobs and single QM job, for the calculation of site pairs shown on the left and the right diagram, respectively.

4 Handling the Dataflow

Handling the flow of heterogeneous modeling data in the OLED workflow posed a big challenge for us because the UNICORE middleware and the GridBean technology do not provide a ready-to-use solution. When connecting different GridBeans inside a workflow, input and output files can be declared as workflow files, which are stored on a UNICORE workflow storage. The process of data transfer from step to step via such files is called dataflow. UNICORE provides the workflow storage as a file container solution without functionality to handle the data structure and data interfaces to applications in individual workflow steps. To tackle this deficiency we use a combination of CML[2], Open Babel[9,13] and the OpenMolGRID library[7,17,18]. Currently, a generic data model with common interfaces to applications and to storage (via APIs) is subject of intensive development and first results will be available soon[4].

4.1 CML and Open Babel

The essential task is to specify a basic data model that can be derived by the input/output data of the simulation applications. Almost all codes use atomistic data describing molecular and crystal structures. A de facto standard for representing structural information is the Chemical Markups Language (CML). This language is based on XML and is extendable but, unfortunately, only few applications use it. This is why a mechanism to convert between CML and application-specific file formats was necessary.

A simple way to convert between several chemical file formats is provided by Open Babel. The toolbox can read, write and convert over 110 chemical file formats, is open

source and offers an API to integrate into individual codes. To facilitate the integration and to ensure maximum reusability in our workflows a GridBean for Open Babel has been developed. The GridBean offers a GUI where it is possible to specify the file format to convert from and the file format to convert to. Additionally, the user has an input field where they can specify additional program parameters that are available when executing the application on a command line. Unfortunately, several codes used in the OLED workflow like TURBOMOLE and DEPOSIT use proprietary file formats that are not supported by Open Babel.

4.2 OpenMolGRID

A more generic approach to handle different file formats is given by the OpenMolGRID project[7,17,18]. This project has dealt with similar dataflow challenges in automatic workflows for large-scale molecular design and engineering problems. The OpenMolGRID library provides a Java API including a model for atomistic data, a parser/writer for application data to the model, a process wrapper and an API for operations on the model.

For the OLED workflow we adapted and substantially extended the library (see Ref. 4 for more details). Here we will provide a closer look on the runtime environment of the OpenMolGRID library. In Figure 3 the server and client side deployment of the OpenMolGRID library is depicted. Each step of the OLED workflow is implemented by a GridBean. A GridBean collects the user input, including application parameters and input files, and sends this data to the UNICORE server. Instead of invoking the application directly, the UNICORE/X calls the responsible ApplicationWrapper that is based on the ApplicationWrapper super-class from the OpenMolGRID library. The ApplicationWrapper has a three-step life cycle (see Figure 3):

1. Pre-processing phase: The ApplicationWrapper validates the user input files and parameters submitted by the URC/GridBean. If the input data is accepted the ApplicationWrapper triggers a Parser which reads the CML input file that is created by a previous workflow step. The Parser builds the atomistic model which is used to create the application-specific input file.

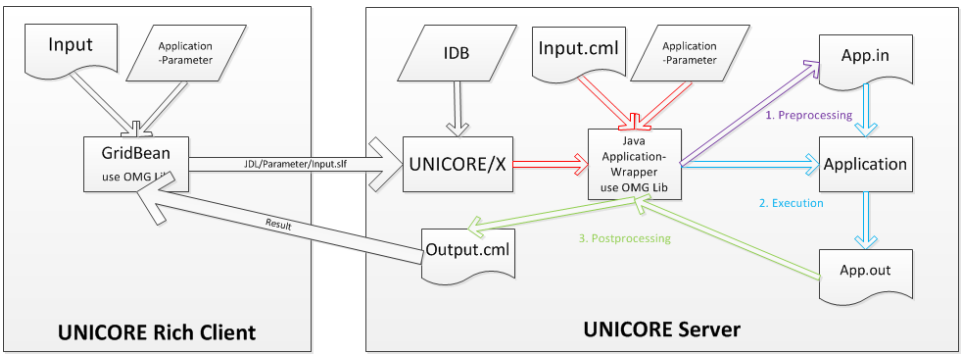

Figure 3. OpenMolGRID library is used on both UNICORE client and server sides.

2. Execution phase: The ApplicationWrapper starts the application as a new process. The ProcessWrapper observes the stdout and stderr streams and thus can act on specific events.

3. Post-processing phase: A specific parser class reads the output files and creates an atomistic model. Then, this model is written to a CML file which can be passed to subsequent workflow steps.

5 Developer Infrastructure

Currently, several partners of the MMM@HPC project are involved in the development process of GridBeans and ApplicationWrapper. To manage the source code of these components by multiple developers dedicated MMM@HPC developer services have been established. The infrastructure includes a subversion server for the source code, and a Maven repository for all project and third-party libraries. Furthermore, the project provides an Eclipse update site (MMM@HPC Update Site) where all GridBeans developed for the OLED workflow are published. In Figure 4 all these services are shown schematically.

On the other hand, to facilitate the development process a special MMM@HPC Development Toolkit has been created and distributed to new developers. This is a stand-alone pre-installed and pre-configured development environment based on Oracle Virtual Box image containing URC, UNICORE services, TSI for execution and storage, Eclipse and further development tools necessary for GridBean and ApplicationWrapper development. Also a developer's manual for using the toolkit is provided.

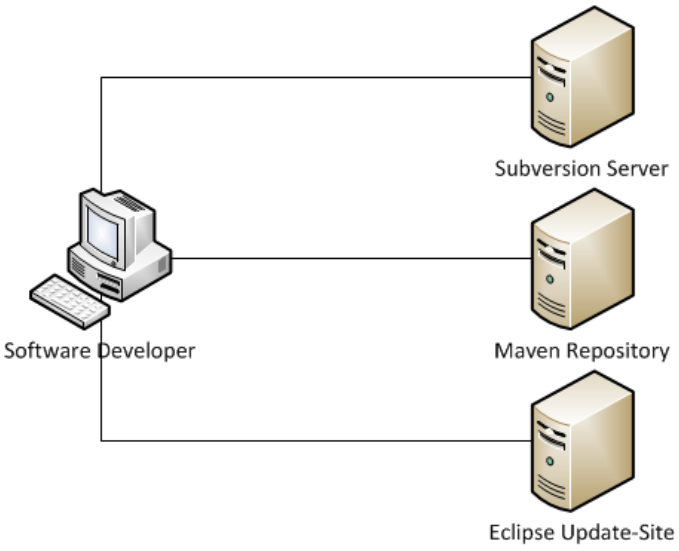

Figure 4. Developer infrastructure services of MMM@HPC

6 Sharing the OLED Workflow

The aim of developing a UNICORE-based OLED workflow is to give the materials science community a reusable toolbox that supports their work in developing novel nano-materials. Thus an effective dissemination of the workflow components is a high priority task. Sharing the workflow includes in first instance the GridBeans developed. To this end, we use the same approach as used in the Chemomentum project[15].

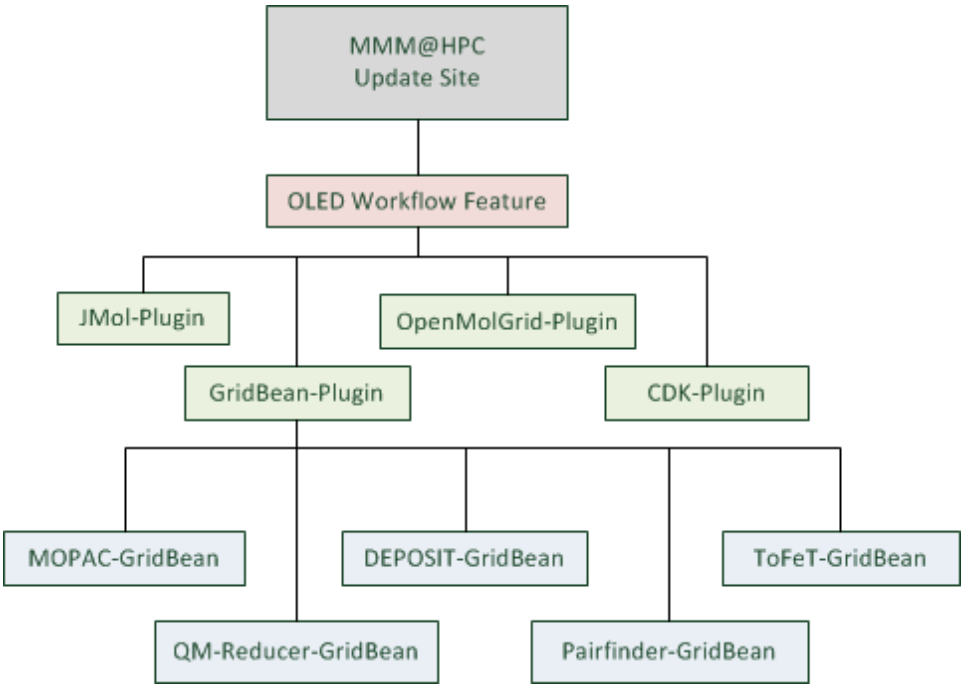

Figure 5. Update Site with the released OLED Workflow Feature, both provided by the MMM@HPC project

The UNICORE Rich Client is a runtime environment for GridBeans. The client is based on Eclipse which comes with a build-in update service that uses the HTTP protocol. One benefit of this service is that updates can be easily distributed from a standard web server. An update site is based on several components like features and plug-ins. More technical details regarding the structure of Eclipse update sites can be found in Ref. 21.

Figure 5 displays the structure of the current MMM@HPC Update Site containing the OLED workflow feature. The feature itself contains several plug-ins. All GridBeans are bundled in a single plug-in (GridBean plug-in). Some GridBeans use third-party libraries for displaying three dimensional (e.g. Jmol plug-in) and two-dimensional (e.g. CDK plug-in) molecular structures by reading and converting chemical formats like CML, PDB or Mol2 (OpenMolGRID plug-in).

Besides sharing the GridBeans it is necessary to distribute the workflow description[20] including the control flow and the involved workflow files. Workflow instances created

with the workflow editor of the URC are stored as projects in the users workspace. A project folder contains amongst others a *.flow file which describes the whole workflow in a XML-based language. For sharing an OLED workflow realization with others, the workflow designers export the relevant projects from the URC workspace as a zip archive. This zip file is stored on the project's webpage where material scientists can download it for further usage. Before importing the workflow archive into the URC, a user has to install the OLED feature from the update site including the GridBeans used in the workflow. One goal in the ongoing development process is to integrate the workflow project as a plug-in of the OLED feature. Then users do not have to take care about possible dependencies between the workflow project and the installed GridBeans in their URC.

7 Conclusion

With the realization of a UNICORE workflow for the OLED simulation protocol we effectively achieved a proof of principle for multiscale models in nano-material research. Particular attention has been paid to the calculation of hopping sites. The number of jobs required for this task can grow to a very large number. We discussed the pros and cons of a batch and a single-job approach to tackle this challenge and showed that both allow a scalable solution in variable usage cases. Future real-data deployments of the workflow in OLED simulations will elucidate which approach performs better.

Data exchange between workflow steps is an important issue. For the OLED workflow we use three different approaches to enable seamless dataflow in heterogeneous applications. The Chemical Markup Language serves as container for all atomistic data transferred in the workflow. Furthermore, with an extended version of the OpenMolGRID library it is possible to write ApplicationWrappers that manage arbitrary file formats and that can be used to process input and output of applications currently not supported by Open Babel.

The developed OLED workflow is shared by the project partners with the materials science community employing the Eclipse update mechanism and provided on our own MMM@HPC Update Site. In addition, the MMM@HPC Development Toolkit provides an all-in-one environment to accelerate development of GridBeans and application wrappers, especially for newcomers. Now, several project partners from Europe and Asia use the provided MMM@HPC development infrastructure to develop new GridBeans and application wrappers for individual workflow steps of further multiscale models for Li ion batteries, polymer-based electronic devices, and carbon-nano-devices.

Acknowledgements

This work has been funded by the 7th Framework Programme of the European Commission within the Research Infrastructures with grant agreement number RI-261594, project MMM@HPC.

References

1. *Multiscale Material Modelling on High Performance Computer Architectures (MMM@HPC)*, www.multiscale-modelling.eu.
2. Chemical Markup Language - CML, http://www.xml-cml.org/, 2010
3. S. Bergmann, *GridBean Developer's Guide*, http://www.unicore.eu/documentation/manuals/unicore6/files/GridbeanDevelopersGuide.pdf, May 2009.
4. S. Bozic and I. Kondov. *Dataflow Management: A Grand Challenge in Multiscale Materials Modelling*, eChallenges e-2012 Conference Proceedings, 2012.
5. M. Carpene, A. Emerson, I. Kondov, and S. Bozic. *Experience with unicore services for multiscale materials modelling*, UNICORE Summit 2012 Proceedings, 30 and 31 May 2012, Dresden, Germany, volume 7, 2012.
6. B. Demuth, B. Schuller, S. Holl, J. Daivandy, A. Giesler, V. Huber, and S. Sild, *The UNICORE Rich Client: Facilitating the Automated Execution of Scientific Workflows*, 2010 IEEE Sixth International Conference on e-Science (e-Science), 238–245, 2010.
7. W. Dubitzky, D. McCourt, M. Galushka, M. Romberg, and B. Schuller, *Grid-enabled data warehousing for molecular engineering*, Parallel Computing, 30(9–10), 1019–1035, 2004.
8. J. A. Elliott, *Novel approaches to multiscale modelling in materials science*, International Materials Reviews, 56(4), 207–225, 2011.
9. R. Guha, M. T. Howard, G. R. Hutchison, P. Murray-Rust, H. Rzepa, C. Steinbeck, J. Wegner, and E. L. Willighagen, *The Blue Obelisk — Interoperability in Chemical Informatics*, Journal of Chemical Information and Modeling, 46(3), 991–998, 2006.
10. T.E. Karakasidis and C.A. Charitidis, *Multiscale modeling in nanomaterials science*, Materials Science and Engineering: C 27(5–8), 1082–1089, 2007.
11. I. Kondov, R. Maul, S. Bozic, V. Meded, and W. Wenzel, *UNICORE-Based Integrated Application Services for Multiscale Materials Modelling*. UNICORE Summit 2011 Proceedings, 7–8 July 2011, Torun, Poland, Mathilde Romberg, Piotr Bala, Ralph Müller-Pfefferkorn, and Daniel Mallmann (Eds.), volume 9 of IAS Series, pages 1–10, Jülich, 2011. Forschungszentrum Jülich GmbH, Zentralbibliothek Verlag.
12. G. Makov, C. Gattinoni, and A. De Vita, *Ab initio based multiscale modelling for materials science*, Modelling and Simulation in Materials Science and Engineering, 17(8), 084008, 2009.
13. N. O'Boyle, M. Banck, C. James, C. Morley, T. Vandermeersch, and G. Hutchison, *Open Babel: An open chemical toolbox*, Journal of Cheminformatics, 3(1), 33, 2011.
14. R. Ratering, A. Lukichev, M. Riedel, D. Mallmann, A. Vanni, C. Cacciari, S. Lanzarini, K. Benedyczak, M. Borcz, R. Kluszcynski, P. Bala, and G. Ohme, *GridBeans: Supporting e-Science and Grid Applications*, Second IEEE International Conference on e-Science and Grid Computing, 2006 (e-Science '06), pages 45–52, 2006.
15. B. Schuller, B. Demuth, H. Mix, K. Rasch, M. Romberg, S. Sild, U. Maran, P. Bala, E. del Grosso, M. Casalegno, N. Piclin, M. Pintore, W. Sudholt, and K. Baldridge, *Chemomentum - UNICORE 6 Based Infrastructure for Complex Applications in Science and Technology*, Euro-Par 2007 Workshops: Parallel Processing, volume 4854 of Lecture Notes in Computer Science, pages 82–93, Springer, 2008.

16. B. Schuller and M. Schumacher, *Space-Based Approach to High-Throughput Computations in UNICORE 6 Grids*, Euro-Par 2008 Workshops: Parallel Processing, volume 5415 of Lecture Notes in Computer Science, pages 75–83. Springer, 2009.
17. S. Sild, U. Maran, A. Lomaka, and M. Karelson, *Open Computing Grid for Molecular Science and Engineering*, J. Chem. Inf. Modeling, 46, 953–959, 2006.
18. S. Sild, U. Maran, M. Romberg, B. Schuller, and E. Benfenati, *OpenMolGRID: Using Automated Workflows in GRID Computing Environment*, Advances in Grid Computing - EGC 2005, volume 3470 of Lecture Notes in Computer Science, pages 464–473, Springer, 2005.
19. UNICORE Team, *Client Video Tutorials*, `http://unicore.eu/documentation/tutorials/unicore6/`.
20. UNICORE Team, *UNICORE Workflow System manual*, UNICORE, 2012.
21. L. Vogel. *Extending Eclipse - Plug-in Development Tutorial*, 2011. `http://www.vogella.com/articles/EclipsePlugIn/article.html`.

Secure Multi-Level Parallel Execution of Scientific Workflows on HPC Grid Resources by Combining Taverna and UNICORE Services

Sonja Holl[1], **Olav Zimmermann**[1], **Bastian Demuth,**[1]
Bernd Schuller[1], **and Martin Hofmann-Apitius**[2]

[1] Jülich Supercomputing Centre,
Research Centre Jülich, 52425 Jülich, Germany
E-mail: {s.holl, olav.zimmermann, b.demuth, b.schuller}@fz-juelich.de

[2] Fraunhofer Institute for Algorithms and Scientific Computing (SCAI),
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Scientific workflows have emerged as a flexible and easily comprehensible way to implement complex multi-step data analysis problems. With surging data volumes and rising complexity of these workflows the demand for parallel, distributed execution of such workflows in high performance computing (HPC) and high throughput computing (HTC) environments has increased significantly.

One of the major challenges when migrating client based tools to large distributed environments is security. Some workflow management systems (WMS) like Taverna, a system widely used in the life science community, provide basic security features. Hence, during the execution of a Taverna workflow on the client individual workflow steps can be securely distributed via UNICORE on the Grid. But WMS generally lack extended security mechanisms, such as trust delegation to support an extended distributed Grid submission mechanism of entire workflows. We describe a novel security propagation mechanism in this paper to enable a secure workflow submission. Additionally, we describe our optimization use case and its performance in an established three tier Grid architecture.

1 Introduction

State of the art scientific research often requires skilled analysis of large amounts of data. Depending on the type of data and the scientific approach various multi-step combinations of scientific methods are required to obtain the desired results. To handle these scientific workflows, scientific workflow management systems (WMS) have become an increasingly popular choice as they allow for easy comprehension, editing, and dissemination of analysis recipes[7]. As complex scientific workflows may incur high computational demands they benefit from the ability to use high-performance-computing (HPC) and high-throughput-computing (HTC) resources, as provided by different types of interoperable e-Science infrastructures[8]. Access to these computing infrastructures is mainly provided via Grid middleware systems, such as UNICORE (UNiform Interface to COmputing REsources)[4], ARC (Advanced Resource Connector)[3], or Globus Toolkit[1]. To enable the use of HPC/HTC resources for individual workflow steps, we have developed a plugin for the WMS Taverna which enables a secure Grid access[2] via UNICORE. A more complex scenario, arising e.g. in scientific workflow optimization set-ups, is the distributed execution of many individual workflows on the Grid. In this scenario not only the architecture setting is more ambitious but also the security demands are more difficult as distributed workflow

execution on the Grid requires a delegation of trust. To support a secure distribution of scientific workflows, e.g. for workflow optimization applications, on the Grid, we have developed a three tier workflow submission architecture that implements a novel embedded security propagation mechanism based on Taverna 2 and UNICORE 6. It utilizes security assertion markup language (SAML) assertions in combination with the UNICORE incarnation database. In section 2 we describe our HPC/HTC scenario based on the distributed execution of scientific workflows. Section 3 shows the realization of the submission mechanism in a workflow management system and section 4 the implementation of the corresponding three tier architecture. Details of our developed security propagation mechanism are described in section 5. Afterwards, we show results of distributed workflow executions in section 6. We conclude our paper with a discussion of results in section 7.

2 Motivation

Scientific workflows may combine various data inputs and contain several different applications, each having its own set of parameters. The number of possible parameter combinations for these workflows provide a considerable challenge for parameter optimization. Additionally, the optimal parameter combination of a workflow may depend on its input data and some parameters will have more influence on the final result than others. To search for optimal parameters of a scientific workflow stochastic optimization is an efficient alternative to brute force parameter screening, becoming unfeasible for systems with many free parameters. Examples for popular stochastic optimization algorithms are Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). These iterative algorithms test a population of workflows in each generation of the search. Each workflow in the population has distinct parameter settings. The rating of a workflow instance is evaluated based on a user specified objective function. These ratings are then used by the optimization algorithm to guide the choice of parameters for the next generation of workflow instances.

When performing this scientific workflow parameter optimization on the client several evolving generations of workflows are created. That means, during the evolution of one workflow generation, one workflow is instantiated several times with differing parameter sets. It means that for each parameter set up the workflow is executed once and the fitness function is stored. To perform the workflow execution in a reasonable time, the individual executions should run in parallel. To lower the workload on the client a shift of the parallel workflow executions to HPC resources is required. An approach to this improved parallel workflow execution support is described in the following.

3 Taverna Grid Workflow Plugin

While there are several workflow management systems that have been used in life science disciplines, they differ in having a client and a server version and coming equipped with a wide range of scientific methods. Taverna is an open source, domain-independent workflow management system written in Java. The Taverna Workbench provides a graphical interface for creating and monitoring workflows, while the Taverna Workflow Engine is responsible for their execution. Workflows in Taverna consist of a user defined arrangement of services. Each service provides input and output ports for the underlying application

that can be connected, thereby defining the dataflow between the applications and hence the workflow structure. Each service in Taverna represents a Web Service, Java class, local script or other extended services. The Taverna system can be easily extended by means of several Application Programming Interfaces (API). To realize our workflow parameter

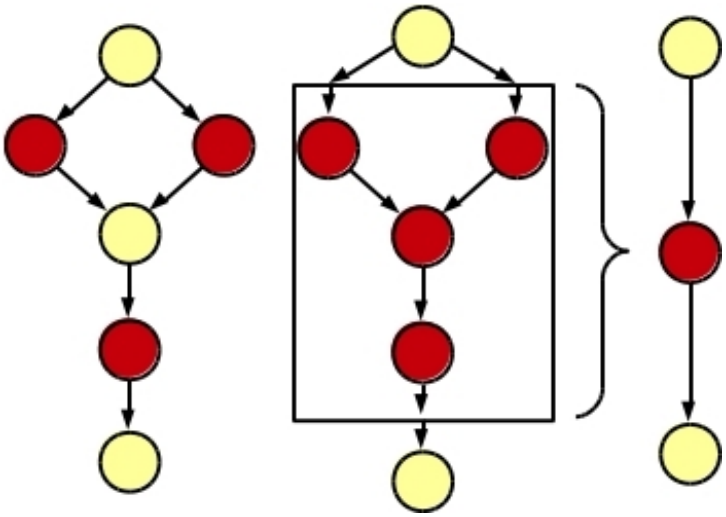

Figure 1. The left picture shows the original workflow. The activities marked in red should be optimized. The middle illustrates the created sub-workflow grouped into the original workflow. The right side shows the created global workflow, whereas the red activity represents the sub-workflow.

optimization use case in the Taverna Workbench we have developed a new plugin which enables the submission of a complete workflow to the Grid. Similar to the previously developed Taverna Grid Plugin[2] it acts as an extended activity. However, the new plugin acts on the Taverna sub-workflow level. A sub-workflow is defined as a set of several individual activities combined as a dependent dataflow. In the activity context this addresses the wrapping of an activity around a group of several activities to be represented as one activity in a greater workflow. The new plugin takes advantage of this mechanism and defines the group of to be optimized activities as well as the in between lying activities as a new sub-workflow in the original workflow as shown in Figure 1. During execution of the original workflow, the plugin invokes the execution of the sub-workflow as a Grid job submission. This job is send to the Grid including minimum two separate files, one containing a description of the input parameter and another including the workflow file. On the Grid the workflow is consumed by a Taverna server using the provided input files. To optimize a scientific workflow, several executions of the sub-workflow are required in each generation. To speed up the execution, the plugin makes use of the parallel execution mechanism in Taverna. Using this, several threads are started to execute several sub-workflows in parallel, each of which using a different parameter set. On the Grid, the execution can be distributed on several target systems to take advantage of the computing power available in e-science infrastructures.

4 Multi-level Parallel Architecture

The optimization use case is embedded in a three tier execution architecture, shown in Figure 2. The architecture can utilize even large Grid resources as it implements a hierarchy with up to three levels of parallel execution in its architecture. At the uppermost layer the Taverna Workbench is established on the client. In the Workbench users can set up the workflow and configure inputs and outputs as well as describe the optimization composition. As described before in section 2, parameter sets are created in each generation of the optimization process by the stochastic algorithm. The workflow is then instantiated with each of these individual parameter sets. Each individual workflow instance is then sent to the UNICORE service orchestrator. Because of the fact that all workflow executions are independent the execution can be done in parallel using Taverna's parallel execution mechanism. The UNICORE service orchestrator manages the distribution of the workflow jobs on the Grid infrastructure and monitors the status of the jobs. For distribution it implements various brokering strategies to find the best suited set of resources and enables load balancing. Using this feature, the workflows are submitted in a distributed way and one instance of Taverna is used, to execute a single workflow, as shown in the Figure 2 as second layer.

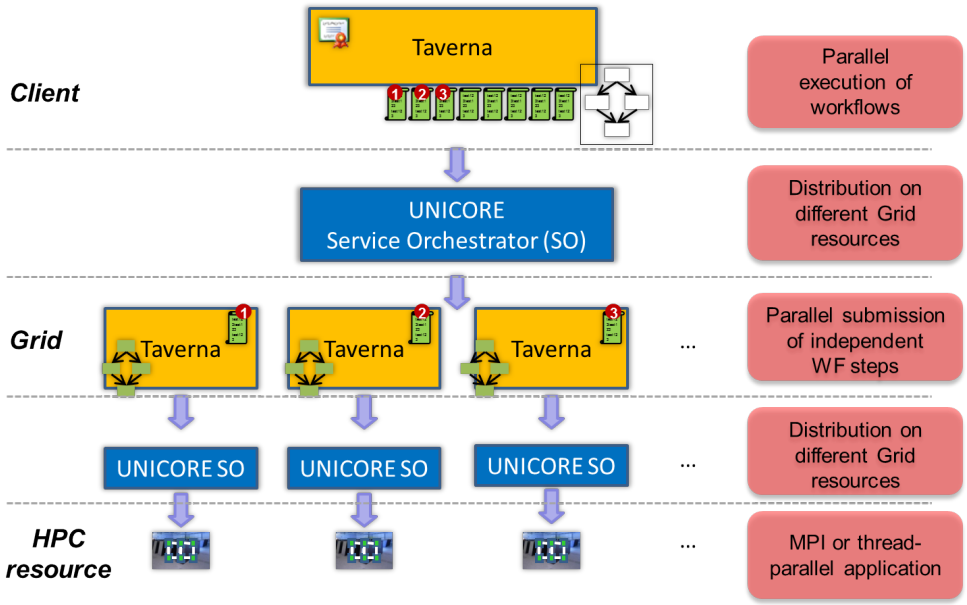

Figure 2. The three tier execution architecture. On the first tier the Taverna Workbench on the client, on the second tier the Taverna server, on the third tier thread parallel or MPI applications.

At the second layer, each workflow itself may contain several independent steps that due to the parallel execution model in the Taverna and the job distribution of the UNICORE Service Orchestrator can be represented as individual UNICORE jobs and thus be submit-

ted in parallel on the Grid for execution on high-performance-computing (HPC) resources.

Finally, at the third tier in Figure 2 some of the workflow steps may themselves be implemented as MPI- or thread-parallel applications, which constitutes the third level of parallel execution. Together these three layers of parallel execution enable high utilization of Grid resources and provide significant speedup of the optimization process.

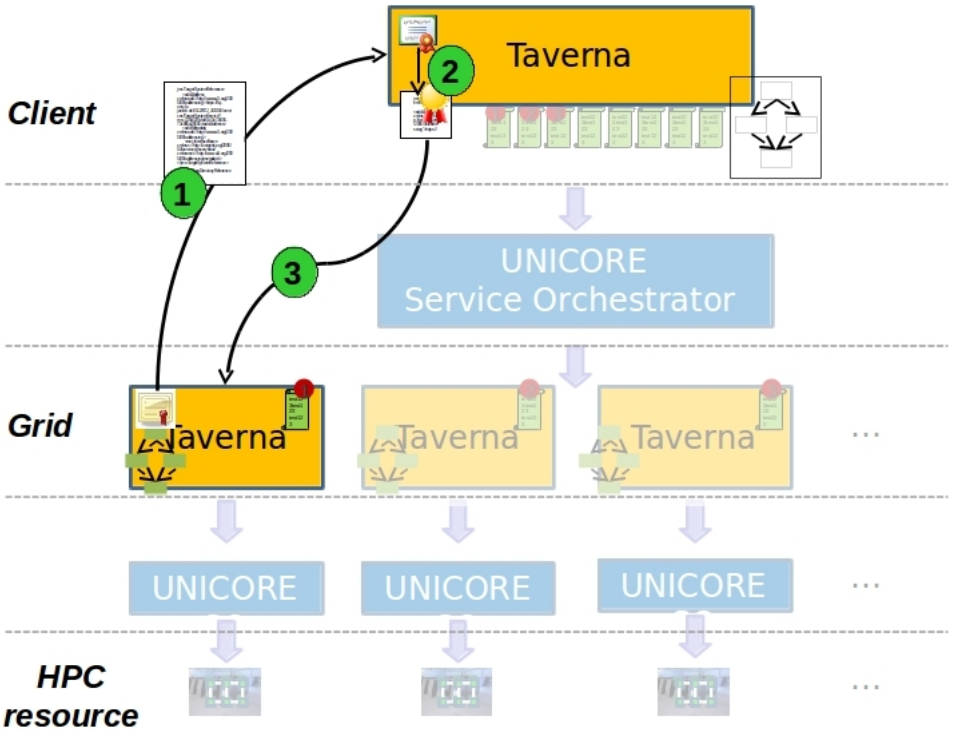

Figure 3. The security propagation mechanism. In step 1 the DN is published via the IDB to the Taverna Workbench. Taverna produces a trust delegation in step 2. In step 3 the trust delegation is forwarded to the Taverna server, which uses the trust delegation to assign a SAML assertion for job submission.

5 Security Issues

To enable a secure distribution of scientific workflows on the Grid, we have developed an architecture implementing a novel security propagation mechanism. The mechanism utilizes SAML assertions in combination with the UNICORE incarnation database. As the Taverna Workbench uses the Credential Manager to access the users X.509 certificate, this basic security feature had to be extended on the client and server side for the secure execution of workflows on the Grid.

The security issue arises at the Taverna server located at the second layer in the architecture (Figure 2). Here the execution of each workflow takes place. As described before,

the workflows themselves may contain several Grid activities, which require to be submitted to the Grid infrastructure[9]. For this procedure the certificate of the user is required to conform the high security standards existing in Grid infrastructures. Usually, the execution of the workflow takes place on the client machine, where the certificate is available. Thus, the user is allowed to send signed jobs to the Grid infrastructure for execution. In the present case, the Taverna server has to submit these jobs to the Grid infrastructure. Unfortunately, the server is not granted with access to the target systems nor the user certificate is available at the server. This is based on the fact, that the private user certificate should never be stored on an external storage and has to be kept in privacy. Furthermore, it is not secure to send the private user certificate over network to the server. Consequently, the user has to delegate his/her access rights to the server in any other way. To equip the server with these delegation rights, we investigated a new mechanism based on SAML assertion in the Taverna Workbench as well as in the Taverna server. The general mechanism is based on trust delegation using the user certificate and the certificate of the Taverna server. In detail, the server certificates distinguished name (DN) is stored in the incarnation database (IDB) of the UNICORE server. In the IDB, each available application and depended meta data are stored for the specific target system. Within the meta data of the Taverna server application, the DN of the server is provided. During the request of the UNICORE Service Orchestrator, which collects target site specific information such as available applications, it also fetches the DN of the server, as shown in Figure 3. This information is provided to the Taverna Workbench, which extracts and stores required information, such as the DN, or application specifications on the client machine. This stored DN and the user's certificate is taken to create a trust delegation, which delegates the user's credentials to the server. It is sent as a file together with the workflow job to the Taverna server. The server can use this trust delegation to create a SAML assertion. With the SAML assertion the server can submit the individual application job. This mechanism allows the user to delegate all rights for execution of the individual thread-parallel or MPI activities to the Taverna server.

6 Results

To demonstrate the successful establishment of architecture and security, we set up two workflows representing two different use cases from the area of data mining. The first workflow contains three activities, of which we want to optimize two. As a result, a sub-workflow is automatically created containing two activities, one UNICORE Grid activity and one local script. The second workflow consists of ten activities, of which we want to optimize nine. We obtain a sub-workflow consisting of eight UNICORE activities and one local script. For comparison, the sub-workflows were first executed in parallel on a Grid resource via the developed workflow Grid plugin as shown on the left side of Figure 4. Secondly the sub-workflows were executed in parallel on the local client machine, as shown on the right side of Figure 4. Comparing the four statistics we can say, that the parallel execution of the small workflow as Grid Jobs and the parallel execution on the client machine produce a similar workload on the client. However, comparing the parallel execution of the large workflow as Grid Jobs with the parallel execution on the client machine shows that our developed plugin reduces the workload on the client machine. Following, we can say that the scaling of the new Grid workflow plugin in Taverna is independent of the size of the workflow and parallel execution of activities. It continually produces a constant workload

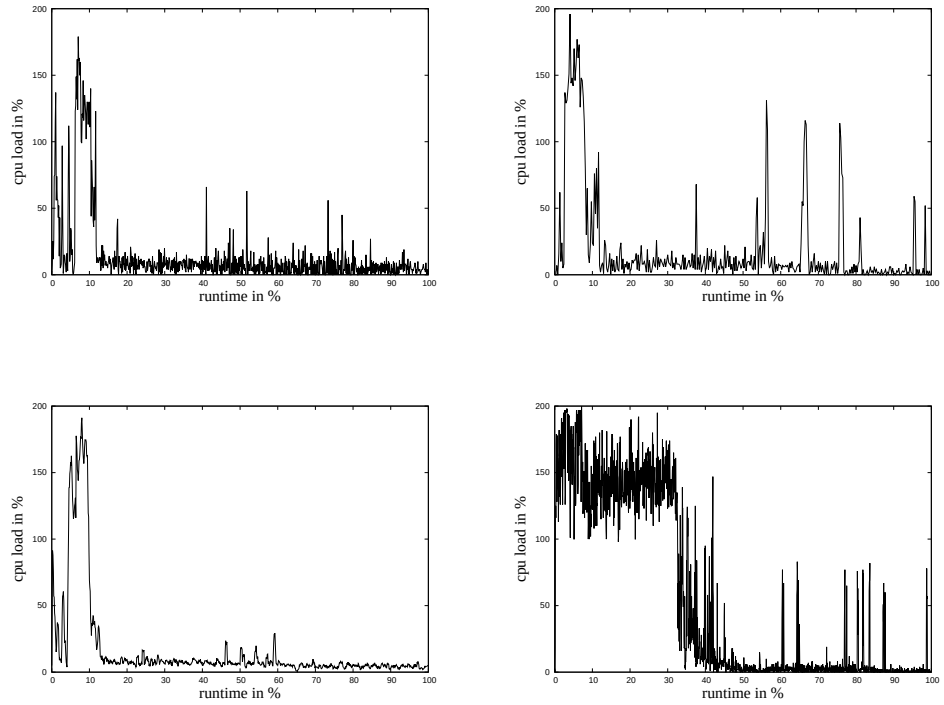

Figure 4. Profiles of CPU load produced by the Taverna client **Upper left**: small workflow, using Taverna server **Upper right**: small workflow, without offloading to Taverna server **Lower left**: large workflow, using Taverna server **Lower right**: large workflow, without offloading to Taverna server

at the client which can already be reduced to the workload of only one single job execution when adapted to the currently developed parameter sweep extension in UNICORE.

7 Conclusion

In this work, we have introduced a new security propagation mechanism to enable a secure distribution of workflow executions to a UNICORE based Grid infrastructure. The mechanism is based on trust delegation and enables the client to read the server DN from the IDB to create a trust delegation. This in turn can be taken by the server to submit jobs on behalf of the user making use of SAML assertions. The mechanism was integrated in a three tier architecture, enabling the execution of multiple Taverna workflows on a Grid resource. The distributed execution of workflows reduces the client workload during parallel execution of several workflows on the client. We offer in our paper a positive result, as the parallel workflow execution on the Grid does not influence the client workload. Additionally, the scaling is independent of the workflow size and parallel activities inside the workflow.

As future work, we want to utilize a new UNICORE feature called parameter sweeps, which is under development. It allows to only send one job to the UNICORE server containing sweeping information. With these information the server can create a set of jobs, each with an individual parameter set. Regarding our optimization use case, we only instantiate one job containing the parameter set ups for one generation. On the server the individual workflow instances with the different parameters are then created. The adaption would again significantly affect the workload on the client, as we only have to send one Grid job containing the different parameter sets.

Acknowledgements

We would like to thank the Taverna developer team for their work and great support.

References

1. I. Foster, *Globus toolkit version 4: Software for service-oriented systems*, IFIP International Conference on Network and Parallel Computing, number 3779, pages 2–13. Springer-Verlag, 2005.
2. S. Holl, O. Zimmermann, and M. Hofmann-Apitius, *A UNICORE Plugin for HPC-Enabled Scientific Workflows in Taverna 2.2*, IEEE World Congress on Services, SERVICES 2011, pages 220–223, 2011.
3. H. N. Krabbenhoeft, S. Moeller, and D. Bayer, *Integrating ARC grid middleware with Taverna workflows*, Bioinformatics, 24, pages 1221–1222, May 2008.
4. 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, W. Ziegler, *Unicore 6 - recent and future advancements*, Annals of Telecommunications, 65(11-12), pages 757–762, 2010.
5. P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble, *Taverna, reloaded*, SSDBM 2010, Heidelberg, Germany, 2010.
6. Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers, *Examining the Challenges of Scientific Workflows*, IEEE Computer, 40, (12), pages 26–34, 2007.
7. E. Deelman, D. Gannon, M. Shields, and I. Taylor, *Workflows and e-Science: An overview of workflow system features and capabilities*, Future Generation Computer Systems, vol. 25, no. 5, pages 528–540, 2009.
8. I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, *Workflows for e-Science: Scientific Workflows for Grids*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
9. K. Benedyczak, P. Bała, S. van den Berghe, R. Menday, and B. Schuller, *Key aspects of the UNICORE 6 security model*, Future Generation Comp. Syst., vol. 27, no. 2, pages 195–201, 2011.

A Data Driven Science Gateway for Computational Workflows

Richard Grunzke[1], Georg Birkenheuer[2], Dirk Blunk[3], Sebastian Breuers[3], André Brinkmann[4], Sandra Gesing[5], Sonja Herres-Pawlis[6], Oliver Kohlbacher[5], Jens Krüger[5], Martin Kruse[3], Ralph Müller-Pfefferkorn[1], Patrick Schäfer[7], Bernd Schuller[8], Thomas Steinke[7], Andreas Zink[5]

[1] Technische Universität Dresden, Germany

[2] Universität Paderborn, Germany

[3] Universität zu Köln, Germany

[4] Johannes Gutenberg-Universität Mainz, Germany

[5] University of Tübingen, Germany

[6] Ludwig-Maximilians-Universität München, Germany

[7] Konrad-Zuse-Institut für Informationstechnik Berlin, Germany

[8] Forschungszentrum Jülich, Germany

The MoSGrid Science Gateway provides convenient ways to analyse and model three-dimensional molecular structures using computational chemistry workflows. Molecular dynamics, quantum chemistry, and docking are the considered application domains. The science gateway including the data repository and underlying infrastructures was developed on top of the Liferay-based version of WS-PGRADE (Web services Parallel Grid Runtime and Developer Environment) using gUSE (grid User Support Environment), UNICORE, and XtreemFS technologies. The gateway allows users to seamlessly interact with the chemistry applications using data from the MoSGrid repository and running on Distributed Computing Infrastructures (DCIs). Data in the repository is centred on the Molecular Simulation Markup Language (MSML), which was developed in MoSGrid. For applications on the one hand and for analysis and visualization on the other hand format conversions from and to MSML are performed. Metadata, extracted from the MSML, is used to search the repository. The MoSGrid Science Gateway offers the use of data-centric workflows by integrating and extending a multitude of technologies. Users are enabled to run molecular computational simulations easily and efficiently with consistent data representations.

1 Introduction

The Molecular Simulation Grid (MoSGrid)[8] is a BMBF funded project of the German D-Grid initiative aiming on easing the access and use of molecular simulations in computational chemistry in a grid environment. Computational chemistry is an established discipline in natural sciences; it targets on modelling and analysing three-dimensional molecular structures. Important application domains are molecular dynamics, quantum chemistry, and docking. Each of these domains consists of a diverse set of scientific simulation programs and data flows. The data flows of the chemical simulations consist of many possible steps, including file transfers, data conversions, and molecular analyses. Hereby, the available state of the art simulation codes, hand in hand with today's high

performance computing infrastructures, allow molecular simulations to solve increasingly complex scientific questions. Therefore, more and more scientists are using these tools.

However, even today's most powerful simulation instruments still have limitations, especially due to the design of the user interfaces. Many sophisticated tools are command line driven and not supported by a graphical user interface. As a consequence, the new users have to become familiar, not only with the large number of methods and chemical theories, but also with the use of the codes and the handling of the data flows. To lower the hurdle of using these programs, intuitive and data centred user interfaces are paramount.

MoSGrid offers a science gateway that allows easy access to complex molecular simulations. The included web-based graphical user interface allows the simulation code independent setup of simulation workflows that are submitted through the UNICORE grid middleware[23] to the underlying clusters. Every user can apply commonly used metadata enriched workflows that are available in a recipe repository. The metadata description allows efficient searching for workflows by a description of the underlying dataflow. This lowers the hurdle for applying computational chemistry methods even for novice users.

In their every-day work, experienced users can use, develop, improve, and publish increasingly complex workflows. As a result of these efforts, the variety of mapped data flows increases steadily, allowing less experienced users to deal with increasingly complex scientific questions.

In the remainder of this paper the data driven molecular simulations are described in more detail. Section two gives a brief summary of the technological background. Section three depicts the structure of MoSGrid's software stack. Section four outlines the supported application domains and section five the metadata enriched data repository, including descriptions of the metadata, the data management, the adapters and parsers for the data conversions, the connection to the distributed file system, and the UNICORE connection. The paper is completed by a short summary and outlook.

2 Background

In the following the technical background including Liferay[13], WS-PGRADE[5], XtreemFS[12], and UNICORE is described.

2.1 Liferay

Users only need a computer with a connection to the Internet and a web browser to gain access to a portal. The open-source portal framework Liferay is widely used in grid and cloud projects and supports the standard JSR168[1] and its successor JSR 286[20]. It provides an adaptable portal interface, configurable database connections, and default applications inside a portlet container. As soon as database connections are centrally configured in the portal framework, so-called portlets can access relational databases via served libraries. Portal frameworks are web-based applications necessitating an application server. The MoSGrid project applies Liferay deployed inside the application server Apache Tomcat[2]. The latter controls the access of users and programs to resources and ensures the integrity of data during transfers via HTTP or HTTPS. Moreover, Apache Tomcat offers libraries for uploading and downloading data, which are facilitated by Liferay. Thus, features for transferring data from a user's computer to the application server and vice versa can be easily implemented in a portlet.

2.2 WS-PGRADE and gUSE

gUSE[5] consists of a set of web services providing a collaborative application development environment for DCI services. These services include a workflow engine, a workflow storage, an application repository, an information system, and a monitoring system. Several types of DCIs are supported for the submission of workflows like grid and cloud infrastructures, batch systems, and desktop grids (e.g., UNICORE, gLite[18], BOINC[6]). Additionally to the management of local data, gUSE is able to manage input data and output data via distributed file systems like LFC (Logical File Catalogue)[18] in gLite jobs. In the MoSGrid project, gUSE has been extended by the capability to utilize XtreemFS in UNICORE jobs.

WS-PGRADE is a highly flexible graphical user interface of gUSE, which applies in its current version Liferay. It enables the users to manage the whole lifecycle of workflows and share workflows and data via repositories.

2.3 UNICORE

UNICORE is a standards-based grid middleware providing secure access to federated compute and data resources. This includes cluster, grid, and cloud resources as well as access to desktop grids via the 3G-bridge interface[15]. UNICORE has been described in detail[23]. Apart from its traditional features like job submission and management, the MoSGrid science gateway relies on the new data management features for remote files that were introduced in recent releases.

Specifically, UNICORE has been extended with powerful metadata capabilities. Users can annotate files either manually or by using the automated file crawling and metadata extraction features. Metadata are indexed using the well-known Apache Lucene search engine[3], which offers powerful query functionality. To perform CRUD (create, read, update, delete) operations on metadata, execute queries and trigger the integrated automated metadata extractor, the UNICORE command line client can be used.

In the UNICORE metadata solution, the metadata is stored on the same storage as the data using hidden files. This simplifies consistency checks considerably. The metadata is schema-free, using key-value pairs stored in JSON[14] format.

2.4 XtreemFS

The distributed file system XtreemFS is applied in MoSGrid for storing data related to chemical simulations.

XtreemFS is an object-based file system for grid and cloud computing environments. The server and clients can be distributed worldwide. Replication, which ensures availability in case of failures, allows for storing data among many servers. XtreemFS offers scalability by adding further servers to the system.

Data is managed by object storage devices (OSDs), which offer more capabilities and a more sophisticated interface than network-attached block storage devices in traditional parallel file systems. Metadata and replica catalogues (MRCs) store all metadata of the file system, such as filenames and the directory tree as well as file sizes, timestamps and ownership information. The OSDs store the content of files as objects, where each object refers to a chunk of data of a given size.

The servers are accessed through a FUSE (Filesystem in Userspace) client or a client using the Java API, which offer a POSIX file system interface. XtreemFS supports both X.509 certificates and SAML assertions for authentication and POSIX access rights and ACLs for authorization. Both, the JAVA API and support for SAML assertions were implemented within MoSGrid. Currently, MoSGrid is operating 16 OSDs with a capacity of 96 TiByte and one MRC service.

3 MoSGrid Software Structure

This section describes the general structure of the MoSGrid Science Gateway. It is based on the technologies Liferay, WS-PGRADE, gUSE, XreemFS, and UNICORE. They are described in the previous section.

The workflow-enabled grid portal WS-PGRADE facilitates Liferay and its features. It provides a flexible user interface for the underlying gUSE and allows creating, modifying, submitting, and monitoring workflows. gUSE provides the high-level middleware services including a workflow system and is able to submit jobs to UNICORE and other DCIs.

A repository is created to store and make data accessible. It is based on UNICORE, which contributes metadata capabilities, and XtreemFS, which provides distributed file storage capabilities. UNICORE was extended by several new software components. An URL scheme was implemented to directly access XtreemFS via UNICORE (see section 5.5.1). In order to index the scientific simulation data, a metadata extractor was written, which filters metadata and stores them into JSON (see section 5.5.3). The term metadata refers to data describing other data for the purpose of being efficiently searchable.

In order to ensure a well-structured and universal representation of the data in MoSGrid the Molecular Simulation Markup Language (MSML) is used, which is based on the Chemical Markup Language (CML)[19] and was developed in MoSGrid.

A submitter for gUSE was implemented to index this metadata and to submit jobs to UNICORE. To search and access stored data in the repository, a search GUI is being implemented. Furthermore a Java API and a file browsing portlet were developed in order to access XtreemFS via the Science Gateway.

Data in MoSGrid is stored in the repository and passes through different phases and layers of the Science Gateway and the underlying infrastructure. A detailed description can be found in chapter 5.

4 Application Domains

The field of computational chemistry evolved three major branches of computational techniques. These application domains are molecular dynamics (MD), quantum chemistry (QC), and docking. They cover most use cases arising in computational chemistry and related fields. Each of these domains consists of a diverse set of scientific simulation programs and specific workflows.

4.1 Molecular Dynamics

The simulation of molecular motions and dynamics is the main focus of this application domain[17]. In order to achieve reasonable time scales, atoms are treated as point masses,

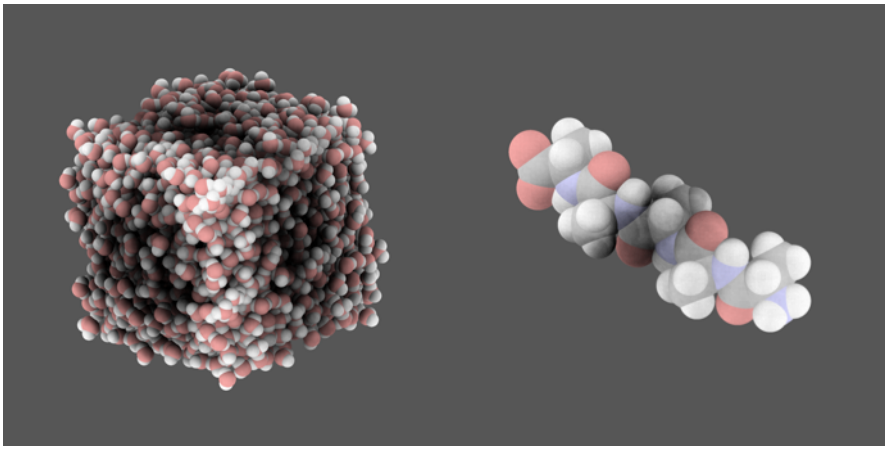

Figure 1. On the left side a typical water filled simulation box is shown, surrounding a protein (right side).

following a classic mechanics approach. Chemical bonds are represented by springs, connecting the simplified atoms. This representation of simulation systems is described energetically by force fields, an empiric collection of mathematical functions for all inter- and intramolecular interactions. The integration of Newton's equation of motion allows the analysis of forces upon atoms and consequently the prediction of atomic positions. A visualisation of a structure can be seen in figure 1.

Regarding workflows two general approaches are of interest for users from the MoSGrid community:
1. Expert users are mainly interested in using the available compute resources. Therefore a simple workflow is integrated into the corresponding MD portlet, allowing the submission of existing job descriptions. For Gromacs this is the so-called topol.tpr file[10].
2. Novice users need guidance in applying simulation protocols frequently used in the field. A prerequisite for scientifically meaningful protein based MD simulations is the solvation, minimization, and equilibration of the molecule of interest. The workflow for this task follows the dimensions and time scales typically used in the field[16]. Additionally a number of analysis and visualization features are integrated into the MD portlet, allowing the display of energy curves and visualization of molecules, beside an overview over numerical and textual information.

4.2 Quantum Chemistry

In the Quantum Chemistry domain, Gaussian09[7] and Turbomole[24] are integrated into the MoSGrid portal. For a quantum chemical calculation, one needs geometry data, information on calculation type (optimization, frequency calculation etc.), a method information (e.g. Density Functional Theory (DFT) with functional, Coupled Cluster (CC), or second rank Møller-Plesset perturbation theory (MP2)), basis set, charge, and spin information. Hence, the amount of input data is rather small. A zinc complex for example is shown in figure 2.

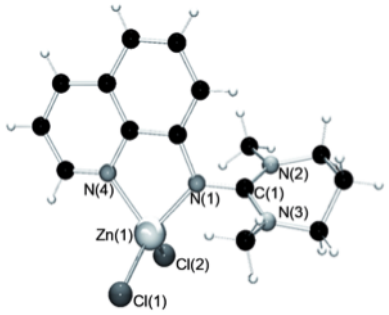

Figure 2. A zinc guanidine complex in ball and stick representation[9].

The general difference between Gaussian09 and Turbomole lies in the different input types: Gaussian09 requires one single input file (.gjf), which contains all information listed above. Turbomole needs a folder with a defined file content. The files contain the coordinates, basis sets, auxiliary basis sets, molecular orbitals of an initially guessed electronic distribution (calculated in the GUI of the portal). The so-called control file assembles all connections together for processing of the calculation. Within the QM portlet, the user provides all this information code-independently. After submission a pre-processing step in the portal translates all data into MSML and generates the corresponding generic input file(s).

Like the input format the output data format is also code-dependent: Gaussian09 produces a textual log-file (.log) with all relevant information. A binary checkpoint file (.chk) is needed for orbital visualization and restarting of calculations. Turbomole produces a large number of log-files (e.g. one for each processor node), but the essential information is collected in a log-file. Furthermore, there is detailed information on the optimization convergence, the gradients, the energy, and dscf statistics that are all given in small separated ASCII files.

In MoSGrid, the data is parsed for user-desired information (e.g. convergence, frequencies), for a faster data analysis and visualization, and translated to MSML for further storage in the data repository. The use of workflows helps to facilitate processes because in quantum chemistry, different types of calculations are performed in sequences: a molecule has to be optimized, then a frequency analysis, and further calculations for electronic spectra have to be performed[9].

4.3 Docking

Docking describes a general search method for identifying the binding pose and energy of ligands to a given receptor (see Figure 3). Originating from the field of computer aided drug design the typical use case focuses on finding a strong binding medical drug for a therapeutically interesting protein target. Typically based on aforementioned force fields different application employ different search algorithms in order to identify optimal ligand poses. For example, FlexX[21] decomposes the ligand into fragments, rebuilding it inside the binding pocket of the receptor. For the evaluation of binding energies so-called scoring

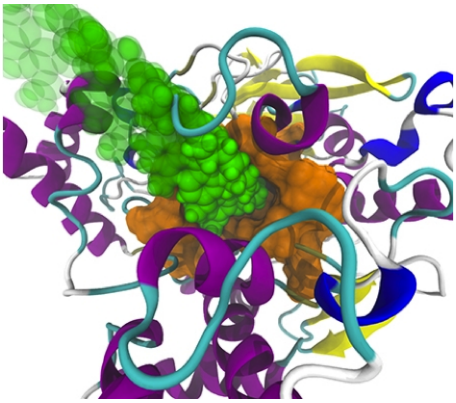

Figure 3. Schematical illustration of a ligand (green) approaching a proteins binding pocket (orange).

functions are employed, estimating the free energy of binding using multiple semiempiric approaches. The CADDSuite[4] for example not only contains the usual bonded and non-bonded force field energy terms, but also an estimate for the rotational entropy contribution.

The currently available docking portlet covers all preparation, placement and (re)scoring steps expectable for a scientifically meaningful docking study. First the receptor gets prepared, fixing hydrogens and removing undesired compounds. Furthermore co-crystalized water molecules are identified, potentially relevant for ligand binding. In parallel the ligand database, ranging from a single molecule to up to a couple of thousands, is split into appropriate parts. This step balances the desired runtime to available resources. Afterwards the ligands are docked to the receptor and energies are estimated. Finally the top scored ligands are displayed to the user including all numerical information.

5 MoSGrid Repository

The MoSGrid repository stores and handles all data and metadata related to MoSGrid. It consists of XtreemFS for raw data storage and UNICORE for data handling and metadata capabilities. To support the handling of data from different molecular simulation domains, MSML is introduced to provide a common data representation. The data management is described in terms of data flows and format conversions with parsers and adapters. Finally, the XtreemFS and UNICORE integrations are depicted.

5.1 Metadata with MSML

In each of the three supported domains in MoSGrid a variety of programs were developed to cover special computational aspects.

However, all of them supply common core functionality, e.g. in QC solving the Schrödinger equation or in MD integrating the equation of motion, or in all of them calculating energies. This common base enables the implementation of a common description within and beyond each domain. Besides, every domain has unique and distinct needs and properties regarding a description.

In MoSGrid a description language was needed to address these properties but also to enable the definition of abstract and generic simulation descriptions. A common data format has been missing to store the simulation results transferably and comparably within every domain. Therefore, the Molecular Simulation Mark-up Language (MSML) has been developed. This language is the basis for storing the simulation metadata, i.e. the simulation setup description and the results of a performed simulation. MSML is based on the Chemical Mark-up Language (CML) and uses a subset of it as well as extends it by necessary, new features, e.g. enumerations. The most important features for the development in MoSGrid are the concepts of dictionaries and conventions published with the release of CML 4. The dictionaries, plain CML documents, contain the controlled vocabulary, meaning the terms that are allowed to be used in a certain domain, e.g. MinimizerAlgorithm or ElectrostaticsAlgorithm.

Besides, the conventions establish a meta structure onto the XML document by defining relations and constraints between the entries of a dictionary. Thus, CML 4 provides a scaffold that can be filled with specific syntax and semantics.

The common scaffold (see Listing 1) used for every domain contains the MSML header, a part with computational system requirements, a list of parameters for characterizing the simulation, and a property part storing the simulation results. The <Environment/>tag contains the memory requirements, the amount of processors, information that define the computational environment. In the <Initialization/>tag the starting structure and the simulation parameters are defined. Eventually, the <Finalization/>tag holds the results information, e.g. a minimized structure and/or simulation related properties like the energy or a docking score.

Listing 1. Common MSML Scaffold

```
<CML-Header>
<JobList>
<Job A>
<Environment>
        # nodes
        # cores
        amount of memory
        ...
</Environment>
<Initialization>
        molecular structure
        simulation parameters
</Initialization>
<Finalization>
        resulting structure
        various properties
</Finalization>
</Job A>
<Job B/>
</JobList>
```

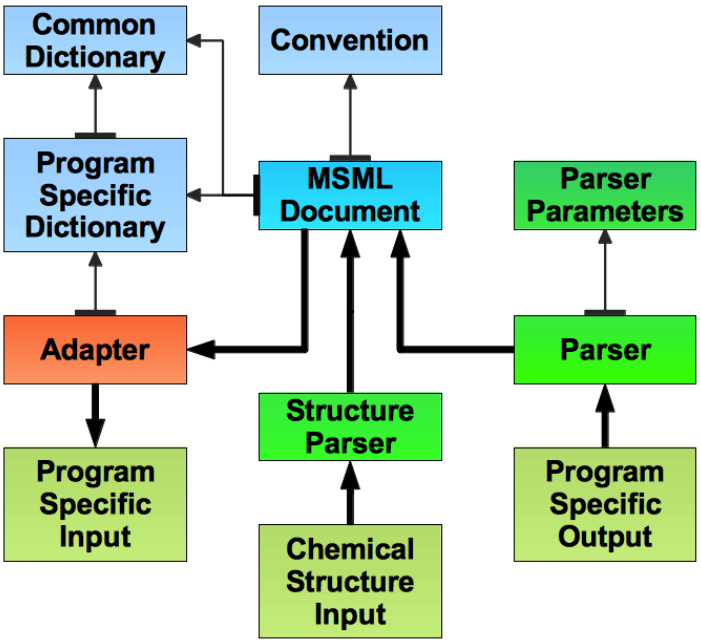

Figure 4. Translation mechanisms used in MSML to convert specific in- and output to and from MSML.

An abstract description of simulations in MD and QC were derived from workflows and requirements specified by the MoSGrid community. The descriptions have been implemented in common dictionaries, which represent the abstraction layer between programs and the concepts in every domain. Therefore, a program specific dictionary has to be provided containing references to the common dictionary of the domain. The common and specific CML documents can be used in unity to achieve the translation from the abstract MSML to the program specific input via an adapter program (see figure 4).

Additionally to the program specific input dictionaries output dictionaries have been developed to perform the abstract storage of simulation results. These are oriented along the special user requirements in each domain. These terms are filled with parser programs that extract the relevant result data from the program specific output. They convert it into to the respective SI unit, the standard metric system applied in sciences, and store it in the MSML document holding the initial simulation description.

5.2 Data Management

MoSGrid's data management consists of data and metadata storage, many possible transfers, conversions, and analysis steps.

Besides using existing data from the repository, users are able to upload new data to the repository via a portlet. For the automatic conversion and insertion in the repository, several steps have to be completed. The chemical structure parser performs the conversion of the data to a CML representation, which is aggregated with the MSML document (see

also 5.3.1. Structure Parser). Afterwards, the metadata extractor is executed to acquire relevant information from the MSML file and write it into a JSON representation. The meta information in the JSON file is read by the UNICORE metadata service and stored by the underlying Apache Lucence. By finally indexing the meta information, it is available for search and information retrieval.

To utilize the data stored in the repository, a user chooses a portlet according to his research domain of interest. He selects one of the available pre-configured workflows in the specific domain portlet, which provide sound default parameters for the applications chained in workflows. These can also be freely adapted to suit the user's needs. Then stored MSML data is selected as input for the workflow.

At the beginning of a workflow, an adapter is used to transform the MSML document to the specific input format of the targeted chemical simulation code. This generated data, including the simulation parameters and along with the program specific structure format, serves as the input for the computational workflow. During the simulation steps inside the workflow a MSML representation of the results is created and continuously extended with further results as more of the jobs of the workflow are executed.

By extracting and indexing this resulting metadata at the end of a workflow, the data can subsequently be found and retrieved by the user via a graphical search interface as described in section 5.5.3.

5.3 Parser and Adapter

MSML represents the central data format, which allows for interoperability between different jobs and workflows independent of the used applications. Thus, different tools for the conversions from and to MSML are needed. These tools can be distinguished into three main types; structure parsers convert input structure formats like PDB and SDF/Mol to MSML, parsers in general convert mainly unstructured application output to MSML, and adapters convert MSML to application specific input formats.

5.3.1 Structure Parser

In MoSGrid, we developed a tool, a structure parser, for the conversion between chemical structure formats such as PDB, SDF/MOL, and MSML. PDB support was developed based on BioJava[11], whereas SDF/MOL support was developed using the Chemistry Development Kit (CDK)[22]. The structure parser uses MSML as the internal data structure. For every supported structure format there is a reader and a writer tool. The reader parses the content of a file into the internal MSML structure. The writer converts the MSML description to the targeted format. Using MSML as the central hub eases parsing between different formats, as only one pair of reader and writer for every new format has to be developed. As part of a typical workflow, a user provides a chemical structure as input in form of a PDB or SDF/MOL file. This file is validated by the structure parser and converted to CML. The CML description of the structures is stored together with the MSML document.

5.3.2 Parser

Every simulation code supported by MoSGrid produces its very own, mainly unstructured output. Internally, MSML is used to represent and convert data. Thus, there is a need to

transfer unstructured output to MSML. A parser has been implemented that uses regular expressions to extract the necessary data and to store it in an object that can be serialized to MSML. Users are able to configure the parser via regular expressions, considering the tools' output. To aid them with the creation process a GUI has been implemented. The parser needs to be installed on grid resources as it is invoked within a workflow.

5.3.3 Adapter

Every domain provides several pre-configured simulation workflows. Some parameters have to be given manually to support flexibility of the workflows fitting to the users' needs. During the submission process of a workflow, an input mask allows the provision of mandatory and optional parameters. The input mask is tightly coupled to an MSML template, which stores all entered simulation parameters. The input formats vary between simulations codes used by a workflow. Thus, there are specialized adapters, which are able to transform the input parameter section of a MSML file into any input format. This may be a simple string of command line arguments but also more complex file formats are possible. During the submission of a workflow, the adapters' output is forwarded to the corresponding application of the workflow.

5.4 XtreemFS Integration

In this section it is described how XtreemFS is integrated to serve as the basis of the file storage and therefore the MoSGrid repository. XtreemFS aides MoSGrid in two key points: The first one is the support of security based on X.509 certificate infrastructures using UNICORE XUUDB mapping information and SAML assertions. XtreemFS is integrated into several components of the MoSGrid architecture: The first component allows for accessing XtreemFS by use of a file browsing portlet, which provides standard file system operations within a web browser like reading, writing or moving files, changing access rights and ACLs. The portlet was developed based on Vaadin and the XtreemFS JAVA API. When the portlet is initialized, a X.509 service certificate is used to authenticate access to the XtreemFS servers. For accessing a file, the JAVA API verifies the user's SAML assertion and the user's DN is used to authorize against the XtreemFS MRC server. The second component includes XtreemFS references in WS-PGRADE gUSE workflows. This allows for input files to be uploaded into the job's working directory and output files to be downloaded back into XtreemFS. This unloads the WS-PGRADE portal from handling file transfers itself.

To allow for these references to be included into UNICORE jobs, several changes to WS-PGRADE had to be realized. First, WS-PGRADE was extended to support the XtreemFS URL schema used in UNICORE. This required the modification of the UNICORE submitter to allow the conversion of XtreemFS URLs to UNICORE staging definitions. Second, WS-PGRADE was extended to support the use of these XtreemFS URL definitions from within application portlets. URLs can be set for a gUSE workflow and are passed to the UNICORE submitter for inclusion into UNICORE jobs. The last component is the integration of XtreemFS into UNICORE. This enables files in an XtreemFS volume to be accessible from within UNICORE. A URL schema for easily referencing files in an XtreemFS volume from within UNICORE was developed (see 5.5.1).

To summarize, data is stored in and transferred to XtreemFS by the user using a portlet and a web browser. The files stored in XtreemFS are referenced in WS-PGRADE gUSE workflows. These are translated to UNICORE jobs and UNICORE staging information by the use of the UNICORE submitter. Finally, UNICORE handles file staging from and to XtreemFS.

5.5 UNICORE Integration

This section describes how UNICORE is integrated. Via the XtreemFS URL scheme, files in XtreemFS can be directly accessed from within UNICORE jobs. The gUSE submitter is able to interact with UNICORE to manage jobs and make metadata available for searching. A metadata extractor has been developed to extract metadata from MSML and features are offered to efficiently search data.

5.5.1 XtreemFS URL Scheme

To allow optimal integration of XtreemFS within UNICORE jobs, a simple extension of UNICORE was required. The goal was to achieve data staging from and to XtreemFS without additional overhead. There are two cases to consider: On the one hand, XtreemFS can be mounted locally at the job execution site. In this case, file staging should be done by simple copying. On the other hand, XtreemFS is not available locally, but only through a UNICORE storage service (SMS). In this case, a remote data movement operation is required. The UNICORE XtreemFS data staging support is implemented through the new URL scheme "xtreemfs://". Each execution site can configure whether XtreemFS is available in the local file system or at a remote SMS.

5.5.2 gUSE Submitter

gUSE provides a set of services for the management of workflows including the data-driven workflow engine and submitters. Jobs within the same workflow may be configured for diverse DCIs and the workflow engine invokes each with an appropriate submitter. In MoSGrid we developed a component called UNICORE submitter, which is used to translate gUSE workflow jobs to UNICORE job definitions. These job definitions include data staging information, requirements to the cluster and the application or script to run on the target grid cluster.

The submitter checks every available resource, whether it matches the requirements defined in a gUSE workflow job such as memory, CPUs, cores or walltime. As part of this check, the UNICORE submitter queries the UNICORE IDB of the connected grid clusters. The IDB contains meta information about the simulation codes, available on its cluster. This allows selecting the appropriate cluster for executing the recipes substep jobs, even if uncommon code is used. Once a matching UNICORE resource has been found, a job on the UNICORE resource is created based on the UNICORE job definition. The UNICORE submitter passes the XtreemFS references in the data staging part of the job description, so file transfer from and to XtreemFS is exclusively handled by the UNICORE middleware.

5.5.3 Metadata Integration

To extract information from the MSML representation of the data a tool, the metadata extractor, was created. It was written in Java and writes data to the JSON format. Data in this format can then be indexed by the UNICORE metadata service to make it available for searching.

Searching capabilities are planned to be made available via several options. One is using a separate repository portlet. It will allow browsing the data repository and finding specific data via a search field. The search field will allow searching in any metadata field and will show a view of the data, which corresponds to the result.

Another option is the inclusion of a repository view in the monitoring tab of the domain portlets. This will allow seeing a view of the repository, which shows data related to the currently running workflow.

Also an option in the domain portlets is planned, which allows using the repository portlet for choosing the input data for a calculation. It is envisioned to temporarily overlay the domain portlet with the repository portlet. The user then can perform a search, chooses input data, and the repository portlet returns references to the data.

6 Summary and Outlook

The paper presents the data handling aspects of the MoSGrid Science Gateway. The underlying technologies, the application domains, and the MoSGrid repository are described. It is shown that the data handling penetrates every layer of the MoSGrid infrastructure and how MoSGrid deals with this situation by having developed the MSML data representation, using XtreemFS for raw data storage and using UNICORE for metadata management.

In the future it is planned to further integrate the metadata capabilities. A WebGL visualization portlet is being developed to be able to show results in a representative manner ready to be included in scientific publications. Re-implementations and improvements of the domain portlets are underway using a portlet-API to support a unified look and feel and to lower the development complexity. To make the MoSGrid Science Gateway available to the general public, it is planned to fully release it as open-source software.

Acknowledgements

This work is supported by the German Ministry of Education and Research under project grant #01IG09006 (MoSGrid) and by the European Commission's 7th Framework Programme under grant agreement #RI-283481 (SCI-BUS), #261585 (SHIWA), and #RI-261556 (EDGI).

References

1. A. Abdelnur and S. Hepper. JSR 168: Portlet Specification. http://www.jcp.org/en/jsr/detail?id=168, Oct 2003.
2. The Apache Software Foundation: Apache Tomcat. http://tomcat.apache.org/tomcat-6.0-doc/.
3. Apache Lucene. http://lucene.apache.org.
4. CADDSuite (Computer-aided Drug Design Suite). http://www.ball-project.org/caddsuite.
5. Z. Farkas and P. Kacsuk. P-GRADE Portal: a generic workflow system to support user communities. *Future Generation Computer Systems*, 27(5):454–465, 2011.
6. Z. Farkas, P. Kacsuk, Z. Balaton, and G. Gombs. Interoperability of boinc and egee. In *Future Generation Computer Systems*, volume 26, pages 1092–1103, 2010.
7. M.J. Frisch and et al. Gaussian 09, Revision B.01, 2010. Gaussian, Inc., Wallingford CT.
8. S. Gesing, P. Kacsuk, M. Kozlovszky, G. Birkenheuer, D. Blunk, S. Breuers, A. Brinkmann, G. Fels, R. Grunzke, S. Herres-Pawlis, J. Krüger, L. Packschies, R. Müller-Pfefferkorn, P. Schäfer, T. Steinke, A. Szikszay Fabri, K. Warzecha, M. Wewior, and O. Kohlbacher. A Science Gateway for Molecular Simulations. In *EGI User Forum 2011, Book of Abstracts*, pages 94–95, 2011.
9. S. Herres-Pawlis, G. Birkenheuer, A. Brinkmann, S. Gesing, R. Grunzke, R. Jäkel, O. Kohlbacher, J. Krüger, and I. Dos Santos Vieira. Workflow-enhanced conformational analysis of guanidine zinc complexes via a science gateway. *Studies in Health Technology and Informatics*, pages 142–151, 2012.
10. B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *Journal of Chemical Theory and Computation*, 4(3):435–447, 2008.
11. R.C.G. Holland, T. Down, M. Pocock, A. Prlić, D. Huen, K. James, S. Foisy, A. Dräger, A. Yates, M. Heuer, and M.J. Schreiber. BioJava: an Open-Source Framework for Bioinformatics. *Bioinformatics 2008*, 24(18):2096–2097, 2008.
12. F. Hupfeld, T.i Cortes, B. Kolbeck, J. Stender, E. Focht, M. Hess, J. Malo, J. Marti, and E. Cesario. The XtreemFS Architecture - A Case for Object-based File Systems in Grids. *Concurrency and Computation: Practice and Experience*, 20(17):2049–2060, 2008.
13. Inc. Liferay: Liferay. http://www.liferay.com.
14. Java Script Object Notation (JSON). http://www.json.org.
15. M. Keller, J. Kovacs, and A. Brinkmann. Desktop Grids Opening up to UNICORE. In *UNICORE Summit 2011 Proceedings*, IAS Series, pages 67–76, 2011.
16. J. Krüger and G. Fels. Potential of Mean Force of Ion Permeation through alpha7 nAChR Ion Channel. In *Proceedings of IWPLS'09*, 2009.
17. J. Krüger and W. B. Fischer. Exploring the conformational space of vpu from hiv-1: a versatile adaptable protein. In *Journal of Computational Chemistry 28(14)*, pages 2416–2424, 2008.

18. E. Laure, C. Gr, S. Fisher, A. Frohner, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, R. Byrom, L. Cornwall, M. Craig, A. Di Meglio, A. Djaoui, F. Giacomini, J. Hahkala, F. Hemmer, S. Hicks, A. Edlund, A. Maraschini, R. Middleton, M. Sgaravatto, M. Steenbakkers, J. Walk, and A. Wilson. Programming the Grid with gLite. In *Computational Methods in Science and Technology*, volume 12, pages 33–45, 2006.
19. Peter Murray-Rust and Henry S. Rzepa. Cml: Evolution and design. *Journal of Cheminformatics*, 3(44), 2011.
20. M.S. Nicklous and S. Hepper. JSR 286: Portlet Specification 2.0. `http://www.jcp.org/en/jsr/detail?id=286`, June 2008.
21. M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. A Fast Flexible Docking Method Using an Incremental Construction Algorithm. *Journal of Molecular Biology*, 261:470–489, 1996.
22. C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann, and E.L. Willighagen. The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. *J. Chem. Inf. Comput. Sci.*, 43(2):493–500, 2003.
23. A. Streit, P. Bala, A. Beck-Ratzka, K. Benedyczak, S. Bergmann, R. Breu, J. M. Daivandy, B. Demuth, A. Eifer, A. Giesler, B. Hagemeier, V. Huber S. Holl, 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. *JUEL-4319*, February 2010.
24. TURBOMOLE v6.2 2010, A Development of University of Karlsruhe and Forschungszentrum Karlsruhe Gmbh, 1989-2007, TURBOMOLE Gmbh, since 2007. `http://www.turbomole.com`.

LEGO MINDSTORMS NXT Navigation with UNICORE

Sandra Bergmann and Matthias Richerzhagen

Jülich Supercomputing Centre,
Research Centre Jülich, 52428 Jülich, Germany
E-mail: {s.bergmann, m.richerzhagen}@fz-juelich.de

The easy application integration technique provided by UNICORE is demonstrated by controlling a LEGO MINDSTORMS NXT robot. This device consists of a micro controller brick, sensors, motors and multiple Lego building blocks. In order to control the robot and avoid performance problems, several components were developed to provide a stable access to the robot. To prove the concept, the use case represents a workflow consisting of two jobs: the first job computes the shortest path on a given map, whereas the second one navigates the robot along the path.
In the future, the scheduling methods of the UNICORE workflow system must be improved for situations where the Job execution takes place on the same local machine. Furthermore this use case can be used to demonstrate users how easy they can integrate new applications.

1 Introduction

UNICORE provides an easy integration technique for any type of applications. Most of the users run complex computations through UNICORE; e.g. the OpenMolGRID project[6] runs computations in molecular science and engineering or the UIMA-HPC project[7] runs applications in the context of data extraction methods.
Besides the computations of simulation, UNICORE can also be used to operate specific devices. In order to show how easy it is to control devices by a UNICORE application, the remote navigation of a LEGO MINDSTORMS NXT robot will be demonstrated. LEGO MINDSTORMS NXT is a programmable robotics kit released by Lego in late July 2006[2]. It consists of a brick, taking input from up to three motors, four sensors (sound sensor, light sensor, touch sensors and ultrasonic range sensor) and is integrated in multiple LEGO building blocks. The NXT brick is the heart of the robot containing a firmware developed by Lego.
The retail version of the LEGO MINDSTORMS NXT set contains software for writing programs which is based on National Instruments LabVIEW and provides a visual programming language[1]. In order to be more flexible in controlling the robot we choose a custom-firmware called leJOS NXJ which allows the execution of Java programs[3].
To ensure flexible combination of several robot control programs, two different tasks: `Travel` and `Rotate` were developed. These tasks denote the corresponding control programs on the NXT robot. The argument passed to the tasks defines the length in centimetres or in case of `Rotate` the number of degrees.
The UNICORE application is defined in the simpleidb file and starts the client program which connects to a server for executing the specified task.

The demonstration of sequential combinations of tasks in a UNICORE workflow will be presented in Section 2. The use case contains the shortest way calculation on a given map and the navigation of the robot along the computed way. Section 3 describes the interaction between the components which are necessary to control the LEGO MINDSTORMS NXT robot through UNICORE. The last section concludes the described methods and provides an overview about possible extensions.

2 Use Case

The use case represents a two-step-workflow: the first job computes the shortest path on a given map whereas the second one navigates the LEGO MINDSTORMS NXT robot along the path. The logical design of the workflow is presented in Figure 1. The first job contains sub steps which are used to compute the shortest path. Input arguments are a map, start and target coordinates and the start rotation of the robot. The Dijkstra path finding algorithm which is used to calculate way points is provided by the leJOS NXJ firmware. These way points, which the robot has to pass by, will be translated to tasks in the second sub step. Tasks are commands in the format: T20 or R90, which means to travel 20cm or to rotate 90 degrees. These tasks are used as input argument for the second job, which then executes the tasks on the robot.

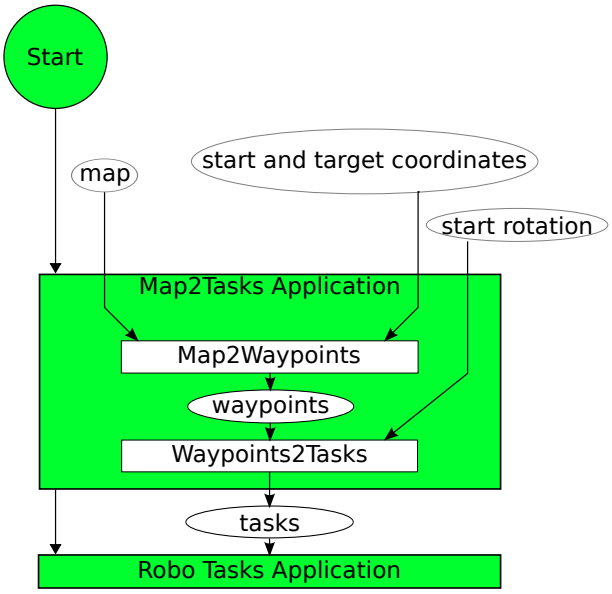

Figure 1. Use-case workflow

Tests showed several problems during the navigation of the LEGO MINDSTORMS NXT robot. External influences such as the ground the robot is moving on or the battery status result in an imprecise movement.

3 Technical Details

This section describes several components which have been developed to control the LEGO MINDSTORMS NXT robot. The architecture is shown in figure 2. The application defined in the simpleidb is a Java client program, called NXTJobClient, which connects to a server, the NXTJobServer. Every task will be sent through the client connection and then executed via a chosen worker class. Details about the different workers are given in Section 3.2.2

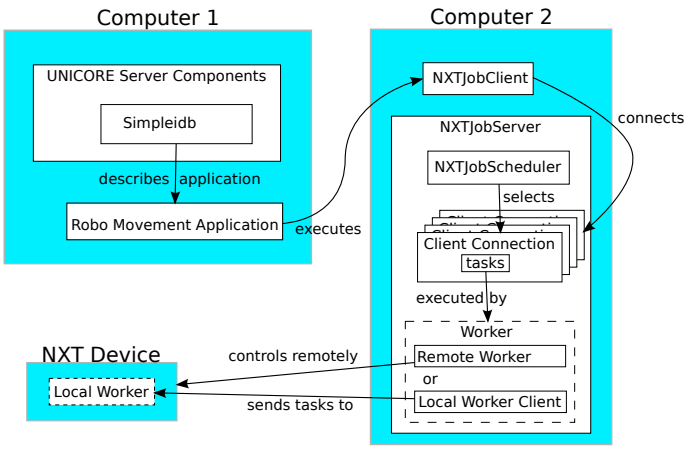

Figure 2. Architecture: Interaction between involved components.

3.1 Applications

The simpleidb is used to specify the applications. The first one is the `Robo Movement` application which is used to perform one task, `Travel` or `Rotate`. In order to execute a bundle of tasks, another application called `Robo Tasks` application, was implemented. The second one gets a file as input, which contains commands such as "T20" or "R180". Each of the application specifications in the simpleidb starts a corresponding NXTJobClient first.

3.1.1 NXTJobClient

The status of a running LEGO MINDSTORMS NXT robot program is not available to UNICORE. After a successful upload of the program to the LEGO MINDSTORMS NXT robot, the status for this job is "successful finished" even if the execution of the program is going on. To avoid this problem the NXTJobClient and NXTJobServer were implemented. The NXTJobClient connects to the NXTJobServer, which is responsible for the execution of the tasks. The NXTJobClient submits all tasks to the NXTJobServer followed by an end signal which tells the server that all jobs have been transmitted. The successful execution feedback from the NXTJobServer to the NXTJobClient terminates the NXTJobClient and the next job can be started.

3.2 NXTJobServer

The NXTJobServer maintains a constant bluetooth connection to the LEGO MINDSTORMS NXT robot. The reason for using a server is to avoid the time intensive bluetooth connection and disconnection process for every single task. The NXTJobServer is actually a TCP server, which waits for incoming connections transferring tasks from NXTJobClients.
If more than one user requests a connection to the LEGO MINDSTORMS NXT robot at the same time, multiple connections will be generated. The server uses the NXTJobScheduler to choose the client connection, which has transferred all tasks to it. All other connections will be pushed to the end of a queue on server side, which stores all waiting NXTJobClient requests.

The server consists of multiple components:

- One thread for each client connection.
- One NXTJobScheduler, that selects the client connection which has submitted all tasks to the server.
- One worker class, which executes the tasks remotely or locally.

3.2.1 NXTJobScheduler

The NXTJobScheduler is used to choose the client connection in the queue, which has finished the transfer of all tasks to the server. This ensures that the execution of tasks from different connections will not be mixed up and that jobs with a large list of tasks can not block small jobs. To have a real-time execution of generated tasks, piping between UNICORE Jobs is required. This feature is not yet implemented[5]. The other way to achieve real-time execution is to integrate the NXTJobClient into the compute intensive task-generation-job, but that would destroy the separation between compute and execute jobs.

3.2.2 Workers

There are two different workers available. The first one is a worker which remotely controls the LEGO MINDSTORMS NXT robot, e.g. to set the speed of motors.
The second worker communicates with a program running on the LEGO MINDSTORMS NXT robot, which receives tasks for the LEGO MINDSTORMS NXT robot.
The difference between both is the precision of the movement. The remote controlled robot does not move as exactly as the locally controlled robot. The reason is the feedback received from the motors, that are needed to control the robot. It has to be transferred over the slow bluetooth connection first. This delay results in an inaccurate movement.

4 Conclusion & Future Work

This paper shows the integration of applications in UNICORE, which are used to operate a LEGO MINDSTORMS NXT robot. Besides the configuration in UNICORE, the implementation of multiple components was necessary to ensure an efficient and stable processing of tasks. Based on a small use case, the concept was successfully proved.
To have a real-time execution of generated tasks, UNICORE should be able to pipe input and output between Jobs, not only files. Another issue is to simplify the UNICORE workflow system. The UNICORE workflow system consists of two components, the workflow engine, which is responsible for the execution of the workflow in the right order, and the service orchestrator, which organises the choice of the right target system. In case of the applications described above, that execute Java programs every time on the same machine, the process of choosing a target system is redundant. In order to speed up the UNICORE job execution of the use case described in section 2 the workflow system must be simplified concerning the scheduling mechanism.
This method of controlling a LEGO MINDSTORMS NXT robot can be used to demonstrate what UNICORE features are provided to the user; especially in UNICORE courses, students can learn how easy it is to integrate new applications into a Grid.

References

1. NI LabVIEW for LEGO MINDSTORMS, `http://www.ni.com/academic/mindstorms/`, accessed May 2012.
2. LEGO MINDSTORMS NXT Kit contents, `http://mindstorms.lego.com/en-us/history/default.aspx`, accessed May 2012.
3. LEJOS - Java for LEGO Mindstorms, `http://lejos.sourceforge.net/`, accessed May 2012.
4. Rebecca Breu and Sandra Bergmann, *Integration of Applications in UNICORE 6*, `http://www.unicore.eu/documentation/presentations/unicore6/files/03_Breu.pdf`.
5. Krzysztof Benedyczak, Aleksander Nowiński and Piotr Bała, *Flexible Streaming Infrastructure for UNICORE*, Lecture Notes in Computer Science, in Euro-Par 2007 Workshops: Parallel Processing, Volume 4854, 2008, DOI: 10.1007/978-3-540-78474-6, 94-103, `http://dx.doi.org/10.1007/978-3-540-78474-6_13`.
6. Sulev Sild, Uko Maran, Andre Lomaka, and Mati Karelson, *Open Computing Grid for Molecular Science and Engineering*, Journal of Chemical Information and Modeling 2006, 46 (3), 953-959.
7. Mathilde Romberg, *UIMA-HPC: High-Performance Knowledge Mining*, inSiDE Journal, Gauss Centre for Supercomputing, Autumn 2011, Vol. 9 No. 2.

A Service-Oriented Approach of Integration of Computer-Aided Engineering Systems in Distributed Computing Environments

Gleb Radchenko[1] **and Elena Hudyakova**[1]

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

Computer-Aided Engineering (CAE) systems demand a vast amount of computing resources to simulate modern hi-tech products. In this paper we consider the problem-oriented approach to access remote distributed supercomputer resources using the concept of distributed virtual test bed (DiVTB). DiVTB provides a problem-oriented user interface to distributed computing resources within the grid, online launch of CAE simulation, automated search, monitoring and allocation of computing resources for carrying out the virtual experiments. To support the concept of DiVTB the CAEBeans technology was developed. CAEBeans technology provides a solution for the development and deployment of DiVTB, integration of most common CAE systems into the distributed computing environment as grid services (based on the UNICORE grid middleware) and web access to CAE simulation processes.

1 Introduction

Computer-Aided Engineering systems are now one of the key factors ensuring competitiveness of any high-tech production. The use of such systems makes it possible to conduct virtual experiments where real-life experiments are difficult or impossible. This can greatly improve the accuracy of case analysis and design decisions and shorten the way from the generation of an idea to the real industrial production[1].

The accuracy of the computer simulation depends on the level of detail of meshes used for computational experiments. The computational complexity of problems of engineering analysis keeps increasing, and their solutions require significant computing resources. The only way is to use multiprocessor systems.

For an ordinary user the process of solving of engineering problems involving supercomputing resources is associated with certain difficulties. First of all, the interface and features of all software components that provide the simulation technological cycle need to be explored[2]. Second, specific knowledge and skills in the high performance computing are required. And last but not least, the cost of the purchase, installation and administration of powerful supercomputer equipment and software is extremely high. All these factors complicate the process of embedding powerful engineering methods in the research and development of traditional manufacturing companies.

An alternative to creating your own supercomputer centre is renting remote computing resources from the supercomputer centres which offer utility computing services. The utility computing assumes that diverse computational resources can be brought together on demand and that computations can be implemented depending on demand and service load[3]. However, there is a range of issues related to the mechanism of remote resources

usage. Issues of secure data exchange, transparency of allocation and use of remote resources can deter industrial users from such an approach. These issues can be resolved by the grid computing concept[4] together with cloud computing concept[5], whereby the end user has a problem-oriented service that provides problem resolution based on distributed computing resources.

In the past few years the approach to supplying the CAE systems in the form of distributed computing environment resources is booming. In[6,7] the collaborative design of a system for engineering analysis by a dynamically distributed development team is introduced. As members of such groups need to interact constantly in the development process, peer-to-peer computing was selected as the basic platform for this system. However, the use of this approach is not aimed at providing the end user with a finalized product accessing distributed network resources, but rather at supporting the collaborative design process of a development team.

In[8] a concept of the Virtual Test Bed (VTB) (a prototype of the virtual cooperative engineering environment) was introduced. The VTB has been designed as an architecture to facilitate the integrated execution of different simulation programs with other supported non-simulation software. The proposed solution is built on the Extensible Modelling and Simulation Framework (XMSF) platform, supports such distributed computing approaches as CORBA, SOAP, DCOM and is highly targeted at the specific field of application (the development of new generation space crafts) which does not restrict its use in a related area of CAE simulation.

In[9] the REST web-services architecture for distributed simulations is introduced. The proposed approach supports interoperability of distributed computing resources and robust data exchange during simulation process.

The analysis of virtual test beds for CAE systems shows:

- the lack of a unified approach for creating virtual test beds allowing CAE simulation in the distributed computing environment;
- the lack of adaptation of the VTB problem-oriented user interface for the needs of the certain categories of users and the lack of streamlined web interface for inserting the tasks;
- lack of support for standard grid protocols interaction;
- lack of problem-oriented resource broker, which would transparently provide resources of distributed computing environment (including hardware, software and license resources) in compliance with the current task requirements.

In[10] the concept of GridBean in GPE grid middleware platform was introduced. A GridBean is an object responsible for generating the job description for grid services and providing graphical user interface for input and output data. This platform introduced an easy way to integrate legacy stand-alone applications in the grid environment as grid services. Subsequently the GridBean concept was integrated into the UNICORE middleware platform[11].

In this paper we provide a review of the Distributed Virtual Test Bed (DiVTB) concept to provide resources of CAE systems in the grid computing environment and the CAE-Beans technology, a combination of concepts and techniques used for the development and

operation of DiVTB. The UNICORE grid middleware is used to implement grid services that integrate CAE systems in grid environment.

2 The Distributed Virtual Test Bed

2.1 Basic Terms

Let a CAE-problem be assumed as the combination of:

- a geometric model of a research target and/or computational mesh which divides a simulated area into discrete sub-areas;
- boundary conditions, physical characteristics and parameters of the components interaction in a simulated area;
- a description of unknown variables which values will be obtained as a result;
- the requirements for hardware and software resources that provide a simulation process.

Let a CAE-parameter be the value, which affects the final result of simulation and can vary within a certain range. Each parameter has specific semantics and describes a feature of the subject area or a solution process.

We offer a concept of a Distributed Virtual Test Bed (DiVTB) to provide access to resources of CAE systems in a distributed computing environment. DiVTB is a software solution which provides an engineering simulation in the grid computing environment for a certain CAE-problem. DiVTB includes (Figure 1):

- an interface for a CAE-problem input;
- a driver (a set of software tools enabling the use of grid resources for virtual experiment);
- a set of grid services (a set of supercomputers in a distributed computing environment, which uses the installed software components to provide the solution of a CAE-problem and implements safe standardized remote communication methods).

A combination of concepts and techniques for DiVTB development and operation is defined as *the CAEBeans Technology*.

The CAEBeans Technology includes:

- the DiVTB structure concepts and development techniques;
- the administrative concepts: the patterns of work and the distribution of duties between the DiVTB developers and users;
- the software solution for the DiVTB development, deployment and operation.

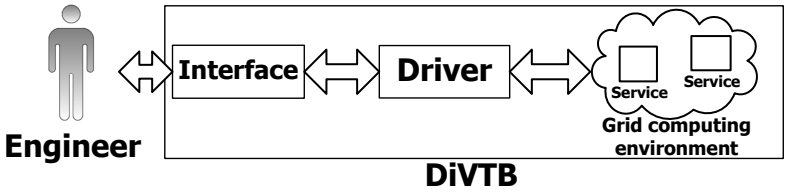

Figure 1. General diagram of the grid distributed virtual test bed (DiVTB)

2.2 The Structure of DiVTB

CAEBean is a primary structural unit of DiVTB. According to the CAEBeans technology we distinguish 4 structural levels of DiVTB, each represented by its CAEBean type (Figure 2):

- Conceptual level (a hierarchy of *conceptual CAEBeans*);
- Workflow level (a *workflow* CAEBean);
- Physical level (consists of a set of *physical CAEBeans*);
- System level (consists of a set of *system CAEBeans*).

A conceptual level of DiVTB is formed based on the conceptual CAEBeans. A conceptual CAEBean defines the *problem-oriented interface for DiVTB* based on a set of its CAE-parameters. Through a conceptual CAEBean a user can launch the DiVTB simulation, follow a solution process and obtain the results. The CAEBeans Technology provides the opportunity to design a *generalization hierarchy* of conceptual CAEBeans. The conceptual CAEBeans of lower hierarchy levels adapt a problem-oriented interface of the DiVTB for the needs of a certain category of users. These CAEBeans specify the CAE-problem defined by a parent conceptual CAEBean through assigning specified values to some CAE-parameters.

The workflow level of DiVTB is represented by a workflow CAEBean implementing a CAE-workflow for a certain CAE-problem. Let a CAE-workflow be defined as a directed graph with the following types of nodes:

- an action performed by a certain CAE system;
- a node controlling the problem solution flow.

We use the elements from the UML 2.0 activity diagram notation to form a CAE-workflow. The information required to initiate any node of a workflow is stored in a *complete descriptor* of DiVTB. Let a complete descriptor be defined as a set of CAE-parameters that uniquely describes a CAE-problem which can be simulated by DiVTB. Any workflow node as a part of a solution can refer to a *complete descriptor* for input values and store results in the corresponding output parameters.

A physical level of DiVTB is represented by the physical CAEBeans. The main function of a physical CAEBean is a conversion of a problem-oriented description of a CAE-problem (a set of parameters in complete descriptor) into the set of files needed to launch

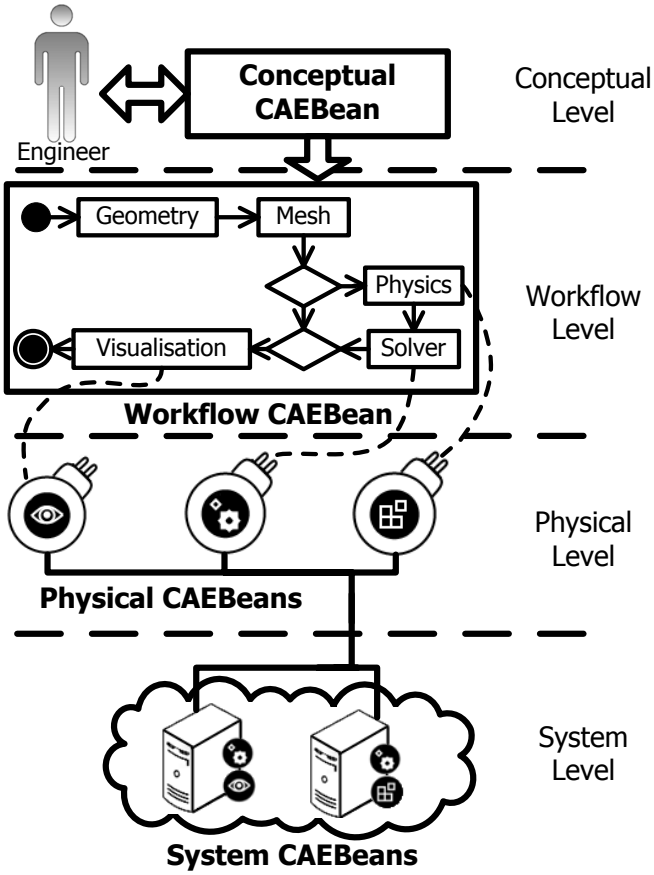

Figure 2. A generalized scheme of DiVTB levels according to the CAEBeans Technology

a single *CAE-action* (mesh generation, simulation, post processing, optimization etc.) on a specific CAE system. At the end of a problem solving process a physical CAEBean provides conversion of component-oriented results in a problem-oriented form. During the CAE-workflow execution each physical CAEBean must be matched with a grid service to execute a CAE-action. All grid services are implemented through the CAEBeans system. A workflow CAEBean together with the corresponding physical CAEBeans implement *a driver* of DiVTB.

A system CAEBean provides the functionality of a computing resource in a distributed computing environment and implements a service-oriented approach to task input and fetching the results. A system CAEBean provides the isolated workspace for each CAE-action and also the programming interface that allows initial data loading, remote task input and delivery of results.

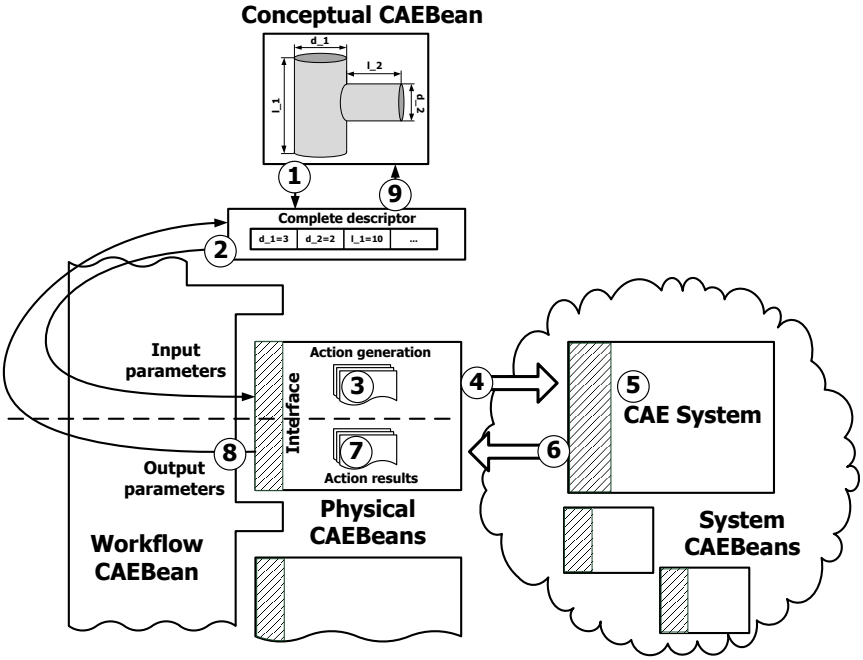

Figure 3. Simulation of DiVTB according to the CAEBeans Technology

2.3 DiVTB Simulation

Let us consider the process of DiVTB simulation according to the CAEBeans Technology (Figure 3).

1. An engineer inputs the values of CAE-parameters for DiVTB by means of user interface provided by an appropriate conceptual CAEBean. The conceptual CAEBean calculates values of CAE-parameters which are not available to the user on the current level of generalization hierarchy. All initial CAE parameter values are stored in the complete descriptor of DiVTB. Finally the engineer launches the simulation process.

2. The workflow CAEBean executes the CAE-actions according to the CAE-workflow. The values of initial parameters of any action (to be defined before the action starts) are taken from the complete descriptor of DiVTB and transferred to the physical CAEBean.

3. The physical CAEBean generates a set of input files for the CAE system wrapped in the corresponding CAEBean system.

4. The resource broker gives an address of the CAEBean system with the best opportunity to execute the CAE-action. The physical CAEBean transfers a set of input files to the allocated CAEBean system and starts the CAE-action.

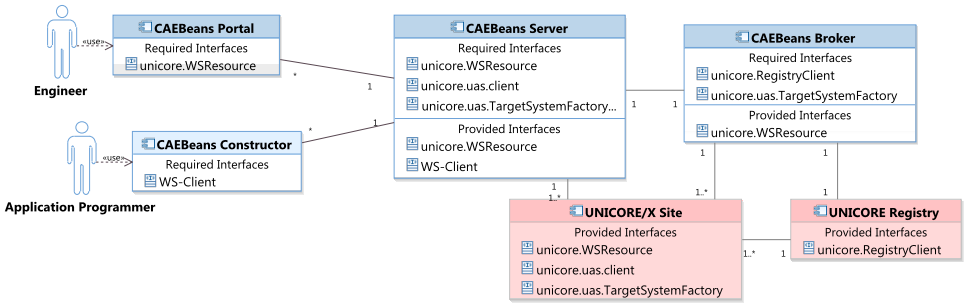

Figure 4. General diagram of CAEBeans components interaction

5. The CAE system executes the CAE-action and generates a set of result files.
6. The Physical CAEBean receives the result files from the CAEBean system.
7. The physical CAEBean parses the results of the action and retrieves the values of the required output parameters.
8. The physical CAEBean puts the obtained values of output parameters into the complete descriptor. The workflow CAEBean triggers the next node of the CAE-workflow (step 2).
9. When the CAE-workflow is complete, all results of the DiVTB simulation are transferred to the conceptual CAEBean from the complete descriptor.

3 The CAEBeans System

To implement the CAEBeans Technology we created a CAEBeans System — a software solution for DiVTB development and implementation. We selected the UNICORE 6 grid computing middleware[11] 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. The CAEBeans System consists of the following components (Figure 4):

- *CAEBeans Constructor* — integrated development environment of DiVTB. It provides the interface for an application programmer to develop the CAEBeans of conceptual, workflow and physical levels. Accordingly, the CAEBeans Constructor user interface is divided into three sections responsible for developing an corresponding type of CAEBean.
- *CAEBeans Portal* — a web application that provides a user interface for the input and solution of CAE-problems by means of an appropriate DiVTB. The web-form generator automatically builds DiVTB user interfaces based on the corresponding conceptual CAEBeans. Also, the CAEBeans Portal supports authentication and user account management.

- *CAEBeans Server* — a grid service for DiVTB storage and simulation. A process of execution of a CAE- workflow is supported by the internal workflow subsystem of the CAEBeans Server. CAEBeans Server provides interfaces for CAEBeans Constructor and CAEBeans Portal allowing to upload a DiVTB to a Server and to start a simulation of a DiVTB.

- *CAEBeans Broker* — an automated system for registration, analysis and allocation of CAE-resources for CAE-action execution. The CAEBeans Broker receives requests for resources from CAEBeans Server, analyses the status of the distributed computing environment and supplies the CAEBeans Server with a CAE- resource best suited to the request. CAEBeans Broker interacts with an UNICORE Registry service to gain an information about resources available in the grid environment.

- *CAE-Resources* — grid services that implement the remote execution of CAE-Actions by means of CAE-systems installed on specific computers in a grid (instances of system CAEBeans). Each CAE-resource is a UNICORE/X site with a set of problem-oriented and system software installed on it. CAE-resource provides:

  - obtaining problem data from CAEBeans Server or an external data source;
  - CAE-action initiation and automated solution;
  - transmission of results into the CAEBeans Server or external data storage.

4 CAEBeans System Test

To test the CAEBeans system we developed DiVTB to solve problems by using common CAE systems: ANSYS CFX, ANSYS Mechanical, ABAQUS, DEFORM, LS-DYNA. CAEBeans system was deployed on the basis of the supercomputer resources of South Ural State University Supercomputer Simulation Laboratory[12].

For the test we chose the issue of tempering and cooling of tubes and analysis of impact of various aspects of the quenching process on the quality of the production ("Thermal Treatment Distributed virtual test bed") on the basis of DEFORM system[13]. This issue was stated by JSC "Chelyabinsk Tube Rolling Plant".

Firstly we created a model of a heat treatment process. We supplied thermal studies of the pipe hardening. We collected information about temperature fields during the quenching process and the magnitude of the initial circumference and curvature along the axis of the work piece. The "Thermal Treatment" DiVTB allows to change the flowing parameters of the heat treatment process: number of inductors, frequency and current strength, a length of the inductors, number and configuration of the jets of water, pressure of cooling flow, velocity of pipe motion, pipe steel grades (Figure 5).

Another application of DiVTB was a virtual test bed for supercomputer simulation of deformation of knitted fabrics on human body[14]. We designed and deployed a DiVTB on the basis of LS-DYNA CAE system that allow to simulate the process of putting on knitted clothes to a torso of a human body. By varying parameters, such as density and method of weaving of knitted material, engineer can analyse the quality of the fit of the final product, view a video of the process of dressing, etc.

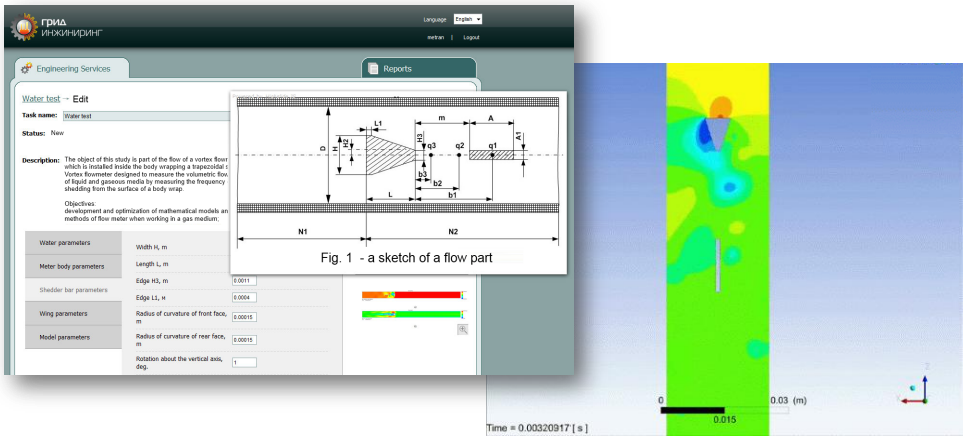

Figure 5. Example of DiVTB user interface (CAEBeans Portal)

Acknowledgements

The work is supported by grants of Council of President of Russian Federation (project MK-1987.2011.9), Russian Foundation for Fundamental Research (project number 11-07-00 478-a) and Ministry of Education and Science of Russian Federation (state task 8.3786.2011).

References

1. B. Raphael, I. F. C. Smith, *Fundamentals of computer aided engineering* (John Wiley, 2003).
2. T. M. Buriola, S. Scheer, *CAD and CAE Integration Through Scientific Visualization Techniques for Illumination Design*, Tsinghua Science & Technology **13(1)**, 26–33, 2008.
3. M. N. Huhns, P. M. Singh, *Service-Oriented Computing: Key Concepts and Principles*, IEEE Internet Computing **9(1)**, 2–8, 2005.
4. I. Foster, C. Kesselman, *The Grid 2, Second Edition: Blueprint for a New Computing Infrastructure* (Morgan Kaufman, San Francisco, 2003).
5. B. Hayes, *Cloud computing*, Communication of the ACM **51(7)**, 9–11, 2008.
6. L. Q. Fan et al., *Development of a distributed collaborative design framework within peer-to-peer environment*, Computer-Aided Design **40**, 891–904, 2008.
7. J. W. Yin, W. Y. Zhang, Y. Li, H. W. Chen, *A peer-to-peer- based multi-agent framework for decentralized grid workflow management in collaborative design*, The International Journal of Advanced Manufacturing Technology **41(3-4)**, 407–420, 2009.
8. J. Sepulveda, L. Rabelo, J. Park, F. Riddick, C. Peaden, *Implementing the high level architecture in the Virtual Test Bed*, in: WSC'04 Proceedings of the 36th conference on Winter simulation pp. 1444–1451, 2004.

9. K. Al-Zoubi, G. Wainer, *Using REST Web-Services Architecture for Distributed Simulation*, in: PADS '09 Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation, pp. 114–121, 2009.
10. R. Ratering et al. *GridBeans: Support e-Science and Grid Applications*, in: E-SCIENCE '06 Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, pp. 45., 2006.
11. A. Streit et al.: *UNICORE 6 Recent and Future Advancements*, Annals of Telecommunications **65(11)**, 757–762, 2010.
12. G. I. Radchenko, *A service-oriented approach of integration of engineering design and analysis systems in distributed computing environments (in Russian)*, in: Parallel Computing Technologies (PaVT'2011): Proceedings of the International Scientific Conference (Moscow, March 28 - April 1, 2011), pp. 606–616, 2011.
13. V. A. Dorokhov, *Development of a grid virtual test-bed for study the effect of tube ovalisation during heat treatment (in Russian)*, in: Parallel Computing Technologies (PaVT'2009): Proceedings of the International Scientific Conference (Nizhny Novgorod, March 30 - April 3, 2009), pp. 457–462, 2009.
14. N. Y. Dolganina, E. A. Zaharov, A. V. Shamakina *Virtual Test Bed for Supercomputer Simulation of Deformation of Knitted Fabrics on Human Body (in Russian)*, in: Gergel, V.P. (ed.) Proceedings of XI all- Russian conference, pp.118–122, 2011.

Brokering Service for Supporting Problem-Oriented Grid Environments

Anastasia Shamakina

Supercomputer Simulation Laboratory,
76, Lenin Ave, South Ural State University, 454080 Chelyabinsk, Russia
E-mail: {sham2004}@bk.ru

Nowadays many planners in Grid environment support scheduling are based on a workflow. However, none of currently existing tools uses additional information concerning specifics of a problem area and the representation of a workflow. This paper describes a scheduling algorithm and architecture of a CAEBeans resources broker. It considers the above aspects as well as utilises resource reservation, thus managing hardware, software and licenses. This broker can be used for an effective search of resources in problem-oriented grid environments. The CAEBeans Broker is implemented in Java as a UNICORE service. This approach involves component independence from a computing platform and provides full information about a current state of a service, and supports secure and reliable performance, lifetime management, dispatch change notifications, management policy of access to the resources and access control certificates.

1 Introduction

Grid computing[1] intends to solve many tasks in various fields, e.g. medicine, engineering design, nanotechnology, climate prediction and others. In the present paper we consider a CAEBeans system[2] that provides access to CAE software (Computer Aided Engineering[3]). The CAEBeans system allows to carry out task decomposition, search for demand resources; submit tasks; monitor tasks execution; and provide results mapping for the user.

In common, any CAE-task solution includes the following technological steps: geometry creation, computing grid generation, definition of boundary conditions, simulation execution, visualization and solution analysis. The CAE-tasks have some peculiarities that should consider characteristics of resources, a set of engineering packages, the number of licenses available and so on.

Many grid environments support complex applications with a workflow[4]. They are usually modelled by a directed acyclic graph (DAG). Such tools as Condor DAGMan[5], CoG[6], Pegasus[7], GridFlow[8] and ASKALON[9] can be listed. Some additional information about problem area specifics and workflow representation can significantly improve the efficiency of planning resources methods. However, none of the currently existing tools consider this peculiarity.

The purpose of this work is to create methods and algorithms for scheduling a workflow, as well as the development of a resource broker based on it. The broker is applied to search for optimum resources in problem-oriented grid environments.

The article is structured as follows. Section 2 contains the main concepts of problem-oriented environments; section 3 describes some use cases regarding the resource broker; section 4 provides the architecture of the resources broker; section 5 considers the process of resource allocation in the CAEBeans Broker component; and section 6 gives a scheduling algorithm. The main results are summarised in the concluding part of this paper.

2 Basic Concepts of Problem-Oriented Environments

Here, we give some definitions of the basic concepts that are necessary to determine a problem-oriented environment.

Task describes the process the transformation of input parameters into output parameters.

Job is a set of Tasks organised as a workflow aimed to achieve a useful result. Job determines the order of tasks execution, conditions under which this or that task will be started, the mutual synchronisation of tasks and information flows between tasks.

Resource is hardware, software and licenses required to perform a task.

Service is a specification of resources to solve a specific class of Tasks. Service defines the format of the input and output data.

Abstract workflow is a set of a Job and requirements for resources.

Activity is an allocation of necessary resources and launch of a specific service with respect to specific recourses for executing a specific Task.

Concrete workflow is a workflow of activities aimed at implementation of a specific Job.

Distributed Problem-Oriented Environment is a set of resources, services, software and middleware focused on the implementation of Concrete workflows for a specific problem domain.

3 The CAEBeans Broker of Resources

The CAEBeans Broker is a component of the CAEBeans system that receives a job from clients, accords resource requirements and finds the most suitable computing nodes for each task from a workflow. We can distinguish the following main tasks of a broker: processing a resource database and analysing resource requests from a CAEBean Server[10]; gathering and providing information about the current state of a problem-oriented environment.

The CAEBeans Broker is implemented in Java as a UNICORE service. This approach involves component independence from a computing platform; provides full information about the current state of the service, and supports secure and reliable performance, lifetime management, dispatch change notifications, management policy of access to resources and access control certificates.

Software component CAEBeans Broker is a service that provides an interface for selecting the most appropriate resources. The search of required resources proceeds for multiple jobs once a resource broker works. In addition, an auxiliary component for data collection interacts with the resource broker. A use case diagram for the CAEBeans Broker is shown in Figure 1.

In the use case “Allocate Resources” sends a request from a client to the resource broker. The CAEBeans Server works as a client and responds to the execution of tasks and their monitoring. The use case begins when a CAEBeans Server specifies requirements for resources which are necessary for the job. The use case “Allocate Resources” includes that of “Set Reservation”, which allows one to set the reservation of selected resources.

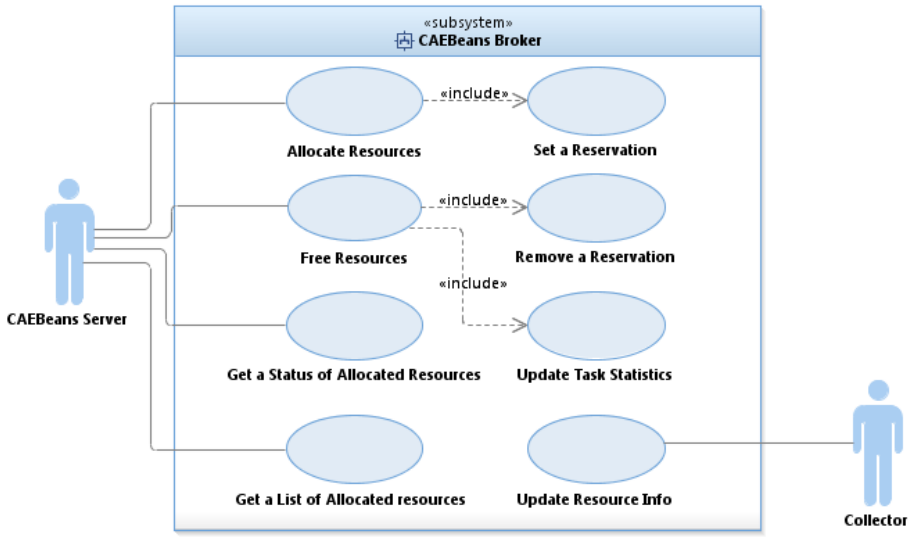

Figure 1. A use case diagram of a CAEBeans Broker subsystem

The use case "Free Resources" notifies the resource broker of a necessity for releasing some chosen resources. This use case also assumes the removal of a resources reservation request[11]. A corresponding removal request is formed by the use case "Remove Reservation".

The use case "Get a Status of Allocated Resources" asks for information about resources allocation. When resources allocation succeeds, a CAEBeans Server asks for a list of allocated resources using the case "Get a List of Allocated Resources".

Use cases "Update Resource Info" and "Update Task Statistics" update the information about characteristics of resources and statistics of tasks performance in the database of the resource broker.

4 The Architecture of the CAEBeans Broker

According to the use case diagram shown in the Figure 1, the following architecture of the resource broker CAEBeans Broker was designed (Figure 2). The CAEBeans Broker consists of the following components:

- The Master accepts requests from a CAEBeans Server and creates an instance of a WSResource that is called a Brokered Workflow.
- Each Brokered Workflow processes a request. It generates a list of required resources to perform a workflow and make a reservation.
- The Component Resource Manager manages the Resource Database that contains information about target systems and reserved resources.

- The Component Statistics Manager manages the Statistics Database that contains information about tasks statistics.
- The Collector works independently from the CAEBeans Broker and carries out the collection of information for the Resource Database.

In the Figure 2 the components of the UNICORE platform that intend to cooperate with the CAEBeans system are marked with red colour.

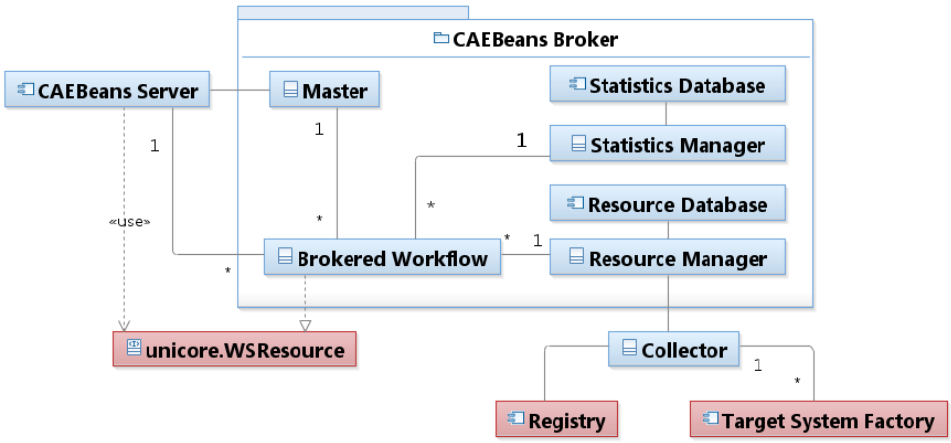

Figure 2. The architecture of the CAEBeans Broker

5 Resource Allocation of the CAEBeans Broker Component

According to the use case diagram shown in the Figure 1 the resource broker provides some methods for resource allocation and information gathering.

Consider how a CAEBeans Broker service ensures the process of finding necessary resources for the job (Figure 3).

A CAEBeans Server component provides access to the main service Master. Then the CAEBeans Server calls for the “Allocate resources” method for workflow resource allocation, as well as passes an abstract workflow as an input parameter. The abstract workflow contains requirements for resources.

The Master creates an instance of a Brokered Workflow, representing a WSResource in the terms of UNICORE. The Master initialises a concrete workflow, obtaining necessary information from the abstract workflow. It writes down the concrete workflow by the Ehcache framework in a cache at first and then on a disk. The Master returns the unique identifier of the created instance to the CAEBeans Server.

The Brokered Workflow gets a run-time estimate for each task and starts a search for demanded resources in the Resource Database. If the search for the Brokered Workflow succeeds, a list of resources is received. The selected resources are reserved.

The CAEBeans Server checks the status of resource allocation by directly addressing the instance of the Brokered Workflow. In case of a successful resource allocation, the CAEBeans Server requests for a list of the allocated resources for its workflow. After the workflow execution the CAEBeans Server sends a request for releasing the resources to the Master.

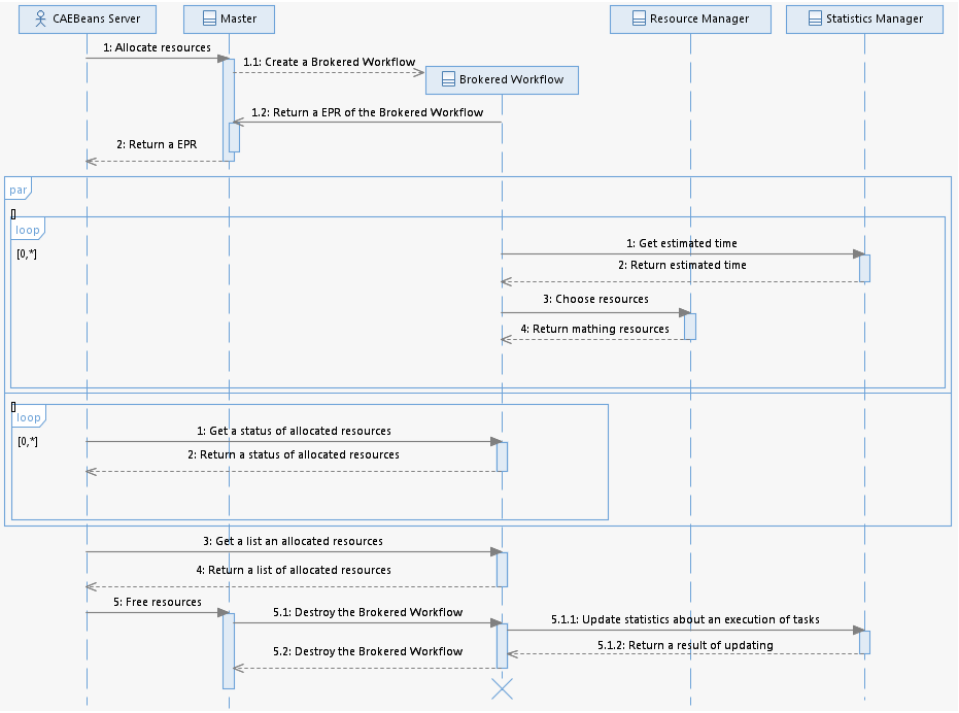

Figure 3. A sequence diagram of resource allocation by the CAEBeans Broker

The process of information gathering for the Resource Database is presented in the Figure 4. Updating steps repeat themselves consistently during the whole time of the Collector's execution. Updating information about resources signifies gathering data concerning hardware, software and licenses that is available for the CAEBeans Broker. The statistics of task performance includes information about the real time of tasks performance with specific parameters and on specific computing nodes.

6 Scheduling Algorithm of the CAEBeans Broker

The present work suggests developing a scheduling algorithm for distributed problem-oriented computing environments and considers some additional information regarding specifics of a problem area and workflow representation. It discusses using resource reservation and managing hardware, software and licenses in distributed computing environments.

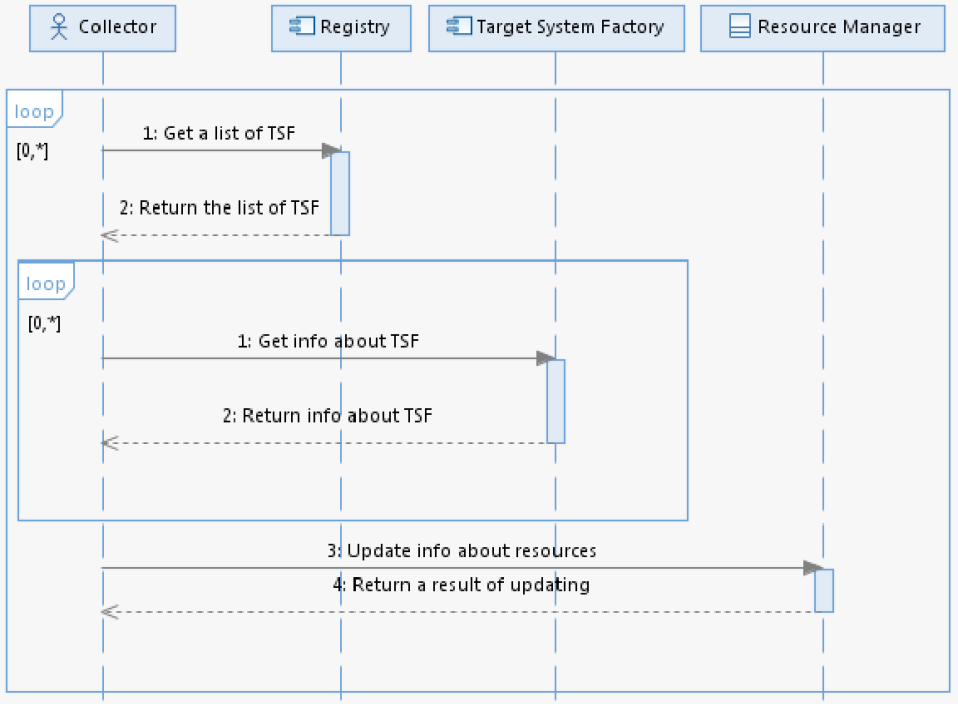

Figure 4. A sequence diagram of information gathering by the CAEBeans Broker

Here, the following methods are used:

- the method of two-phase resource reservation;
- the method of dominant sequence clustering;
- the accounting of task problem parameters for estimating its run-time performance.

The method of two-phase resource reservation. Reservation of resources in distributed computing environments is now a complex challenge, as it demands for existence of a component for a centralized scheduler of resources and for instant decision-making regarding reservation of resources in a task control system. The method of two-phase resource reservation is offered for the solution of this problem. This method allows carrying out preliminary reservation (marking) of demanded resources at the first phase, at the same time prospecting an optimum schedule for a workflow. Some time, confined by the priority of workflow performance, is necessary for making a plan of a workflow. After this timeout, the first phase finishes. Within the second phase, a final workflow resource allocation is completed or a release of all marked resources is carried out. In both cases, the allocation/release operations are carried out atomically.

The method of dominant sequence clustering. The workflow mapping to computing resources is implemented by the adapted method of dominant sequence clustering (DSC[12]). The essence of the DSC method lies in the minimisation of parallel time spent on workflow

performance. However, in the original DSC method scheduling is carried out in two steps. At the first step, there proceeds an association of several tasks into clusters (groups), at the second step, mapping of task clusters to real computing resources is performed. The adaptation of this method means the usage of preliminary resource reservation at the step of the association of clusters: a search for the free slots on computing resources will be done directly when workflow scheduling is accomplished.

The accounting of task problem parameters for estimating of its run-time performance. The time of the performance of separate tasks in a workflow is specified by a user default. However, in most cases users tend to overestimate the time of execution. The task statistics is stored in a Statistics Database to provide more exact assessment of time required for task performance. The database is managed by the Statistics Manager. The Statistics Manager provides information about the exact time of task execution based on the stored data concerning task parameters and the architecture of computing nodes that this task was executed on.

In the Figure 5 a scheduling algorithm of the CAEBeans Broker is presented. At the initial step, a job is presented in the form of a workflow. For each task from the job create a separate group. Every available resource has its own colour. It is necessary to paint of tasks in colours of available computational resources.

```
// Phase I
T = 0;                      // Time of the first phase
V_uncol = V;                // A set of uncolored nodes
R = ∅;                      // The set of reserved slots
while (T < T_lim){
  colored_method(G);        // The method of coloring graph G
  T++;
}
// Phase II
if (V_uncol ≠ ∅){
  for (i = 1; i = |V_uncol|; i++){
    n_i = V_nocol[i];
    s_i = eft(n_i);
    S = {s_i};
    reserve(S);
    R = R ∪ S;
  }
}
allocate(R);                // Allocate the reserved slots
```

Figure 5. A scheduling algorithm of the CAEBeans Broker

In the first phase a search of an optimal schedule to run a workflow and a resource reservation using a method of graph colouring. However, the first phase has a limit time T_{lim}. The limit time is based on the fact that a problem of finding of an optimal schedule is NP-complete.

During the execution of the second phase is searched nodes, which there were no reserved slots on computing resources. The set of uncoloured nodes is V_{uncol}. We allocate the first appropriate free slots for each node from the set V_{uncol} on computing resources. We perform a final resource allocation for all nodes.

Here a procedure $colored\_method(G)$ produces the graph colouring by the method described below. A function $eft(n) = s$ will search a slot s for the node n using an algorithm Earliest-Finish-Time.

```
st = 0;                    // A step of the method
Vuncol = V;                // A set of uncolored nodes
while (Vuncol ≠ ∅){
  DSst = domseq(Gst)       // Find a dominant sequence
  PTst = partime(Gst);     // Calculate a parallel time
  EDSst = {eij = (ni,nj), где ni,nj ∈ DSst};
  EDSst = EDSst \ {eij = (ni,nj), where ni and nj ∉ Vuncol};
  sort(EDSst);             // Sort weights of edges
  for (m = 1; m = |EDSst|; m++){
    eij = EDSst[m];
    S = search(ni,nj);  // S = {si,sj}
    if (S ≠ ∅){
      Vuncol = Vuncol \ {ni,nj}; // Subtract colored nodes
      reserve(S);              // Reserve slots
      eij = 0;
      break;
    }
  }
  st++;
}
```

Figure 6. A method of graph colouring of the CAEBeans Broker

In the Figure 6 the method of graph colouring is presented.

Here a function $domseq(G) = DS$ returns a dominant sequence, containing at least one uncoloured node. A procedure $sort(EDS)$ sorts weights descending, a procedure $reserve(S)$ reserves slots from a set S on the computational resources. A function $search(n_i, n_j) = S$ searches slots s_i and s_j, satisfying the requirements n_i, n_j respectively at the same computational resources.

7 Conclusion

In this paper, a scheduling algorithm in a distributed problem-oriented computing environment is described. It considers some additional information about specifics of a problem area and workflow representation; using resource reservation; managing hardware, software and licenses of distributed problem-oriented computing environments. Some use cases of a resource broker are presented and the process of resource allocation and architecture of the CAEBeans Broker are proposed.

The further work includes the following: conducting computational experiments for evaluating the effectiveness of a scheduling algorithm on supercomputers “SKIF-SUSU Aurora” and “SKIF Ural” at South Ural State University.

Acknowledgements

The present work is supported by grants given by the Council of President of the Russian Federation (Project MK-1987.2011.9) and the Russian Foundation for Fundamental Research (Project No.11-07-00 478-a).

References

1. I. Foster, C. Kesselman, *The Grid 2, Second Edition: Blueprint for a New Computing Infra-structure*, Morgan Kaufman, San Francisco 2003.
2. G. I. Radchenko, *A service-oriented approach of integration of engineering design and analysis systems in distributed computing environments (in Russian).* In: Parallel Computing Technologies (PaVT'2011): Proceedings of the International Scientific Conference (Moscow, March 28 - April 1, 2011), pp.606–616, SUSU Publishing House, Chelyabinsk 2011.
3. B. Raphael, I. F. C. Smith, *Fundamentals of computer aided engineering*, John Wiley, 2003.
4. J. Yu, R. Buyya, *A Taxonomy of Workflow Management Systems for Grid Computing* Grid Computing **3**, 171–200, 2005.
5. Condor `http://www.cs.wisc.edu/condor/`.
6. G. Laszewski, I.Foster et al. *CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids* // Proceedings of the ACM Java Grande 2000 Conference, pp. 97–106, CA, USA, June 2000.
7. E. Deelman, J. Blythe et al. *Pegasus: Mapping Scientific Workflows onto the Grid* // Proceedings of Grid Computing: Second European AcrossGrids Conference (AxGrids 2004), pp. 11–26, Nicosia, Cyprus, January 2004.
8. J. Cao, S. A. Jarvis et al. *GridFlow: Workflow Management for Grid Computting* // Proceedings of the 3rd International Symposium on Cluster Computing and the Grid (CCGrid03), pp. 198–205, Tokyo, Japan, May 2003.
9. M. Wieczorek, R. Prodan, T. Fahringer *Scheduling of Scientific Workflows in the ASKALON Grid Environment* ACM SIGMOD Record **34**, 56–62, 2005.
10. R. S. Fedyanina, *CAEBeans Server: a runtime environment of problem-oriented shells over engineering packages (in Russian).* In: Parallel Computing Technologies (PaVT'2010): Proceedings of the International Scientific Conference (Ufa, March 29 - April 2, 2010), pp. 621–628, SUSU Publishing House, Chelyabinsk 2010.
11. G. Mateescu, *Quality of Service on the Grid via Metascheduling with Resource Co-Scheduling and Co-Reservation* International Journal of High Performance Computing Applications **17**, 209–218, 2003.
12. T. Yang, A. Gerasoulis, *DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors* IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 9, pp. 951–967, 1994.

Next Generation of Virtual Organizations in UNICORE

Krzysztof Benedyczak[1], **Piotr Bała**[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}@icm.edu.pl

Virtual Organizations (VOs) concept is one of the fundamental building blocks of the Grid middleware. The current technology landscape is depicted, with a discussion of VO related solutions available in the well-established grid middlewares. In this paper an in-depth analysis of Virtual Organization types is presented. The current features of the UNICORE middleware are presented and compared to other approaches. The analysis shows that VOs used in production deployments do not offer the user driven resource sharing and UNICORE is no exception in this case. UNICORE lacks also some of the fundamental features in the VO area. The paper provides a detailed discussion of the identified gaps and presents planned design of user-controlled sharing of resources in UNICORE, together with other desirable improvements in the VO handling.

1 Introduction

Virtual Organizations (VOs) are one of the main pillars of the Grid middleware, providing solutions for users management, cooperation of resource providers and control of resource access. Virtual Organizations aim to assemble users and resources on the Grid level, i.e. above the administrative boundaries of sites providing resources and above the real, home organizations of participating users. This definition is extremely broad and obviously can cover a great variety of different realizations.

Despite of long research and development, the VO solutions which are present in the well-established Grid software stacks are still missing several key features, required to foster their adoption. The main problem, this paper is focused on, is that VOs used in production deployments, do not offer a possibility of *end-user driven* resource sharing. The existing solutions require administrators to set up VOs, and subsequently assign users and resources to them. Such approach is problematic for small, ad-hoc VOs, where users (otherwise authorized to use the Grid) want to create their own experiments, control co-workers and easily share results.

UNICORE is no exception in this case and does not provide support for the above use case. What is worse, UNICORE lacks also other, more fundamental features from the VO area, like dynamic assignment of local accounts for VO members. This makes UNICORE VOs administrator driven and also less useful. Therefore it is not surprising that the most of the UNICORE deployments, including the biggest DEISA/PRACE and D-Grid, are not using the VO infrastructure.

This paper is organized as follows: an in-depth analysis of types of Virtual Organizations is presented in the section 2 and the most popular solutions used to obtain the VO-related data are shown in section 3. Selected and important parts of the current technology landscape are presented in the subsequent section 4. The existing features of the

UNICORE middleware are also presented there.

The main part of the paper, which can be found in the section 5, contains a detailed discussion of the planned design of the user-driven sharing of resources in UNICORE, together with required improvements in the VO handling. The challenges are assigned to the UNICORE components from all middleware tiers. Finally a short outlook into novel solutions which can be built on top of the planned developments is provided.

The paper assumes that the reader is familiar with the UNICORE middleware[1], and has at least a basic knowledge of general security concepts: authentication, authorization, Public Key Infrastructure, SAML[2] and XACML[3].

2 Classification of Virtual Organizations

Virtual Organization (VO) concept was bring to life by I. Foster and C. Kesselman in their publication *The Anatomy of the Grid*[4]. The broad definition which can be found there, states that Virtual Organization is *a set of individuals and/or institutions* which are sharing resources together in a highly controlled way. In the case of resources, we can think about data, instruments, software or even direct access to machines.

During the last decade a lot of effort was undertaken to implement the solutions helping to create, deploy and manage the "highly controlled sharing of resources". As the approaches were focused on different aspects of sharing and control, it is desirable to classify them with respect to several key features. The two fundamental and quite independent are: dynamism and relationships complexity.

Highly dynamic VOs are medium or short lived, typically do not contain many members and are created *ad-hoc*. For instance several colleagues from one or several institutions, working on a common scientific project, can form such a dynamic VO to share work results or experiments input. Resources in such VOs are typically composed from the physical machines to which the VO members have access anyway, and data which is controlled by the VO members.

Static VOs are usually big and long lasting. Their main use is to provide an access to the large scale hardware for significant number of users. Sharing can be a constrained by an agreement, usually difficult to negotiate.

The second axis of VOs classification is related to the complexity of the rules governing the sharing of the VO resources. As examples of complex rules we can enumerate:

- a VO can guarantee its members that each of them is granted a specified amount of execution time on the computing infrastructure,
- a VO can control license rules by allowing only a specified, maximum number of concurrent VO users of a restricted application,
- and all sorts of financial contracts.

The less constraint VO models, can offer only a minimal relationships, what usually boils down simply to authorization of the resource access.

It is worth to note that we are mostly interested in the relationships which are machine-checkable. Negotiation of contracts (for instance specifying the amount of human staff or intended scientific area for which the resources are being used) is not considered here as it can be seen as a separate, independent problem.

Virtual Organizations	*Federations*
More centralized.	Fully distributed.
A separate independent "organization" on its own.	All independent organizations and trusting each other.
Not constrained or created as an agreement on VO access rules.	Created as an agreement between organizations on basic authorization attributes semantics.
Authorization is based primarily on the VO decision.	Authorization is based on the home organization decision.

Table 1. Comparison of the most distinct properties of Virtual Organizations and Federations

Finally it is good to relate Virtual Organizations to Federations, which are a very similar concept. While Virtual Organizations are technically managed by a dedicated entity, the federation is based on the is fully distributed control. In the table 1 we compare the most important features and principles of the Virtual Organizations and Federations.

Bare federations are bit less appealing for the Grid community, where large scale resources access control must be controlled in a strict way. However Federations can act as a provider of user identity also for Virtual Organizations, so users can use the same login credentials inside the Federation and Virtual Organization.

3 Provisioning of the VO Information

The information about the VO membership, and the VO-related data, which is typically a set of attributes, must be obtained during getting access to the resources. This information can be used for making an authorization decision, accounting, setting up the user's environment and more, depending on the VO adoption scenario.

In the highly distributed Grid environments, distribution of this information is typically performed in one of the two ways, presented on the diagram 1. In the simple *pull* mode the server asks the VO service for the VO related information on behalf of the user. The biggest advantage of this approach is that this is the simplest solution from the user point of view, who, can be even unaware of the existence of the Virtual Organizations. Unfortunately this approach has also severe drawbacks, namely:

- the target server must be authorized to obtain the information from the VO server,
- when the target server trusts several VO servers (regardless of the number of VOs they are maintaining) it can easily happen that each of them must be queried to find the user information, what can constitute a large overhead,
- finally if the user is a member of several VOs, it is impossible to decide which VO should be effective, without an additional input from the user.

Those problems are addressed by the *push* model, where a user on her own obtains the VO data (in a form of a signed assertion) and propagates it to the target services. This approach solves the disadvantages of the pull model but the user is exposed to the burden of

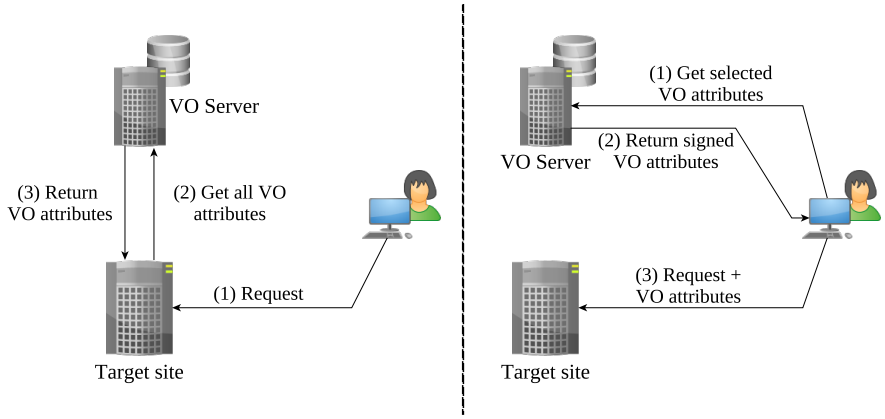

Figure 1. The typical ways of provisioning the VO-related information in the Grid: the pull mode (left) and the push mode (right).

the initial contact with the VO service. Therefore the pull model implementation requires an excellent support in client side tools.

4 The Current Technology Landscape

4.1 VOMS

VOMS[5] is a principal part of the WLCG grid and the gLite middleware, it is also used in NorduGrid ARC. Therefore the VOMS servers currently maintain the largest amount of Grid users.

VOMS service organizes members in a hierarchical groups structure. Additionally VOMS can be used to assign a special role attribute, which is valid only in the specified group context and so called "generic attributes" which are valid in the whole VO context. One VOMS instance can handle only one Virtual Organization. This fact and the internal authorization mechanism determine that VOs in gLite are created and maintained purely by the designated administrators. End-users can apply for the VO membership and, if accepted, for the VO credentials. VOMS releases attributes in a so called VOMS-proxy. VOMS-proxy is a standard Grid proxy certificate with an additional extension containing an attribute certificate. This attribute certificate carries the VO-related attributes.

VOMS-proxy certificates are used for bootstrapping SSL connections, so it forms a typical push scenario. VOMS doesn't support 3rd party queries and the pull mode is not possible. The recent versions of VOMS are providing a SAML endpoint which can be queried to obtain attributes encoded in the SAML 2 format, but this feature is not used in production yet. As the VO attributes are embedded in the client's certificate, the authentication is tightly coupled with the initial authorization: after the proxy is checked, the security stack checks if the embedded attribute certificate was issued by a recognized VOMS instance, what is equivalent to checking if the user is a member of a supported VO.

User identified by a VOMS-proxy can be authorized and then mapped to a local account. Fine grained authorization can be controlled in a different ways; Argus[6] seems to be the most flexible option. It allows for a simple specification of rules using, among others, the VO and group membership as parameters. The final mapping is performed using LCMAPS or again Argus. Users can be statically or dynamically mapped to the local uids and gids. The later option is using a pool of precreated accounts (or groups) to which users are mapped on demand.

4.2 XtreemOS

XtreemOS[7,8] is a well-established project aiming at providing directly in the Linux operating system Grid features, which are traditionally implemented at the middleware level. Virtual Organizations are one of the main areas where the project is focused on.

XtreemOS developed a custom VO management system which is based on the push model. According to the[9], the information about VO membership and VO related data is carried in a set of proprietary X.509 certificate extensions. Such an *XtreemOS-certificate* is issued on demand to VO users during login phase, by a specially designated central service — Credential Distribution Authority. Subsequently the certificate is used to authenticate and further authorize the user when accessing resources.

Virtual Organizations are created by the users. The VO creator becomes its administrator and is responsible for acceptance of other users who want to join the VO. Resources are also part of the VO, but are offered to the VO by resource administrators[1]. Naturally, the VO administrator can reject or remove VO resources.

A VO-user is mapped to system account using a low level operating system extensions (hooked in PAM and NSS). In general, a VO user has a dynamic and in a sense virtual uid assigned on demand. The XtreemOS exploits the lightweight virtualisation capabilities of the Linux kernel (*cgroups*), to separate VO-bound resources in the operating system. XtreemOS allows for using XACML policies defined within VOs, to manage a fine grained access control. It is possible to use three predefined attributes: role, group and sub-group.

4.3 UNICORE (up to 6.4.x) and UVOS

Virtual Organizations support in UNICORE is implemented in the UNICORE Services Environment (USE). As USE is used as a hosting container for all relevant UNICORE services, all UNICORE services share the same set of VO-related features.

Up to the version 6.4.x (based on the USE 2.0.x) UNICORE was able to use only UNICORE VO Service (UVOS) server as a source of VO membership and attributes information. SAML 2 was used as an encoding format and a query protocol so other SAML attribute authorities such as Shibboleth IdP or SAML VOMS could be considered too. Unfortunately such deployments were barely feasible due to incompatible authentication mechanisms and lack of a common attribute profile. Both push and pull modes for obtaining VO data were supported on the server side, but as there was no support for push mode on the client side only the pull mode was fully useful.

[1] Actually this process is slightly more complex, the details can be found in[9].

Despite of the improvements in the configuration options, the significance of the VO-related data is still minimal in UNICORE. The VO related data is used only for authorization of the access to the resources and even for this — indirectly. To explain this we need to discuss briefly the UNICORE authorization mechanism.

UNICORE authorization is using XACML policies which are very flexible but at the same time extremely complicated and difficult to modify. Therefore the default policy is used most of the time, with some trivial changes. The default policy is using a role-based access control, augmented with the information about the WS-resource owners:

- Some basic, read-only operations are available for all authenticated users.
- User with the *admin role* is always allowed.
- User with the *user role* can invoke several designated operations. Those operations usually create WS-resources which are owned by the creator. For instance, submitting a job is allowed for everybody with the user role. They result in a creation of a private Job Management WS-Resource.
- WS-resource owner is allowed to access the owned resource.

It can be noted that VO membership is not mentioned in the above process. Instead, the USE VO-aware attribute collector checks for the role attribute in the configured VO. If it is found, the authorization role attribute is set accordingly and the VO which assigned the role attribute is not taken into account any longer. Of course the VO membership and all other VO attributes are available in the authorization process, however the default XACML policy ignores this information.

UNICORE VO System (UVOS) server manages the users database allowing to assign them attributes, group into VOs and VO subgroups. UVOS can handle arbitrary number of VOs and attributes. Attributes can be (and usually are) visible only in context of a specific VO or its subgroup. UVOS offers a flexible, internal authorization mechanism which makes the distributed administration of its content possible, as different administrators for each group/VO can be defined.

To sum up, in the current UNICORE versions, the VO information can be only used to bootstrap a generic, VO-unaware authorization process. Configuration of the VO sources is flexible and different (VO-specific) attributes might be used. The VO membership is not used for any other purposes. UVOS offers only a static VO features but with support for distributed administration.

4.4 Other Solutions and Summary

Except of the three systems described above, there was a wide range of VO-related efforts which typically were developed in EU-IST projects. We can enumerate most important here (the VO related parts of the projects are highlighted):

- **BEinGRID** project evaluated a range of business scenarios where grid and SOA adoption was considered. It worked on providing real-life business use cases for the Virtual Organizations. BEinGRID deployed software developed in the GRIA project.

- **GREDIA** project was focused on the negotiation and creation of the ontology based contracts governing Virtual Organizations. High level contracts were deployed in PERMIS acting as authorization engine and VOMS acting as attribute authority.

- **Grid4ALL** was focused on providing VO capabilities to solve everyday problems of families, non-profit organizations or small enterprises in case of health care, collaborative learning etc.

- **SIMDAT** similarly to BEinGRID was investigating industrial use cases of Virtual Organizations.

The above list is by far not complete. Unfortunately a simple research on the results of the aforementioned projects leads to a very sad conclusion that results sustainability is a major problem. GREDIA project is the only one from the above where results are still generally accessible, but some of the third-party repositories are missing and therefore the detailed evaluation is hardly possible. For some others even the web pages do not exist anymore.

Using the classification of the VOs, which was presented in the section 2, we can state that both gLite and UNICORE VOs are static. In the case of gLite/VOMS it is necessary to set up a separate server instance to create a new VO. With UVOS it is easier but still requires privileges which won't be given to the ordinary users. XtreemOS allows for a more dynamic VO life cycle: end-users can create VOs, and automatically become their administrators. The resources still must be added to each VO by resource administrators. While this makes sense in many scenarios, it is not likely that administrators will manually assign resources to hundreds of small VOs coming in and out.

All the three described solutions are using VOs primarily to control authorization. Additionally both XtreemOS and gLite use VO data to set up user environment. The other solutions, for example developed by the GREDIA project, allow for complex relationships modeling, under the VO hood. While appearing to be very attractive, they lack sustainability and are not used in large scale Grid deployments, what brings a lot of questions regarding completeness, stability and usability.

5 The Roadmap towards Advanced VO Support in UNICORE

The current state of the Virtual Organizations support in UNICORE is not perfect. There are two significant issues, which are not addressed by the software stack:

- Static VOs are not convenient in adoption, as for each site providing resources to a VO, each VO user must have a local account assigned (assuming that shared account is not suitable). In another words, the UNICORE is missing the ability to initialize a local user environment automatically, based on the VO information. Additionally the lack of a full support for push mode makes use of many VOs problematic.

- It is very difficult for end-users to cooperate: sharing a workflow, job results, or simply some files, must be performed using third party tools. Putting it other way round, UNICORE is missing the support for dynamic VOs.

The requirement for support of complex VO rules such as VO-level SLAs can be considered as a third point, however that UNICORE should provide support for such in-VO relationships which can be enforced by UNICORE. As we can learn from the other solutions that final product must be really easy to be used and complete, we currently foresee only a minimal effort in this direction, which will be a base for further developments.

The changes needed to improve the static VOs support are quite clear as can be directly derived from the missing features.

The roadmap towards the dynamic VOs support is more complicated. The easiest approach will be to model dynamic, ad-hoc VOs as subgroups of the static VOs. As it is described below, such an approach allows for easier users discovery during the dynamic VO creation and solves a problem of resources provisioning to the dynamic VO. The following sections provide a step by step description of the proposed realization.

5.1 Association of Grid Operations with VOs

Up to now a UNICORE operation invoked directly or indirectly by an end-user is not bound to any particular VO. This must be changed, as without a selected VO it is not possible to use a VO-specific configuration during request processing. A good example is accounting of the requests under a proper VO or choosing a correct local group id (gid) for the user's process.

The VO can be chosen by the user implicitly by using the push mode or by expressing a preferred VO in metadata of the request. Both options were implemented in USE 2.1.x, used by the UNICORE servers of release 6.5.x. In case of the pull mode, the server uses the user's preferred VO and if it is not present, administrator can configure a prioritized list of VOs.

What is worth noticing, the push mode is also implemented in the UNICORE Command Line Client (UCC) version 6.5.0. Therefore currently only the URC client is missing the push mode support.

Impact: ability to apply arbitrary configurations/restrictions/rules defined per VO.

5.2 Sharing Physical and Virtual Grid Resources

The requirement necessary to support dynamic Virtual Organizations, is the ability to share resources *already accessible to the Grid users*. In the UNICORE case we can *theoretically* consider sharing of several resources, enumerated below. However it should be taken into account that one of the base UNICORE principles is the non-intrusiveness and seamless integration with unmodified local systems. Therefore the resource sharing options should be evaluated on both Grid and operating system levels (where applicable). In the following list we are providing the relevant Grid and operating system resources.

1. Files accessible via a Storage Management Service. Grid resource: SMS, OS resources: files and directories.
2. Running job management (in practice stopping/aborting). Grid resource: JMS, OS resource: process.
3. Submitted job data (both input and output files and job's description). Grid resources: SMS and JMS, OS resources: files and directories.

4. Submitted workflows, i.e. the workflow description and its atomic jobs. Grid resources: Workflow, Workflow Assignments and all of the job's data resources, OS resources: the same as in case of job data.

5. Management of file transfers. Grid resource: FTS, OS resources: network connection(s).

Sharing the ability to abort a job is of minimal importance to end-users and at the same time is very problematic from the operating system perspective so we have decided to ignore this option. Management of the ongoing file transfers can be also considered as of the low importance, as the most of them is ephemeral, and those transfers which live longer are managed by the middleware rather than end-users. The remaining features should be supported.

To enable sharing on the Grid layer we propose to introduce the following changes in the UNICORE Services Environment:

- Introduce a notion of parental relationships between Grid resources. As it can be observed many of the sharing cases are recursive: sharing of a workflow data is mostly the equivalent to sharing all its child jobs. A universal way of establishing whether a Grid resource has a parent and children will allow for implementing the recursive sharing easily.
- Classify operations on all Grid resources as read-only or write/modify. This will allow users to share resources in an easy to understand "read-only" or "full" modes.
- For each resource add a list of VOs, which members are allowed to access it (or in other words, a list of VOs with which the resource is shared). For each of the VOs on the list, subsequent settings regarding read only or full sharing mode will be available.
- Provide a Web Service method for controlling the above list by the resource owner.
- Modify the default UNICORE XACML policy, so it will take into account the VO membership of the user and the allowed VOs of the resource being accessed.

Sharing on the operating system level will be required only in a case of files and directories. The recent UNICORE versions already provide ability to control (also in recursive way) the group ownership of the files and files ACLs. The first, feature can be used to enable sharing of file in a single VO, while ACLs allow for sharing with many VOs. The problem with ACLs is that they are not supported on all systems, so we plan to support both modes.

Impact: support for Grid resources sharing, reflected also on the operating system level.

5.3 Dynamic Attributes and Pool Accounts

Currently either a VO or each site has to manually map each user to a local account. This alone makes the deployment of static VOs difficult. However this is also an obstacle in the case of dynamic VOs, as members of such VOs should share a common local group (gid). Assignment of a gid to a dynamic VO must be easily possible as otherwise creation of dynamic VOs would require an intervention of each participating site administrator, blocking the whole process.

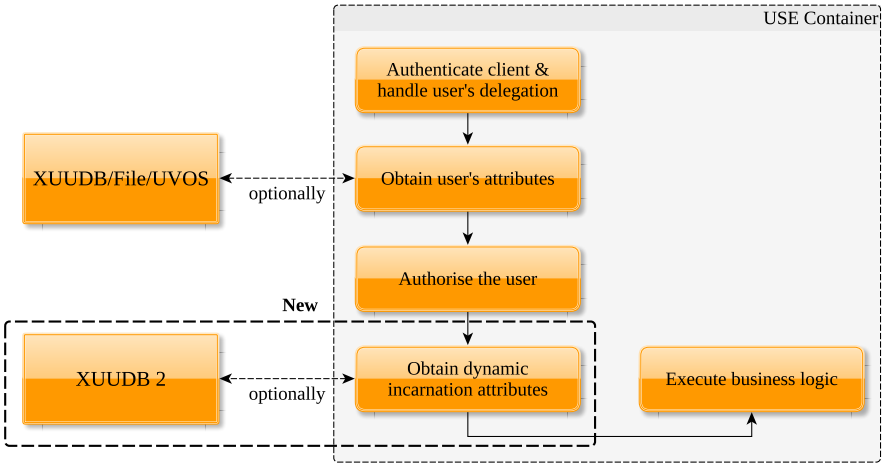

Figure 2. Planned changes in the overall security pipeline of the UNICORE Services Environment.

Example scenarios:

- All users from the VO */vo.plgrid.pl/dynamic/???* should get a gid from a pool *grp1–grp100*.
- All users having the role *mathlab* should get a supplementary gid *mathlabuser*.
- All users with the role *user* and the VO `/vo1` should get a uid from a pool *vo1-user1–vo1-user100*.

The above settings are already managed in the UNICORE with a use of special *incarnation attributes*, which are defined in the static attribute sources as files, XUUDB or UVOS. However in this case we need a new attribute source because the dynamic mapping must be performed only for already authorized users and should not reserve a local uids or gids for users which are banned by the authorization stack. Additionally as an input new attribute source will need the final information on effective static attributes (as VO membership or VO role). Therefore a new mechanism must be integrated after authorization step in the UNICORE Services Environment what is depicted in the figure 2.

As a reference implementation we propose an extended XUUDB service. Extending an existing service will not increase administration effort a lot, and will minimize development effort.

The extended XUUDB service will feature a possibility to assign uids, gids and supplementary gids, by creating logical expressions using static user's attributes as an input. The local identifiers will be assigned from precreated pools, retrieved from a script defined by an administrator or simply fixed.

Impact: easy deployment of static VOs (automatic uids assignment) and dynamic VOs (automatic gids assignment); possibility to define local supplementary gids issued per-VO.

5.4 Required Features of the VO Service

Implementation of the dynamic VO features will most probably take place in the new major release of the UVOS system. UVOS already offers possibility to define subgroups. What needs to be done is a proper separation of privileges, allowing for UVOS-administrator controlled creation of big, parent VOs and user-driven creation of subgroups in allowed places. Similarly, attributes defined in the subgroups should not be propagated to the parent groups.

Finally the VO service must offer capabilities to discover (and register) resources which are assigned to each VO. For this purpose we propose to implement in the UVOS the UNICORE Registry interface (which is strictly based on the WSRF Service Group standard). Each of the VOs would feature one instance of the Registry. Such approach will allow to reuse the code to manage registered services in UVOS, the code to register a resource on a target site and finally simplify a client resource discovery, as support for UNICORE Registry is available there.

Impact: complete management of the VO contents and discovery of its members and resources.

5.5 Client Support

The fundamental principle behind client support of dynamic VOs is obvious: it must be easy.

We foresee the following set of functions:

- VO management: creating a VO; inviting users to a VO; managing invitations; managing VO members.
- Resources management: adding/removing a resource to/from a VO in a desired mode.
- Resources discovery: listing resources accessible from a VO.

The first part of operations will be using the VO service to obtain potential users and manage them. The second part should be integrated with the existing resource management features. For instance in the UNICORE Rich Client the "share it" operation should be available from the resource context menu of the Grid Browser. The last part will require integration of the per VO registry, as provided by the VO service.

It can be noted that none of the operations is complicated, and all can have an intuitive interface, especially in case of a URC plugin.

Impact: possibility to manage dynamic VOs by the end-users.

6 Summary and Outlook

The planned extensions of the UNICORE stack are vast and complicated. However they will bring a significant added-value, and will affect the existing functionality minimally. The built-in features to easily cooperate and share work results should greatly increase productivity of UNICORE users.

It can be noted that the new features can be either disabled, or used together with current solutions in a mixed mode and so the backwards compatibility will be preserved.

For instance selected users can have the fixed account mappings, while the rest has the dynamic mappings.

The planned improvements will also allow for starting a work on other features. Assignment of requests to the VOs creates a foundation for deploying in the UNICORE a per-VO billing system and controlling the VO-bound licenses usage. The new VO service can be further extended to provide more complex per-VO authorization rules, assuming that there will be a need for such. Finally, if UNICORE is extended, we can enrich the list of the UNICORE incarnation attributes to provide a VO-scoped control of the new feature. For example we can add new incarnation attributes governing which Virtual Machine (VM) images (and with what physical resources) are allowed for a VO, if an on-demand VM deployment is made possible in UNICORE.

References

1. Bernd Schuller, *General Introduction to UNICORE*, http://www.unicore.eu/documentation/presentations/unicore6/files/General_Introduction_to_UNICORE.pdf (25.06.2012).
2. N. Ragouzis et al., *Security Assertion Markup Language (SAML) V2.0 Technical Overview*, OASIS Committee Draft, 2008, http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf (25.06.2012).
3. OASIS XACML Committee web page http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml (25.06.2012).
4. I. Foster, C. Kesselman, S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal of High Performance Computing Applications, v.15, 3, pp. 200–222, 2001.
5. R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, K. Lorentey, F. Spataro, *From gridmap-file to VOMS: managing authorization in a Grid environment*, Future Generation Computer Systems, vol. 21, 4, pp. 549–558, 2005.
6. Argus Authorization Service project website https://twiki.cern.ch/twiki/bin/view/EGEE/AuthorizationFramework (25.06.2012).
7. D. Laforenza, *XtreemOS: Towards a Grid-Enabled Linux-Based Operating System*, BNCOD, pp. 241–243, 2008.
8. XtreemOS project website http://www.xtreemos.eu (25.06.2012).
9. B. Aziz (ed.), *Fourth Specification, Design and Architecture of the Security and VO Management Services (D3.5.13)*, XtreemOS project deliverable, 2009, http://www.xtreemos.eu/project/publications/project-deliverables/d3-5-13.pdf (25.06.2012).

File Transfer in UNICORE: State of the Art

Bernd Schuller, Michael Rambadt, and Björn Hagemeier

Jülich Supercomputing Centre,
Research Centre Jülich, Jülich, Germany
E-mail: {b.schuller,b.hagemeier,m.rambadt}@fz-juelich.de

The UNICORE middleware is increasingly used for data-intensive applications where large data volumes need to be transferred reliably at adequate data rates.

This paper presents the state of the art in UNICORE with respect to file transfer, and takes an in-depth look at the existing functionalities, the performance (transfer rates, reliability) and last-not-least the current limitations of the existing interfaces and available implementations.

The recently introduced UFTP file transfer has been deployed and thoroughly tested in realistic environments. Here, we look at the performance of UFTP in two settings: a test installation within the PRACE infrastructure, and a 100GBit/s testbed operated by the TU Dresden and TU Freiberg. The latter study shows the potential of UFTP in a network of the highest performance.

Furthermore, we present work that has been done to integrate the widely used GridFTP transfer into UNICORE, motivated by interoperability requirements in the U.S. XSEDE infrastructure as well as other European software such as dCache.

Finally, we review the reliability aspects of the UNICORE server-to-server file transfer and present some possible improvements.

1 Introduction

UNICORE has been traditionally used to provide access to high-performance computing resources, where the associated data management functionality can be rather limited. It was enough to be able to upload input files for the job and download result files. However, the scenarios where UNICORE is used are changing rapidly towards more data-oriented usage. In contrast to other Grid systems such as gLite, UNICORE does not have dedicated components for data management (i. e. storage elements), rather the UNICORE/X component provides a tightly integrated set of services that offer file transfer functionality together with job management and other functions.

In this paper we focus on file transfer, presenting several recent results and developments, as well as work in progress and future ideas on how to improve the file transfer functionalities in UNICORE.

The paper is organised as follows: the next section presents the state of the art in UNICORE (as of version 6.5.0). Section 3 describes performance tests of the recently introduced UFTP file transfer protocol. In Section 4 we take a look at GridFTP, while Section 5 presents recent work on reliability and possible future extensions and enhancements.

2 File Transfer in UNICORE - an Overview

Data and/or file transfer is an integral part of any distributed computing system, and UNICORE is no exception. Currently two mechanisms exist that have to be distinguished because they have slightly different features.

2.1 File Transfer on the Web Service Level

On the one hand, a UNICORE client can initiate a file transfer by invoking the `Export` (or `Import`) method on the Storage Management Service (SMS) instance that provides access to the source (target) storage resource. This will download (or upload) data onto (or from) the local file system to the remote file system. Data transfers can be monitored via the UNICORE's web service interfaces. As an important special case of this scenario, server-to-server file transfers are possible, where one UNICORE server acts as client. These transfers are set up by invoking the `SendFile` or `ReceiveFile` methods on the Storage management service instance.

For performance reasons, all data transfer should go directly from the source to the target filesystem without any intermediate socket connections. Taking the UNICORE architecture into account, this means that the server-side process that sends or accepts data should run directly on the TSI node.

Due to the integration into UNICORE clients, protocol handlers need to be written in Java. Non-Java solutions are in principle possible (for example using invocation of external tools, or even integration via the Java Native Interface), but these tend to be clumsy and lack features such as detailed monitoring.

Multiple file transfer protocols are supported. In practice, the *BFT* mechanism is the default mechanism supported by all clients and servers. This is based directly on the HTTPS connection between client and UNICORE/X server. The BFT transfer achieves performance of some MB/sec, but this is not enough for high-volume data transfer. One limitation for performance is the fact that multiple "hops" are required: client to gateway, gateway to UNICORE/X and finally UNICORE/X to TSI. Of course the data is streamed through (not cached at each hop), but still each hop approximately halves the performance.

Another important protocol from an interoperability perspective is the ByteIO transfer mechanism, which is standardised by the OGF[1], and implemented by other middleware such as Genesis II[2]. The performance of the UNICORE implementation is fairly bad, reaching only several 100 KB/sec. This is due to the fact that UNICORE only supports the simple way of transmitting the data embedded into the web service SOAP request / response messages. The superior "MTOM" mechanism of attaching binary data is not (yet) supported. Using MTOM the performance should be comparable to BFT.

Currently the most promising file transfer mechanism in UNICORE is UFTP, which has been described in great detail elsewhere[3]. UFTP is based on plain TCP, and (as we will show later) can saturate even high-performance network lines. The most interesting feature of UFTP is that it is modelled after the venerable FTP[4] protocol, and (as FTP) requires only a single open firewall port for firewall transversal, negotiating the required data connections dynamically. Each client-to-server UFTP transfer involves a control connection, which can be used to add additional features to UFTP as will be shown in Section 5.

2.2 Data Staging: File Transfer on the XNJS Level

On the other hand, data can be staged into or out of a job's working directory. For data staging, UNICORE supports additional protocols, such as mailto, scp or GridFTP. Data staging is initiated and monitored by the XNJS component that takes care of job management within a UNICORE/X server. Note that all protocols supported for file transfer also work for data staging, but not vice versa.

Summarizing, there are two levels of integration of any particular file transfer mechanism, the *full* integration on a web service level, or the lesser XNJS-level. In terms of performance, UFTP and GridFTP are to be preferred, with UFTP to be preferred due to its superior security. However GridFTP plays a huge role in practice, and UNICORE would do well to provide a useable, interoperable solution for accessing GridFTP.

3 UFTP Performance Tests

Since its introduction in the UNICORE 6.4.0 release, UFTP is increasingly being deployed and used. This section describes tests that have been done on resources within the PRACE network as well as performance tests done on a 100GBit/sec testbed.

3.1 PRACE Tests

The PRACE Research Infrastructure[5] can be seen as the successor to the DEISA project which connected several European HPC centres using dedicated network lines. Typically, the HPC resources are available via several network interfaces. To deploy and use UFTP in such an environment, client support for selecting an appropriate network interface had to be added to UFTP.

A small testbed consisting of HPC systems in Jülich, CINECA and SARA was used for initial tests. As the sites and network are rather heterogeneous, the performance depends on which sites and which network interfaces are used. On the 1Gbit/s link between Jülich and CINECA, a transfer rate of 70MB/sec was achieved. Similar rates were measured on the 10GBit/s link between SARA and Jülich, showing that the file systems rather than the network bandwidth limit the read/write rates. For comparison, a Jülich internal test was done on the 10GBit/s link between the JUGENE (GPFS file system) and JUROPA (Lustre) systems. Here up to 250 MB/sec were measured.

Obviously, these results are very encouraging, showing that UFTP is a very viable transfer solution. For the first time, UNICORE has a native file transfer solution that is competitive for large files.

3.2 UFTP in a 100GBit/s Testbed

We were fortunate to have the opportunity to test UFTP in a network of the highest performance. A project involving TU Dresden and T-Systems SfR as well as several industry partners like Alcatel-Lucent[6].

For our purposes, the testbed can be described as follows. Two clusters (one located in Dresden, the other in Freiberg) are connected by a 100Gbit/s line. Each cluster consists of a head node and several cluster nodes, connected by a 10Gbit/s line. Thus, between any two nodes, 10Gbit/s can be achieved, while the aggregated bandwidth (using multiple nodes) can be as high as 100Gbit/s.

We tested UFTP in various scenarios involving data sizes up to 10GB. To remove the limitations of the file systems, data was generally read from /dev/zero and written to /dev/null. Due to the high quality of the network connection, best results were achieved using a single UFTP data stream. To simplify the testing, UFTP was used without the involvement of a UNICORE server or client.

- Single server, single client: here UFTP achieved up to 1.2 GB/sec, corresponding to 98% of the line rate
- Multiple server-client pairs: several UFTP servers and clients were set up and started, trying to achieve parallel data transfers between all of the pairs. Again, up to 98% of the line rate (i. e. 12GB/sec for 11 client-server pairs) were measured.

These results show that UFTP can utilize a network of the highest performance and achieve very good transfer rates. Of course, this is mostly due to the quality of the TCP stack, and its usage in the Java virtual machine, which is almost the same as in the native C environment.

4 Improving the GridFTP Integration

UNICORE has been able to use GridFTP for data staging already for a long time. The integration is rather crude, however: for each data staging operation involving GridFTP, UNICORE simply invokes the `globus-url-copy` tool which has to be installed on the TSI node. The required proxy certificate is created on the UNICORE client side, and uploaded to the server.

While this is already sufficient for many cases, there are open issues:

- It should be possible to use existing credentials (e.g. VOMS attribute certificates, or MyProxy certificates) instead of creating a proxy from the end-user certificate
- It is not possible to interact directly from a UNICORE client with a GridFTP server. This requires integration on a web service level.

While solving the first issue is fairly simple, the integration of GridFTP on a web service level is more complicated. Java APIs and libraries for GridFTP and other Globus tools are provided by the jGlobus libraries which are part of the Globus Java COG kit[7]. It was possible to patch the jGlobus library in order to remove all conflicts with the UNICORE libraries. The patched jGlobus was tested in the HiLA API.

As using jGlobus seems to be doable, intriguing possibilities arise for a much better integration with GridFTP: a native UNICORE client module to read/write GridFTP files could be implemented. This would allow direct access of a GridFTP server from a UNICORE client as well as data staging without requiring a pre-installed `globus-url-copy`. A useful and flexible solution for providing the required credentials remains to be developed.

5 Reliability Challenges and other Work in Progress

A critical feature of transfer of large data volumes is reliability. It is not acceptable that a transfer needs to be restarted from scratch in case of errors. This is true for both data staging during job processing and data upload/download by clients. UNICORE currently has built-in support for re-starting failed transfers, where data was already partially moved. Of course such a feature also depends on the protocol that is used. While some protocols (like HTTP) in principle support partial reads or reading from a given offset, some do not.

To overcome this issue, work on a truly reliable server-to-server transfer was started. This can use any protocol provided it allows to read byte ranges. This is true at least for HTTP and ByteIO, and will be implemented in the next release of UFTP.

One weakness of UFTP (and UNICORE file transfers generally) is that they are inefficient for large numbers of (smaller) files. This is due to the overhead having to "create" a file transfer resource for each file before being able to send data. Since UFTP is using a control channel as well as data channels (like FTP itself), an obvious solution to this problem is to transfer multiple files using a single UFTP session. This will remove the overhead of having to initiate a UFTP connection for each file, and should greatly improve performance. This "session" functionality will be released with UFTP 2.0 expected in late 2012.

Thinking further, it is attractive to speculate on implementing a synchronization protocol directly in UFTP. This would allow to "sync" a local with a remote directory using UFTP, allowing advanced use cases such as storage replication. To make synchronization efficient, an algorithm such as the well-known *rsync* algorithm[8] suggests itself.

6 Summary and Outlook

Data management and transfer continues to be a major driver for future enhancements of UNICORE. Rather that adding to the already very strong compute functionality, improving the file transfer functionalities is likely to provide much more added value for end users.

As this paper has tried to show, the UFTP file transfer library is one area where valuable new features can be added: UFTP is reliable, very fast and easy to use, and due to its relationship to FTP it can be extended with high-level functionality like a session mode, partial downloads, or even rsync-like synchronization capabilities. Since the UFTP library itself is small, it can potentially be used in contexts outside of the full UNICORE scenario as well.

A second major area is GridFTP: despite the differences in the security models, and the "traditional" competition between UNICORE and Globus, a more useful and feature-rich integration of GridFTP has become more than an interoperability exercise. Many users have the real need to integrate data from GridFTP servers into the UNICORE environment.

Last not least, there are many things to do to make the UNICORE experience more rewarding for end-users. Having well-performing, reliable, scalable file transfer mechanisms is a key ingredient, and should receive more development effort.

We have shown some of the ongoing development activity in these areas, and are confident that future UNICORE versions will be even better suited for data-oriented applications.

Acknowledgements

The authors wish to thank Andy Georgi and Stefan Höhlig from TU Dresden for generously providing access to the 100Gbit/s testbed.

References

1. M. Morgan, *ByteIO specification 1.0*, Tech. Rep. GFD-R-P 087, Global Grid Forum, 2006.
2. Genesis II website `http://genesis2.virginia.edu/wiki/`.
3. B. Schuller and T. Pohlmann, *Uftp: High-performance data transfer for unicore*, Proceedings of the 2011 UNICORE Summit, IAS Series Volume 9, pp. 135–142, 2011.
4. J. Postel and J. Reynolds, *File Transfer Protocol*, RFC 959 (Standard), Oct. 1985.
5. PRACE Research Infrastructure `http://www.prace-project.eu`.
6. Press release on the 100Gbit/s testbed `http://tu-dresden.de/aktuelles/newsarchiv/2010/6/gigabite/newsarticle_view?set_language=en`.
7. CoG jGlobus `http://dev.globus.org/wiki/CoG_jglobus`.
8. A. Tridgell and P. Mackerras, *The rsync algorithm*, `http://cs.anu.edu.au/techreports/1996/TR-CS-96-05.pdf`.

Providing Grid Data Access on SRM and LFC to UNICORE

Christian Löschen and Ralph Müller-Pfefferkorn

Technische Universität Dresden,
Center of Information Services and
High Performance Computing (ZIH)
E-mail: christian.loeschen@tu-dresden.de
E-mail: ralph.mueller-pfefferkorn@tu-dresden.de

Within the EMI project, the European Grid Middlewares ARC, gLite, UNICORE and dCache will be improved on interoperability and made more integrated. These efforts include the communication between them, where the access to each others services is made easier and more comprehensive.
In the Grid, the data are stored on storage elements of different sizes. Most of them are managed by Storage Resource Managers (SRM) and are accessible via the SRM protocol. Until now, UNICORE doesn't provide a way to use the SRM protocol, thus there is no access to SRM managed storages. To improve this situation for the UNICORE users, one task of EMI is to integrate access to these common Grid storage systems into UNICORE. This will be achieved by implementing an interface for client access to SRM storages.
Additionally to SRM-based management, many files in the Grid are indexed in file catalogues, like the LHC File Catalogue (LFC), which is a part of gLite. There exist file catalogues that are VO-based, as well as for smaller user communities. If UNICORE supports access to the LFC, users may have a structured global view of their data, independent of the actual storage element where the data are stored, and can access these data. Therefore, the second Grid storage integration topic of EMI is the provision of UNICORE access to the LFC file catalogue.
This paper presents which techniques are used to reach these goals, explains details of the UNICORE model of storage handling and how it is used to implement client access to SRM storages and LFC. Typical file operations on SRM and LFC, necessary in the stage-in or stage-out processes in UNICORE jobs are described, as well as the current implementation status, handicaps and future plans.
This work was funded by the EMI project under European Commission Grant Agreement INFSO-RI-261611.

1 Introduction

1.1 The European Middleware Initiative EMI and UNICORE Data Management

The goal of the European Middleware Initiative is to deliver a consolidated set of middleware products based on the four major European middleware providers:

- **ARC** (Advanced Resource Connector) is a lightweight middleware developed by NorduGrid.
- **gLite** provides a framework for building Grid applications, started as part of the EGEE project
- **UNICORE** started as a German Grid middleware system with workflow support, developed at Jülich Supercomputing Centre

- **dCache**, developed by DESY, is more a Grid storage element than a Grid middleware, that stores huge amounts of data under a single virtual filesystem. The dCache storage element is widely deployed within the WLCG.

These middlewares are developed in collaboration within EMI, under the major aspects of harmonization, integration, standardization and interoperability to establish a sustainable model to support, harmonise and evolve the middlewares. It is deployed in the European Grid Initiative (EGI) as part of the Unified Middleware Distribution (UMD[2]). The focus within EMI is on the areas of compute, security, infrastructure and data.[1]
Currently, UNICORE provides access to external storages for iRODS, Hadoop, and the filesystem, whereof iRODS is the only native Grid storage. A benefit for UNICORE users would be to extend the variety of available storages. Since many Grid storage elements support access by the SRM protocol, and many files available via SRM are referenced by the LFC, which is widely deployed within the WLCG, offering those possibilities to UNICORE users seems quite useful. Therefore the integration of LFC- and SRM-based storage access has been accepted as a task of the EMI data area integration work. SRM-based storages are provided by the EMI partners dCache and gLite.

1.2 Data Management in the Grid

In the Grid data management there is a huge challenge: here exists lots of data, like data from experiments, calculated results, temporary files or user and job files, that are owned by many users. Those users are organized in several communities and Virtual Organizations (VO) and the data is distributed over many storage resources world-wide and accessed by different protocols: partly standard, partly storage element-specific protocols. This scenario implies basic challenges like transparent access for the user, access control mechanisms for authorization and authentication, and many more on security, scalability, reliability and transaction and storage management.
Data stored in gLite-based Grids are mostly structured in a hierarchical manner (see Figure 1). On the top there are Grid file catalogues that only store logical filenames (LFN) that point to existing replicas of files distributed on several Grid Storage Elements (SE) by Storage URLs (SURL). The storage elements beneath are managed by Storage Resource Managers (SRM), that control the local storages. They resolve Storage URLs to the actual file location on harddisks and to the Transfer URLs (TURL). These Transfer URLs may be finally used to physically access a file. The Storage URL does not need to have any relation to the Transfer URL and is only important to the SRM, which prepares the file for transfer and generates the Transfer URL.[3]

2 Technical Principles

2.1 The gLite File Catalogue LFC

LFC is the LCG File Catalogue and a part of gLite. The file catalogue offers a hierarchical view of files to users, with a UNIX-like client interface, and provides logical to physical mappings for file identifiers. It can be used locally, or as a global catalogue holding details of files stored on Grid storage elements. Using a file catalogue, a community is able to

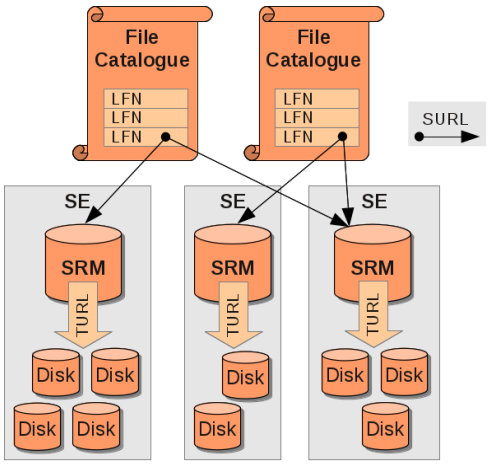

Figure 1. Hierarchical Grid Data Management

keep track of all of its files distributed on various storage elements in the Grid.
The LFC stores all information in logical filenames, which provide a unique identifier for every file, information on physical replicas and system metadata like owner and access rights.[3]

2.2 Storage Resource Management - The SRM Protocol

The Storage Resource Manager is a Webservice Protocol for Grid storage management that is implemented in several Storage Resource Managers. Files in the SRM are referenced by Storage URLs.
Well known Storage Resource Managers are: dCache, that supports tape backends; the gLite Disk Pool Manager (DPM) without tape support, that pools individual disk servers together to a single namespace; StoRM, that acts as a SRM layer on top of a POSIX filesystem, and BeStMan, the SRM reference implementation of the Lawrence Berkley National Laboratory. Further there are the CERN-owned storage elements CASTOR with a tape backend and EOS, the CERN disk only storage element.[3]
Features of the SRM are Space Reservation, Space Tokens, Retention Policy and Access Latency assignment, further Storage Classes, bringOnline- and prepareToGet requests and protocol negotiation of the file transfer protocol, since the actual file transfer is not performed by the SRM but by some file transfer protocol like GridFTP.
SRM supports many POSIX filesystem operations as well as put and get operations, which are always delegated to be done by actual file transfer protocols.

2.3 The UNICORE Storage Interface SMS: Storage Management Service

The Storage Management Service (SMS) is the UNICORE interface to storage resources and a component of the UNICORE/X server. The UNICORE/X server is a central component of UNICORE, and located in the UNICORE service layer[4]. The storages available for

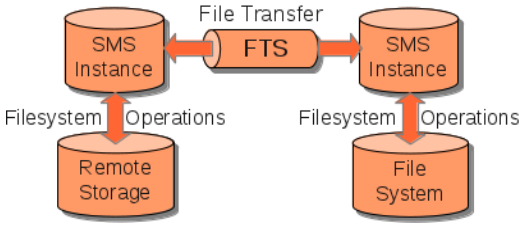

Figure 2. UNICORE File Handling

UNICORE, which are local filesystems on UNICORE servers, remote storage elements, and the storages on the target systems, are managed by the Storage Management Service. It provides an abstract, homogenous filesystem-like view of a storage resource to the user and takes care of all common filesystem operations, like creation, deletion, moving of files and directories or directory listings and metadata. It initializes file transfers for put and get operations, like import, export and stage-in and stage-out of files on the local machine. There are currently only a few SMS implementations to access external storages: iRODS, Hadoop, and the filesystem.

UNICORE distinguishes between the storage access itself and the actual file transfer within UNICORE (see Figure 2). While the SMS is responsible for storage access, reading and writing files, the file transfer within UNICORE is done by the File Transfer Service (FTS), which can handle several protocols.
Typical scenarios to invoke an SMS are the stage-in and stage-out processes before and after job processing, or user-driven data management on UNICORE storages.

3 Enabling SRM and LFC Access for UNICORE

3.1 Integration Tasks

3.1.1 LFC Access

Both, the LFC and the SRM accesses are implemented as UNICORE SMS components. For the implementation of the LFC SMS the gLite Data Management Library developed by the cloud service provider maatG has been used[1]. This Java-based library provides client functions to access the LFC and the gLite DPM and DPNS (DPM Name Server) services. Although the SMS design as a storage access component also provides the possibility to write to storages, writing to the LFC, or in general creating new entries in the LFC, is not possible. This is due to the necessity of two URLs for this operation: the first one indicates the LFN of the LFC entry to be created or appended to, and the second one, the Storage URL, points to the location of the referenced replica. Unfortunately the UNICORE SMS file writing function supports only one URL as the destination to write to. This issue is due to the contrast of the UNICORE SMS as an storage interface and the LFC, which is rather a metadata service than a file storage.

[1] https://project-mgt.maatg.fr/projects/glite-data-management/

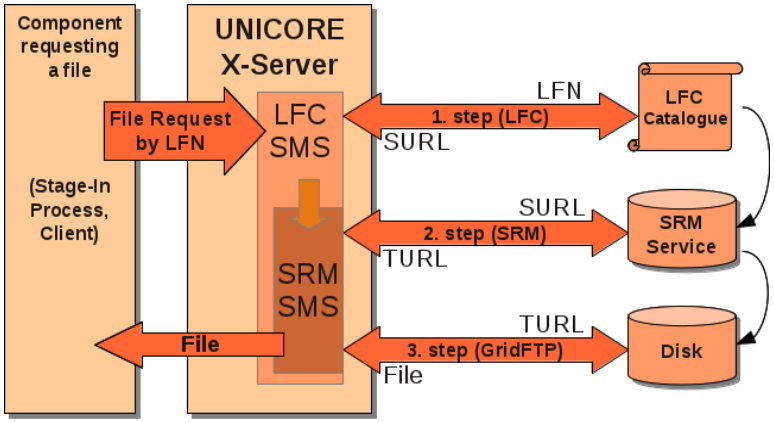

Figure 3. View of the LFC Storage Management Service on Reading a File

The reverse way, 'reading' a 'file' from the LFC is implemented in the following way: When a file is requested by the user or by the stage-in process of a job, the first step is to request the LFC to resolve the LFN to a Storage URL (see Figure 3). Now an SRM SMS is instantiated and the Storage URL is delegated to it. In the second step the SRM SMS performs an *srm-get-request* for the Storage URL to the Storage Resource Manager of the storage element. Currently, only files stored in storage elements providing SRM access are supported with LFC SMS. The SRM returns a Transfer URL. At last, the Transfer URL is used to transfer the file from the storage element using GridFTP. As one can see, the LFC SMS depends on the SRM SMS for the actual file access.

The configuration of the LFC SMS is done by the common UNICORE way within the *uas.config* file. Additionally to the usual parameters, the hostname and the port of the LFC service have to be declared using the *sms.lfc.host* and *sms.lfc.host* parameters. It is also possible, to use environment variables to instantiate the SMS.

As usual on gLite-based grid services, a valid X.509 grid proxy certificate is necessary to access the LFC using the SMS. Therefore the user needs to serve his or her proxy certificate to the UNICORE/X server, using the UNICORE client. The SMS will request the provided certificate by the client connection, and use it to authenticate the user on the LFC. Without a valid proxy certificate, the SMS will refuse to work.

A feature that additionally may be added, is replica selection on requesting an LFN to transfer. If there are multiple copies of a file stored on different storage elements, these replicas may show different access latencies. For example, one replica on a storage element might be stored on disk, and a second one is stored on a tape backend of another storage element. To avoid longer idle time instead of waiting for the latter storage element for staging the file from tape, the replica, which is stored on disk and assumed to be served faster, may be selected for transfer.

3.1.2 SRM Storage Access

For the SRM integration only a small subset of all SRM client functions has been implemented due to the limited functions of the UNICORE SMS interface. The SMS can issue

the common filesystem commands to browse directories, get file listings and metadata, and put or get files. But a lot of SRM functions are not available, like requests for space reservation or bring-online requests.
For the implementation the dCache SRM client library is used, which provides the whole set of SRM client functions.
The SRM SMS is configured similar to the LFC SMS in the *uas.config* file, specifying the additional parameters *sms.srm.host* and *sms.srm.port*. A valid X.509 grid proxy certificate is required, as well.

3.2 Problems and proposed solutions on SRM and LFC integration

The implemented LFC and SRM SMS modules worked well within the module tests. But when trying to integrate the components into the UNICORE/X software stack, it was figured out, that there were a lot of conflicts in the Java dependencies used by both components and the UNICORE dependencies, like different versions of the Bouncy Castle API, or the customized Globus library used by dCache.
In order to solve this problem, two solutions are currently considered: implementing the required SRM client functions using the UNICORE webservice stack, or adapting the dCache SRM client library to use the libraries provided by UNICORE. Both options seem quite complex. The SRM uses an outdated RPC/encoded style of web service, which makes it hard to implement. Forcing the SRM client library to use the UNICORE libraries is expected to turn out number of broken functions, that have to be fixed. A possible solution for the file transfer could be to use the GridFTP functions currently developed within the HiLA.
Another problem is using of X.509 certificates required by LFC and SRM. The user's proxy certificate is only available via the client connection when the user is connected and calls the SMS directly. When the user disconnects and a job tries to get a file from an LFC or SRM storage on behalf of the user, the proxy certificate isn't available anymore. But this is by far the most common use case, instead of loading all files the user.
To solve this issue provisionary, an existing workaround has been used, which was also already used by the existing GridFTP implementation of the UNICORE/X server. Therefore the users proxy certificate, provided by the client connection, is stored on the UNICORE/X server, and read when necessary. But this is a rather imperfect solution. In the medium-term it is planned to switch to the EMI Security Token Service (STS), which is still in development. The STS is designed to transform security tokens from one format into another, like SAML assertions into X.509 certificates and vice versa[5]. Using the STS it is possible to convert the SAML-assertions used by UNICORE into X.509 proxy certificates required by the LFC and SRM access.

4 Conclusion

The UNICORE integration of LFC/SRM access is on a good way to attach the LFC and SRM storages to UNICORE and to establish more interoperability between the UNICORE middleware and other EMI storage elements. As a result, the users within UNICORE work with their files stored on other common Grid storages. Unfortunately is was delayed to make it available already in the EMI 2 release as planned.

Acknowledgements

We would like to thank to Krzysztof Benedyczak at the ICM, Warsaw University, for his valuable comments on the integration tasks, esp. for his remarks on the EMI STS. Further, many thanks to Bernd Schuller at the Research Centre Jülich for providing deep insights into UNICORE and his support on many implementation details.

Abbreviations

EGI European Grid Initiative
EMI European Middleware Initiative
FTS File Transfer Service
HiLA UNICORE High Level API
LFC LHC File Catalogue
LFN Logical File Name
SAML Security Assertion Markup Language
SE Storage Element
SMS Storage Management Service
SRM Storage Resource Manager
STS Security Token Service
SURL Storage URL
TURL Transfer URL
URL Uniform Resource Locator
VO Virtual Organization

References

1. European Middleware Initiative `http://www.eu-emi.eu/middleware`, accessed June 2012.
2. EGI Software Repository `http://repository.egi.eu/category/umd_releases/`, accessed June 2012.
3. Graeme A Stewart, David Cameron, Greig A Cowan and Gavin McCance, *Storage and Data Management in EGEE*, ACSW '07 Proceedings of the fifth Australasian symposium on ACSW frontiers, Volume 68.
4. Achim Streit, Piotr Bala, Alexander Beck-Ratzka et al., *UNICORE 6 - Recent and Future Advancements*, Berichte des Forschungszentrum Jülich, Jülich Supercomputing Centre, 2010.
5. The European Middleware Initiative to support Security Token Services `http://www.eu-emi.eu/further-readings/-/asset_publisher/7AeX/content/security-token-services`, accessed June 2012.

Uniform Distributed Storage View for the UNICORE Rich Client

Andrzej Dembowski[1], **Rafał Kluszczyński**[2], **Piotr Bała**[1,2]

[1] Faculty of Mathematics and Computer Science
Nicolaus Copernicus University, Toruń, Poland

[2] Interdisciplinary Centre for Mathematical and Computational Modelling,
University of Warsaw, Warsaw, Poland
E-mail: {fred, klusi, bala}@mat.umk.pl

In this paper we present a Uniform Distributed Storage View plugin for the UNICORE Rich Client. The plugin provides solution for exposing storage resources distributed between different UNICORE sites as a single logical space. The uniform view can be also used during job preparation allowing users an easy access to the files and directories. Such feature is highly desired by end-users, allowing them to forget about an actual location of their files and to consider grid storage as a single file system.

1 Introduction

The experience with the grid deployment shows that storage is similarly important as compute resources. Moreover, recent developments in the cloud technology allow users to access distributed storage in easy and efficient way. Users get access to the different cloud storages such as Dropbox[1], Google Drive[2] or Globus Online[3]. Such solutions differs in technology but for the user they are visible as folders on the desktop or as a simple web application allowing for easy (often drag and drop) upload and download of the files.

Easy configuration and lack of the access barrier makes such solutions very popular even for non-experienced users. Utilization of the cloud technology in the everyday applications and devices such as editors or tablets provides cloud storage additional popularity despite limited security or lack of control.

Unfortunately, grid middleware is focused on the CPU utilization and still lacks storage solutions which offer functionality similar to the cloud storage. The access to the data is organized according to the computational requirements rather than users' needs.

The UNICORE middleware provides the possibility to expose distributed compute resources but in the case of distributed storage either separate solutions must be deployed and integrated or the system is limited to use separate storage resources independently. To access data a user has to know at which Vsite data is stored and then provide the path to the file. The Grid Browser plugin available in the UNICORE Rich Client is a good example of such hierarchical access to the data (see Figure 1). One should note that once a file is located, it can be used as input and output of the job or workflow and the UNICORE middleware takes care of the necessary file transfers.

There are few possible approaches which can mitigate this situation. One option is the DSMS system architecture developed recently[4]. It combines multiple UNICORE storage resources into a single logical resource. This resource is usable by both grid client software directly and grid services (for instance by the UNICORE Workflow Service). Users do not

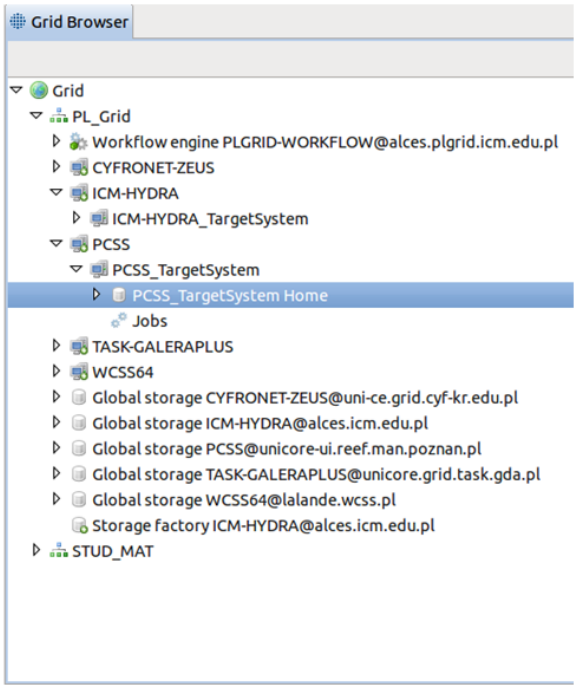

Figure 1. Screenshot presenting the actual version of the grid browser from the UNICORE Rich Client.

have to know where their files are stored physically. Optionally, advanced users have an opportunity to be aware of the distributed nature of the storage and physical files locations.

Another approach, described in this paper, is a uniform view of the distributed storage in the UNICORE Rich Client. This solution does not need installation of new components on the server side and provides similar functionality. Users can switch between traditional view of distributed storage organized in the form of the Vsite tree and a uniform view which hides information of the Vsites and provides a tree of files and folders. Users can browse resources as in the traditional model, use them as input and output files in the jobs and workflows.

2 Uniform Distributed Storage View

The logical structure of the uniform storage view is a contraction of the directory structures of the Vsites. For the directories of the same name but located at different Vsites single entry is generated with the same name and members which are files from all copies of the catalogue. For the files of the same name located in the directories of the same name located on different physical resources, the files are displayed separately in the directory tree with the original name followed by the name of the Vsite they are stored on. This simple solution allows to avoid naming problems and results in the well-defined tree of files and folders. The proposed solution obviously does not detect exact copies of the files and management of such copies is left to the user.

The uniform storage view has been implemented as Eclipse plugin[5] and can be loaded to the UNICORE Rich Client. It is configured as additional view called Storage Browser and does not replace the standard view offered by the Grid Browser. The user is provided with the possibility to switch between both views. An example of the uniform view on the resources is presented in Figure 2. The conflicting names of the files extended with the name of their Vsite are marked.

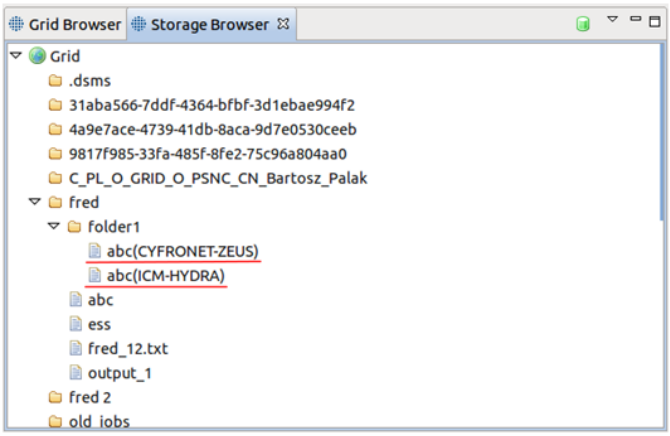

Figure 2. Screenshot presenting the storage browser with the uniform view on the data. Conflicting names of the files are extended with the name of Vsite and are marked.

The uniform distributed storage view can be used to browse resources in the Grid Browser but can also be used to browse files during job or workflow preparation. As presented in Figure 3 it allows the user to pick files for job input or output or to transfer files to and from user's computer. Once a file is selected using uniform storage view, its name is translated to one used by the UNICORE and standard procedures apply.

Some issues arise if a user selects not a file but output directory which consists of more than one physical locations. In such a situation, a file to be stored on disk might be created in any of the physical locations. In most cases this is not a problem, but sometimes the user would like to preserve better control on the location where data is stored for example because of the transfer speed. To address this issue we have provided a dedicated configuration panel which allows to set up Vsite priority, i.e. priority to use one or another Vsite to store data (see Figure 4). The higher value means higher probability of using the particular Vsite to store data, zero means that Vsite will not be used at all.

The uniform data view plugin allows also to perform some management tasks on the distributed storage. In particular it provides information on the disk space available at the different Vsites and measures transfer speed between user's computer and different Vsites as well as between Vsites. The results are displayed in the form of a matrix as presented in Figure 5. The transfer speed is measured by checking the transfer time of a file of known size filled with random data. The size of the file as well as the levels used to colour results can be set up by the user in the plugin preferences panel.

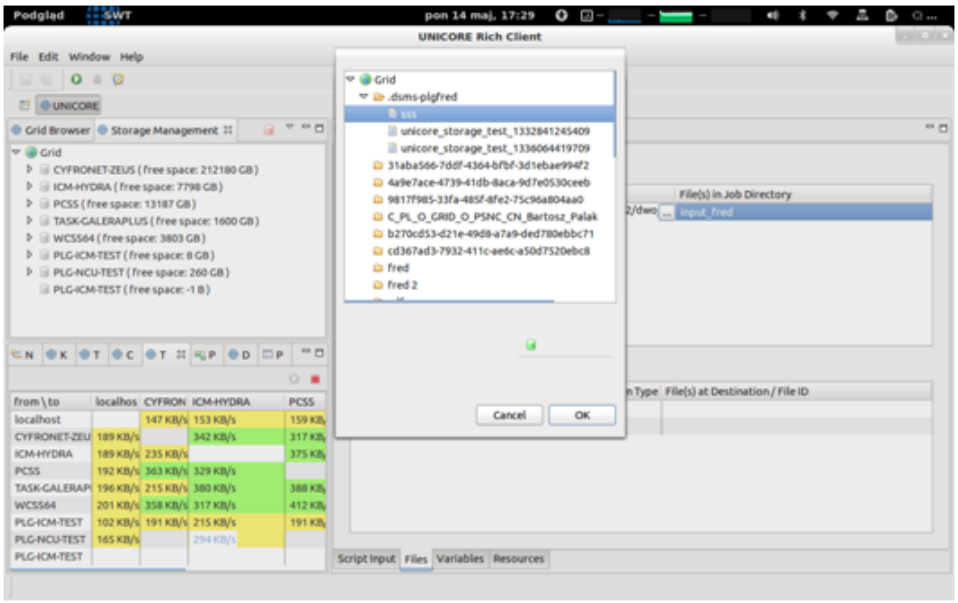

Figure 3. Fragment of a screenshot presenting usage of the uniform storage view for setting files to stage in and out.

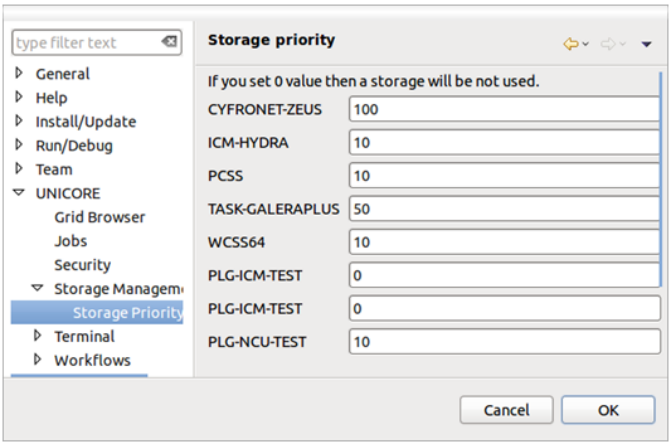

Figure 4. Fragment of a screenshot presenting configuration panel used to prioritize storages.

Another functionality implemented by the uniform storage view plugin is monitoring of the recently or more frequently used files. Such files are detected based on their access time and then can be moved by the user to the optimal physical storage (for example with the highest upload speed).

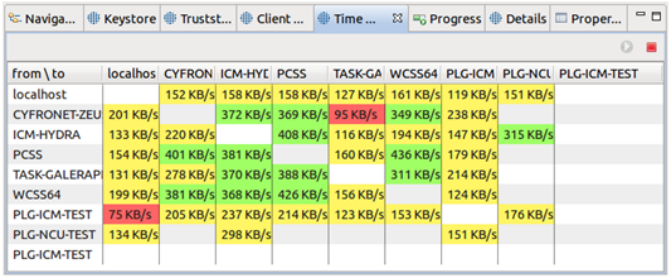

Figure 5. Fragment of a screenshot presenting results of the site-site transfer speed measurements.

3 Summary

The uniform distributed storage view plugin developed for the UNICORE Rich Client extends its functionality. In particular it presents to the user a uniform view of the files and directories in the whole grid hiding information on physical locations. This view can be used also during job and workflow preparation. The plugin allows also to perform some data management tasks on the distributed storage. It provides users with the information of space available at the different Vsites, checks transfer speed between user's computer and storages, provides matrix of transfer speed between different storages. The plugin allows also to monitor files most frequently used and move them to the storage optimal because of the transfer time.

4 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. Dropbox `http://www.dropbox.com`, accessed 21 August 2012.
2. Google Drive `http://drive.google.com`, accessed 21 August 2012.
3. I. Foster, *Globus Online: Accelerating and Democratizing Science through Cloud-Based Services*, Internet Computing, vol. 15 no 3 (2011), pp. 70–73.
4. T. Rękawek, P. Bała, K. Benedyczak, *Distributed Storage Management Service in UNICORE*, UNICORE Summit 2011 Proceedings, 6–7 July 2011 Toruń, Poland, IAS Series vol. 9, pp. 125–134, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 2011.
5. Clayberg E., Rubel D., *Eclipse Building Commercial-Quality Plug-ins*, Pearson Education Inc., 2004.

UNICORE Deployment in Polish NGI. Lesson Learned.

Krzysztof Benedyczak[1,2], Marcin Stolarek[1], Radosław Rowicki[1], Rafał Kluszczyński[1], Marcelina Borcz[1,2], Grzegorz Marczak[1,2], Maciej Filocha[1], and Piotr Bała[1,2]

[1] University of Warsaw,
Interdisciplinary Centre for Mathematical
and Computational Modelling,
Pawińskiego 5a, 02-106 Warsaw, Poland

[2] Nicolaus Copernicus University,
Department of Mathematics and Computer Science,
Chopina 12/18, 87-100 Toruń, Poland

In this paper we describe UNICORE infrastructure implemented in the Polish NGI. Special attention is given to the integration of the UNICORE middleware with the PL-Grid authentication and authorization framework. This solution allows for uniform infrastructure and user management across different middlewares. Monitoring and accounting of the UNICORE infrastructure integrated with the PL-Grid framework are described. We also present utilization of the Polish grid infrastructure using UNICORE access. It shows that UNICORE is popular among users and is considered as important and prospective method to access computational resources.

1 Introduction

The main aim of the Polish National Grid Infrastructure is to build sustainable grid infrastructure for the scientific community in Poland[1]. This has been achieved thanks to the PL-Grid project realized in the years 2009-2012. From its beginning project considered utilization of the most popular and successful grid middlewares such as gLite and UNICORE[2,3]. In this paper we describe UNICORE infrastructure implemented in the Polish NGI. Special attention is given to the integration of the UNICORE middleware with the PL-Grid authentication and authorization framework. The solutions for monitoring and accounting of the UNICORE infrastructure within the PL-Grid are also presented. In result uniform infrastructure and user management across different middlewares is realized.

The installation of the UNICORE is fully integrated with the PL-Grid portal which provides users with easy registration, certificate management and application for resources. In addition, the portal gathers and presents to the user information about the current allocation and resource utilization.

We present usage of the Polish grid infrastructure through UNICORE and compare it to the utilization of the resources with the gLite. The results show that UNICORE is popular among the users and is considered as important and prospective method to access computational resources.

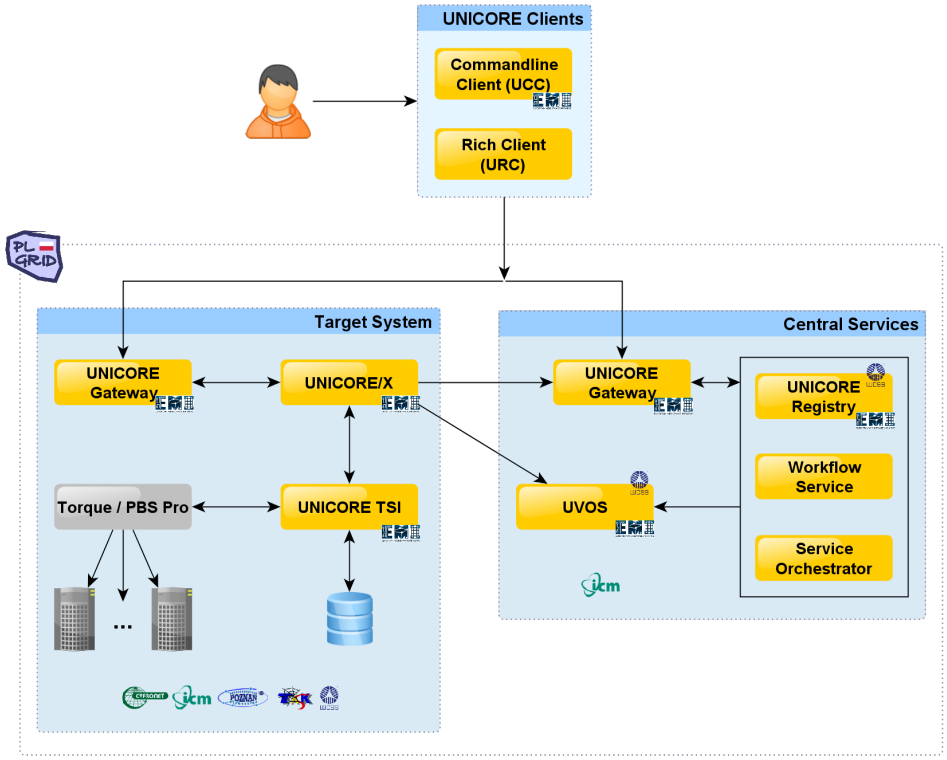

Figure 1. Schematic picture of the UNICORE infrastructure in the PL-Grid divided to the client tier, central services and target system. The components used from the European Middleware Initiative are marked.

2 UNICORE Infrastructure in PL-Grid

The UNICORE infrastructure in PL-Grid consists of 5 HPC centres which act as independent sites providing computational resources (see Figure 1.)[4]. Most of the UNICORE services are deployed at the main site located at ICM. The execution services are installed at all sites providing resources including ICM. The UNICORE security infrastructure is based on the UNICORE Virtual Organizations Service (UVOS)[5]. The UVOS service is populated with the data received from the PL-Grid LDAP repository. LDAP contains information about users, their privileges and certificates. This data is synchronized with the UVOS and is used to authorize and authenticate users.

The UNICORE infrastructure consists of three installations dedicated to the different purposes and users' communities. Because of that, they vary in scale and configuration. The production infrastructure provides users with the resources in the computer centres. The test infrastructure allows for testing solutions developed in the PL-Grid project as well as new versions of UNICORE. The services of the test infrastructure (except UVOS) are independent from other installations. Finally there are training facilities intended for training (including remote training). They allow for the execution of short calculations on

specially prepared system. The training infrastructure is separated from the two others. The details of the user's access to the various components of the PL-Grid infrastructure are presented in the Figure 2.

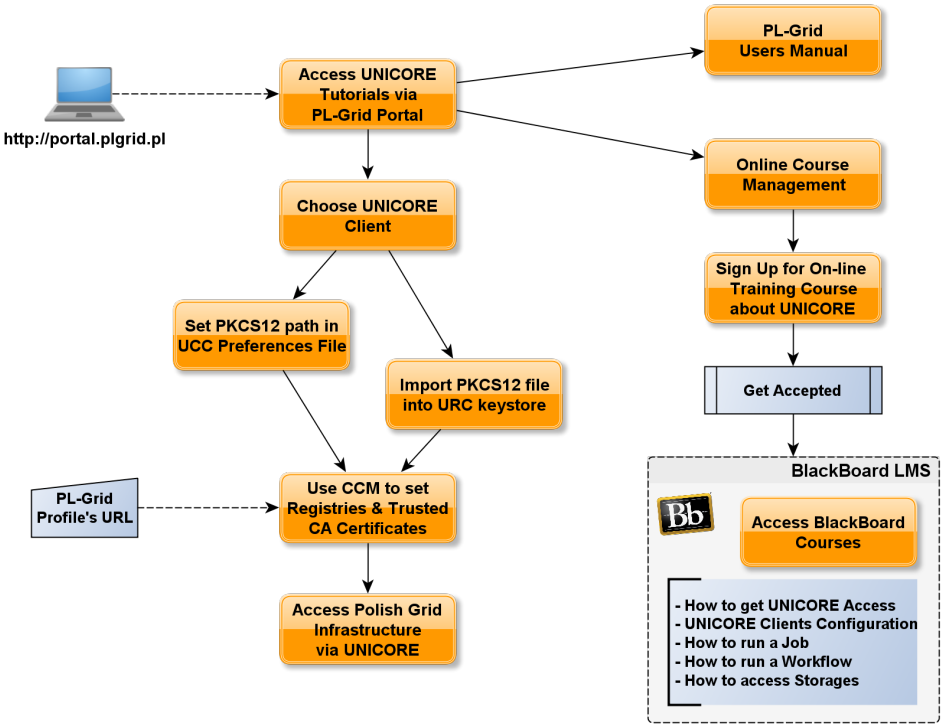

Figure 2. Schema of the user's access to the UNICORE resources including on-line training.

2.1 UNICORE Central Services

Central services are used by the all components of the UNICORE infrastructure. Therefore they are accessed by all users as well as services installed on the target systems. Central services store access addresses of the target systems, steer job submission process and maintain information associated with users. They include: (i) Registry, (ii) UVOS, (iii) Workflow Engine and (iv) Service Orchestrator.

Registry allows user to access all facilities through one address. UNICORE Clients and other middleware components contact global registry to obtain detailed information on the available services and servers. This information is provided automatically while registry location is known. Such solution is very convenient for the users. They have to be aware about URL pointing to the registry only which significantly simplifies access to the distributed resources.

UNICORE Virtual Organizations Service plays crucial role in the authentication and authorization of the users. It is actively used by the target systems installed in all centres. For each computational center there is established separate group in the PL-Grid virtual organization hierarchy. It allows for independent determination of members belonging to the group of the center and allows for modification of attributes within group. In order to reduce the number of connections to the target system it is possible to configure a temporary cache that allows to speed up operations. This eliminates identical queries within a short period of time. This process is especially helpful where trust delegation is used and several connections to UVOS are performed in a short time.

The Workflow Engine and Service Orchestrator are necessary for the processing the workflow jobs executed with the UNICORE system. The services use knowledge about the systems getting the target system information from the global registry. The use of workflows does not require any additional steps beyond the publication of the address in the central registry.

The configuration of the global services and their maintenance is more complicated than for the target system. Therefore we have decided to minimize number of instances. For the relatively small number of institutions providing resources in PL-Grid resulting in small number of target systems the central services are basically run in a single instance at ICM.

2.2 UNICORE Target System

The target system infrastructure provides access to the computational and storage resources available at the computational centres. This functionality is achieved through configuration and maintenance of the following: (i) UNICORE Gateway (ii) UNICORE/X and (iii) TSI.

The first component provides secure access on the basis of acceptance of certificate. The certificates issued for the PL-Grid project by the dedicated Certification Authority are used. The UNICORE/X is responsible for the services that allow access to the resources. Here we define which queues in the queuing system are made available to the user of the UNICORE infrastructure and applications which he can use. In addition, it implements the communication with the TSI, which mediates with the queuing system on the target system.

the services store address of their instance in the global registry, which allows users to detect available resources and use them in the user generated workflows. It should be noted that above services are actively communicating with two central services available at ICM: the global registry and UVOS.

2.3 Reliability

In the case of the UNICORE installation in the PL-Grid the reliability is achieved based on the principle of redundancy of services. Currently supported solution enables application failover scenario[4]. This involves setting up two copies of the same services. One of them is run as the main one and second as a backup. In the absence of contact with the main instance, all traffic is routed to a backup service. Such scenario is used for both central and target system services. To eliminate hardware problems, the main and backup central services are placed on the physically separate machines. The same solution is used for

the target system services. Redundant central services are configured to use the same data source. Furthermore, the reliability of the data source should be ensured. For the MySQL database it can be accomplished in several ways using standard database tools and functionality.

In the case of the UNICORE/X, the reliability is achieved by the redundancy of the data source using standard mechanisms offered by the database engine (e.g. MySQL). TSI does not require any additional configuration and backup. TSI instance works together with the UNICORE/X backup service. In order to achieve good performance, TSI is configured to work with the queuing system on the execution node.

The Gateway is a stateless component and does not need additional source data. This allows for the solution at the network level. On both machines (master and backup) it has the same configuration.

2.4 Monitoring

In the PL-Grid the Nagios system[7] is used as the main monitoring framework. There is a single instance of the Nagios which gathers information about status of all services. In result, system administrators see current status of the whole UNICORE infrastructure. Number of tests (Nagios probes) has been developed to verify operation of the UNICORE services[6,8].

They perform tests of the connection to the UNICORE/X server and test for reading and writing a file in a shared storage. Checks of the status of the disk space and a simple task submitted through a queue system are also executed. Depending on the applications available at the target system, tests are executed in the form of additional tasks.

Each central service has probes to test availability and access. Workflow tests are also performed. They involve sending tasks and files between target systems. Nagios checks the ability to perform tasks by each of the target systems, and automatic file transfer between them. When the test cascade of tasks performed correctly a run of a simple task on the target systems is not performed. This approach significantly reduces the number of jobs required by the monitoring system and improves performance. Additional probes are installed to analyse log files of each service. They require access to the log files and must be run on the machines that host services. Log analysis tests gather information on warnings and error messages appearing in the log files.

2.5 Accounting

One of the aims of the PL-Grid project is to oversee allocation of the resources to individual users or research teams. The accounting of the resources used by individual users and users groups plays important role in this process. In the BAT portal each user is able to view their activity at different target systems. However, in order to provide such information to the users, the history of jobs must be gathered and stored. For this purpose number of services has been developed for the UNICORE middleware[9,10].

Accounting system consists of several components presented in the Figure 3. First of them, **RUS BSS Adapter**, is running on the same machine as queuing system since it requires access to the log files with the resource consumption data. The **RUS Job Processor** is installed on the server hosting UNICORE/X service. It allows for the transmission of the

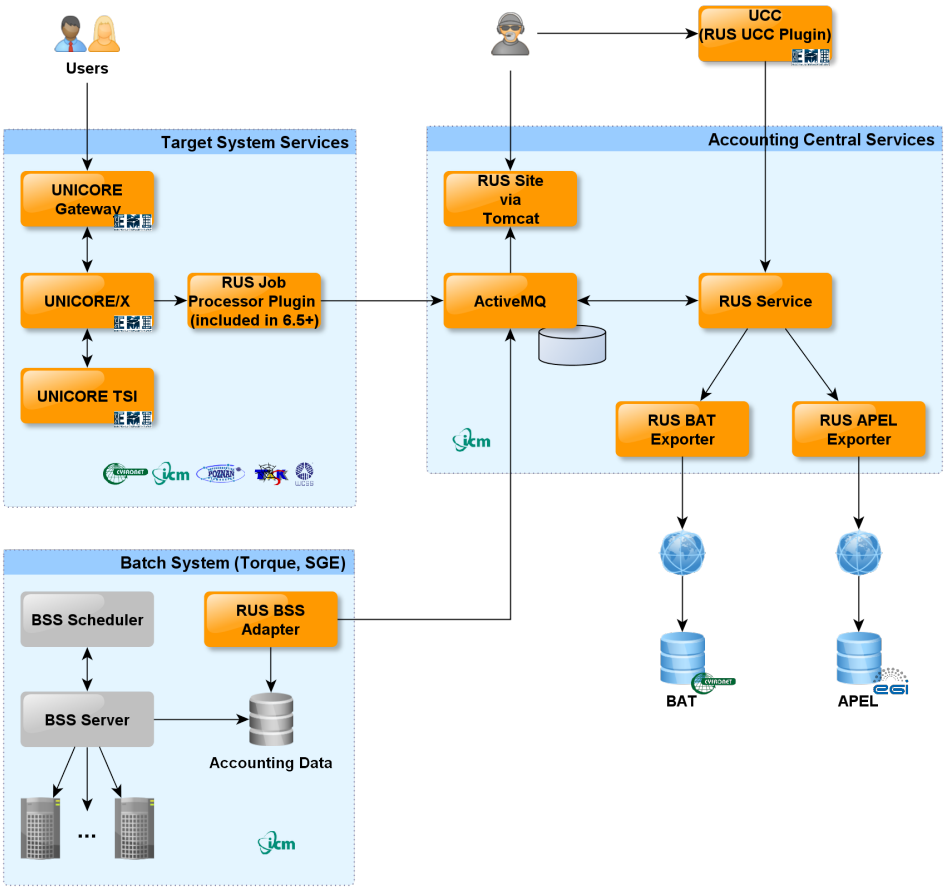

Figure 3. Schema of the UNICORE accounting in the PL-Grid.

accounting data to the **RUS Service** responsible for processing accounting data, integration and preparation to visualization within the BAT. It can be installed at any of the servers running UNICORE Services Environment (USE), in particular on the server hosting central services.

The **RUS UCC Plugin** extends the capabilities of the UNICORE Commandline Client (UCC). The **RUS UR Site** performs visualization of the data collected in the form of additional web application.

3 User Management in the PL-Grid

In order to use the PL-Grid infrastructure user has to register in the PL-Grid Portal. The process is schematically presented in the Figure 4. The portal provides interface for the users to obtain self-generated certificates issued by the PnP CA[11]. User can also apply for the PL-Grid CA certificates, but this requires personal contact with the RA. The PnP CA

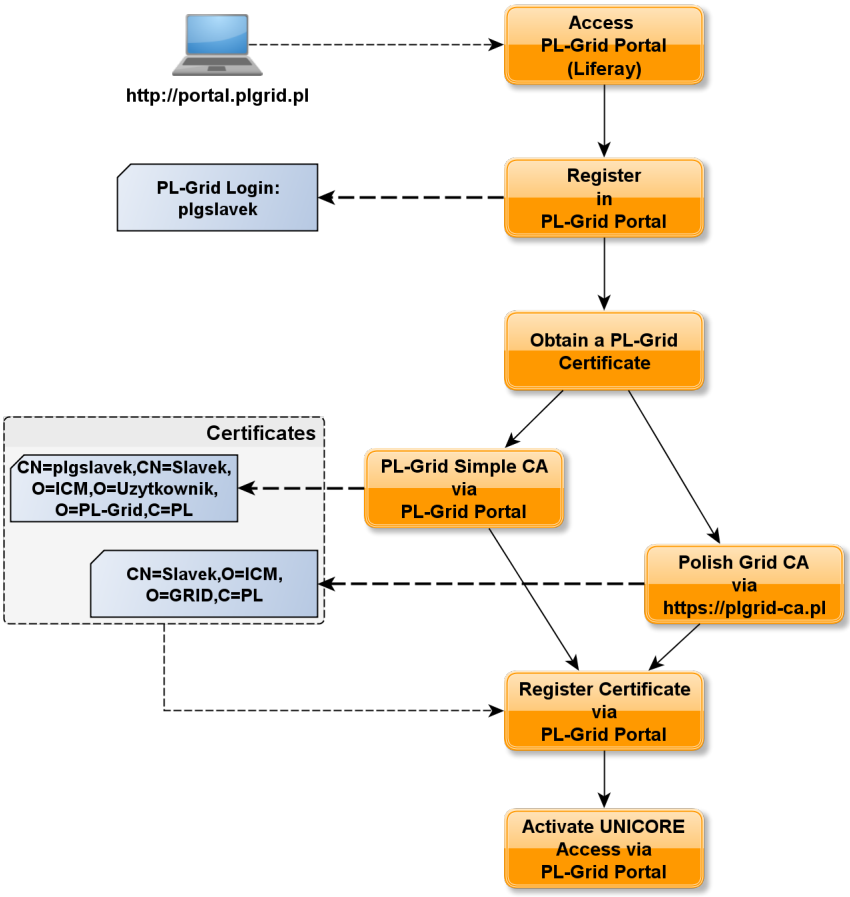

Figure 4. Schematic view of the new user registration in the PL-Grid. Please note that user can obtain certificate from one of the supported Certificate Authorities: Simple CA and PL-Grid CA. The whole process is managed by the PL-Grid portal.

server allows to generate certificate requests (CSR) an process them automatically. The PnP CA provides web interface to quickly approve new certificates for the users. Additionally, CA automatically sends the approved certificates to user, as well as information about new applications to the persons responsible.

The process itself uses an additional registration module registering applications in the virtual organization supplied with the server UVOS (uvos-webapp). It provides a customizable form, which defines data required from the user. Once form is filled server sends notification to the PnP CA with a request to issue a certificate. Additionally, the registration process determines the UVOS group to which the new user is attached.

This solution allows to hide less intuitive parts of the certificate issuing process and facilitates. The procedure performs placement in the UVOS additional information in the form of groups or attributes which are necessary to grant users with the proper privileges.

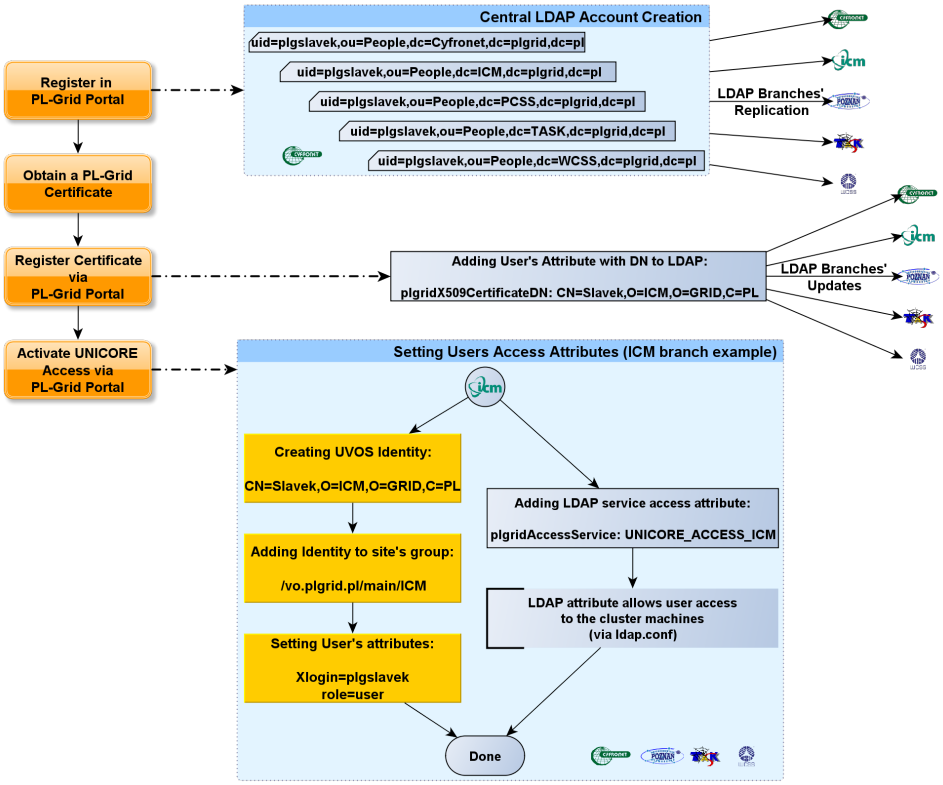

Figure 5. Schematic view of the management of the user certificate. The UVOS service is populated with the data from the PL-Grid LDAP repository. The dedicated branch is created for each resource provider.

4 Grants and groups

The usage of the PL-Grid infrastructure is controlled through grants. The resources allocations and their utilization is accounted within particular grant allocation. Each user can apply for a grant. Grant owner (manager) can add other users to it to share resources. There can be more than one person with the role of the grant management. The grant id is translated to the unix group id which allows users to share data on their accounts on the PL-Grid infrastructure.

In the case of the UNICORE, the groups are used to share data stored by the grant members in the UNICORE global storage available on the PL-Grid infrastructure.

Each user which registers in the PL-Grid portal and obtains certificate receives the default allocation through individual grant. It has some, not huge, resource allocation for short period of time. More extensive usage requires dedicated grant proposal with some description and formal request for resources. The whole process of the grant application, its acceptance and resource allocation is performed within PL-Grid portal. The process is schematically presented in the Figure 6. Once grant is active the proper information is stored in the LADP database and is propagated to the UVOS. Than this information is used

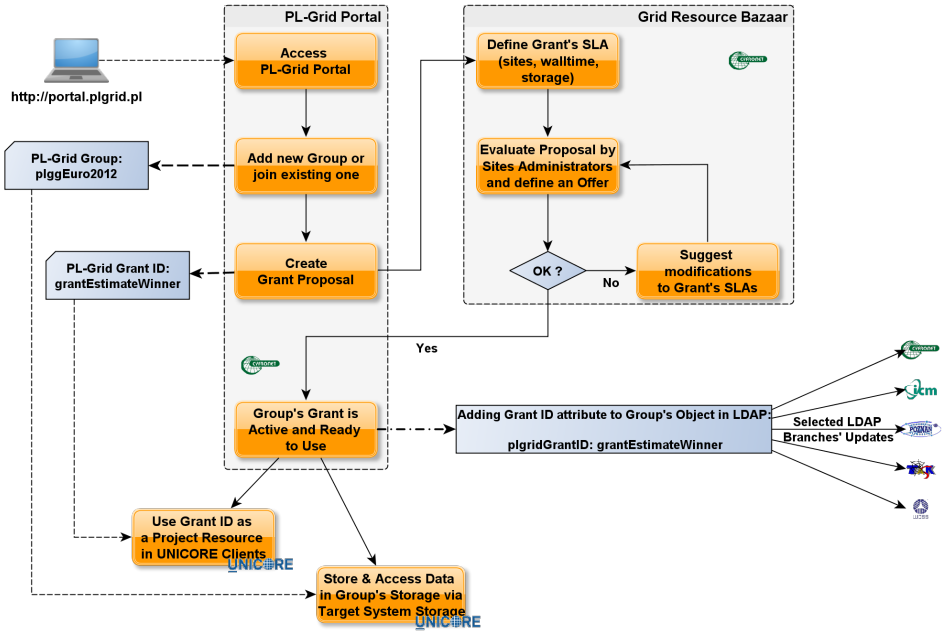

Figure 6. Schema of the process of the application for the grant.

in the job submission process. In particular, grant ID is used as a Project Resource in the UNICORE Clients.

5 Results

The UNICORE infrastructure has been successfully installed and integrated with the PL-Grid infrastructure. Despite initial resistance, all Polish HPC centres installed target system infrastructure and provided access to their resources through UNICORE. The monitoring and accounting of the jobs submitted through the UNICORE is implemented and integrated with the rest of the PL-Grid infrastructure.

Close collaboration with the site administrators allowed to simplify and automate the target system installation. In particular rpm packages prepared by the EMI project[12] has been widely used. The procedures to attach new resources to the UNICORE grid have been prepared and tested.

In result we have obtained stable infrastructure which offers PL-Grid users flexible access to the resources such as CPU and storage. From the user's point of view UNICORE is one of the access modes and does not require any special activities such as special keys, procedures etc.

In order to promote UNICORE in the PL-Grid project special attention has been delivered to the training of the users and system administrators. During three years of the project we have trained over 700 persons in the dedicated training sessions. More than 200 persons has been introduced to the UNICORE during the hands-on activities. Additionally,

Figure 7. Utilization of the PL-Grid infrastructure by the UNICORE and gLite users.

the on line training material has been prepared. It introduces users to the infrastructure, UNICORE, job submission, workflow creation and management. The on-line training has been hosted in the BlackBoard Lecture Management System and has been available to all users registered in the PL-Grid portal.

Thanks to the solutions implemented in the PL-Grid the access barrier to the infrastructure has been significantly lowered which attracted users. The results can be monitored by the number of the CPU hours utilized on the PL-Grid infrastructure and submitted using different middlewares. The data presented in the Figure 7. shows continuous growth for the jobs submitted using UNICORE. We observe that UNICORE is as popular as gLite for users outside high energy physics (LHC) which naturally dominates usage of the PL-Grid resources. However, number of jobs submitted and associated amount of CPU hours utilized on the infrastructure makes UNICORE important way of access PL-Grid infrastructure.

6 Acknowledgements

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

References

1. M. Bubak, T. Szepieniec, K. Wiatr, Kazimierz (Eds.), *Building a National Distributed e-Infrastructure – PL-Grid Scientific and Technical Achievements*, Lecture Notes in Computer Science, 7136, Springer, 2012.
2. P. Bała, K. Baldridge, E. Benfenati, M. Casalegno, U. Maran, K. Rasch K., B. Schuller, *UNICORE - a successful middleware for Life Sciences Grids*. In: M. Cannataro (Ed.), Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine and HealthCare, IGI, pp. 615–643, 2009.
3. A. Streit, P. Bala, A. Beck-Ratzka, K. Benedyczak, S. Bergmann, R. Breu, Jason Milad Daivandy, B. Demuth, A. Eifer, A. Giesler, B. Hagemeier, S. Holl, V. Huber, N. Lamla, D. Mallmann, A Shiraz Memon, M. Shahbaz Memon, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, T. Schlauch, A. Schreiber, T. Soddemann, W. Ziegler, *UNICORE 6 - Recent and Future Advancements*, Annales des Télécommunications, 65(11-12), pp. 757-762, 2010.
4. 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: M. Bubak, T. Szepieniec, K. Wiatr (Eds.), Building a National Distributed e-Infrastructure – PL-Grid Scientific and Technical Achievements, Lecture Notes in Computer Science, 7136, Springer, pp. 56–72, 2012.
5. P. Bała, K. Benedyczak, M. Lewandowski, A. Nowiński, *UNICORE Virtual Organizations System*. In: R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Wasniewski (Eds.), Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, 6068, Springer-Verlag Berlin Heidelberg, pp. 155–164, 2010.
6. P. Bała, K. Benedyczak, M. Strzelecki, *Monitoring of the UNICORE middleware*. In: A. Streit, M. Romberg, D. Mallman (Eds.), UNICORE Summit 2010 Proceedings, 18-19 May 2010 Juelich, Germany, IAS Series, vol. 5, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 2010, pp. 117-123.
7. W. Barth, *Nagios: System and Network Monitoring*, Open Source Press Gmbh, Munich, Germany, 2008.
8. `http://unicore-life.svn.sourceforge.net/viewvc/unicore-life/monitoring/`.
9. Bała P., Benedyczak K., Lewandowski M., *Resource Usage Accounting for UNICORE* In: A. Streit, M. Romberg, D. Mallman (Eds.) *UNICORE Summit 2010 Proceedings, 18-19 May 2010 Juelich, Germany* IAS Series vol. 5 Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 2010, pp. 75-82.
10. `http://unicore-life.svn.sourceforge.net/viewvc/unicore-life/accounting/`.
11. `http://pnp-ca.sf.net`.
12. `http://www.eu-emi.eu/`.

Interoperable Execution of eScience Applications on Grids & Clouds Through Open Standards

Daniele Lezzi[1], **Shahbaz Memon**[2], **Roger Rafanell**[1]
Hakan Soncu[4], **Morris Riedel**[2], **and Rosa M. Badia**[1,3]

[1] Barcelona Supercomputing Center,
E-mail: {daniele.lezzi, roger.rafanell, rosa.m.badia}@bsc.es

[2] Jülich Supercomputing Centre,
E-mail: {m.memon, m.riedel}@fz-juelich.de

[3] Artificial Intelligence Research Institute (IIIA), Spanish Council for Scientific Research (CSIC)

[4] MSR Advanced Technology Labs Europe
European Microsoft Innovation Center
E-mail: Hakan.Soncu@microsoft.com

The advent of cloud computing has offered scientific communities the ability to access computational resources that can satisfy their growing research needs starting to outgrow the size of traditional local resources as PCs and locally managed clusters and also of grid sites. Since grids and clouds are heterogeneous in nature and are based on different middlewares, interoperability between the service interfaces exposing the capabilities of these infrastructures is recognized as an important issue.
This problem is usually handled by using the appropriate adaptors to interact with several middlewares thus allowing the applications to be executed on federated infrastructures. While aiming for federated resources access, there is an overhead for the application clients to continuously detect and adapt to every evolution of the target middlewares.
In the presented work a complementary approach is followed to circumvent this problem by enabling interoperability between different execution services through the adoption of open and widely adopted standards.

1 Introduction and Motivation

The growth of cloud services and technologies has brought many advantages and opportunities to scientific communities offering users efficient and cost-effective solutions to the problem of lack of computational resources where to execute their applications. Recently, several grid initiatives[1–3] and distributed computing infrastructures have evaluated clouds in order to provide existing services through the use of virtualization technologies for the dispatch of scientific applications. On the other hand the spread of these technologies has introduced an important issue of how to make the new systems interoperable with the existing ones like grids. In the past this problem has been solved through the use of adaptors for specific middlewares and APIs[4] to provide the high-level abstraction that developers need in order to make the applications work across different distributed systems.

While the use of these adaptors enables the desired interoperability, there is an overhead posed on both the developers of the adaptors that have to continuously update the interfaces and on the application developers that need to update their components. With the introduction of cloud offerings for research, users started to deal once more with the

search for a solution to the well-known issue of which interfaces to adopt in order to interact with the actual execution environment. Furthermore, the use of IaaS offerings has moved the focus down to the protocols related to the provision of resources, while the work around grids and HPC middlewares[5,6] provided users with services able to offer job management capabilities, keeping them unaware of the technical details of accessing the computational nodes.

This work aims to demonstrate how e-Science applications can be seamlessly executed on heterogeneous infrastructures through the implementation of standards compliant interfaces for both grids and clouds. In particular the paper analyses the work performed in the VENUS-C project[12] for the design of a cloud platform and the implementation of the needed components through the adoption of open standards. The OGF OGSA-BES[7] and JSDL[8] specifications and their associated profiles[9] have been taken as a basis for the implementation of the VENUS-C Programming Model Enactment Services (PMES)for the porting and execution of the scenarios in the cloud infrastructure.

The proposed approach has been validated through the execution of a bioinformatics application on a testbed composed of public and private cloud resources offered by VENUS-C and HPC resources managed by UNICORE. The VENUS-C Platform provides a hybrid environment with private clouds provided by open source technologies like Emotive Cloud[10] and OpenNebula[11], and public virtual instances provided by Windows Azure. Two BES compliant Web Services have been developed, one for the enactment of applications on Azure through the Generic Worker (GW)[13] and another one for the execution of COMPSs (COMP Superscalar)[14] workflows on private clouds. The results demonstrate that the execution of the application benefits from the provision of cloud resources as potential backup of traditional HPC resources, still avoiding the implementation of additional adaptors.

The rest of the paper is structured as follows: section 2 presents the implementation of the standards interfaces for the job management components in VENUS-C and UNICORE, section 3 evaluates the proposed approach through the enactment of a use case in an hybrid environment and section 4 concludes the paper.

2 Implementations

The Basic Execution Service specification defines the service operations used to submit and control jobs within a job management system. BES defines two WSDL port types: BES-Management for managing the BES itself, and BES-Factory for creating, monitoring and managing sets of activities. A BES manages activities, not just jobs, but the activity in the context of the enactment services in VENUS-C is an e-Science code developed for instance as a COMPSs application and executed in the actual infrastructure through the actual

2.1 UNICORE BES

UNICORE is an open source middleware that provides seamless and intuitive access to various abstractions for grid computing resources such as supercomputers, clusters, and server farms. It is implemented using a set of Web Services standards and designed to offer an

easy interface for scientists to perform scientific computations related to Life Science, Virtual Physiological Human, Fusion Science etc. UNICORE has been demonstrated and used in large scientific e-Infrastructures like XSEDE[15], PRACE[16], and EGI. UNICORE specializes in interfacing with the majority of resource management and scheduling systems like Loadleveler[19], SLURM[17], Torque[20], and LSF[18].

One of the design principles of UNICORE architecture is to adopt open grid and Web Service standards. From Web Services standards, it implements the Web Service Resource Framework (WS-RF)[21] and Web services Addressing (WS-A)[22] while with regards to grid standards it is compliant with OGF Open Grid Services Architecture[24]. UNICORE job management services are exposed through an OGSA-BES interface with the job description request model realized through OGF's JSDL specification. Grid expert communities recommend the use of OGSA-BES and JSDL in specialized environment through the use of profiles which impose the elements that are required in a specific deployment. UNICORE supports almost all the recommended job management profiles such as HPC-FS[25] and HPC-BP, both defining elements of OGSA-BES and JSDL in HPC environments.

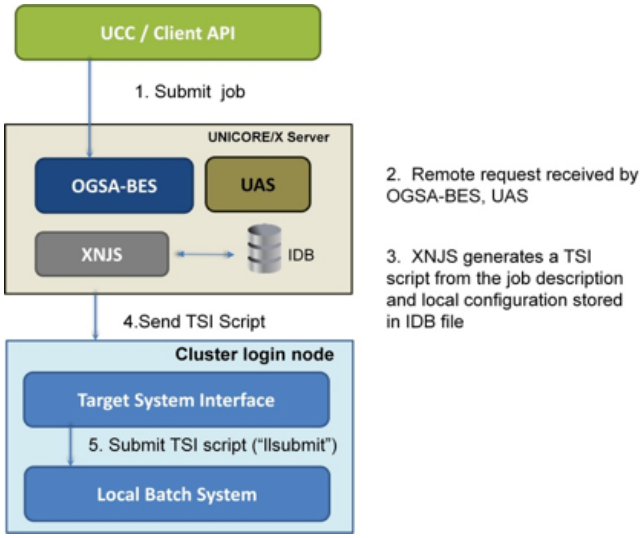

Figure 1. UNICORE-BES Architecture

Figure 1 depicts the UNICORE architecture for job management components and also describes the steps of job submission. The client requests are firstly processed by OGSA-BES and UAS(UNICORE Atomic Services) which are implemented as SOAP Web Services. UAS is a suite of proprietary services that provide job and data management capabilities complementary to OGSA-BES. The client tier includes a command line client and a programmatic API. Following are the optimal steps of job execution initiated by a user.

- The job request is validated against the basic parameters such as whether a BES is allowed to accept any more jobs e.g. this may be due to any unavailability of compute resources or to the fact that the user is exceeding the limit of jobs he is allowed to

execute. Once the request is successfully validated it is transferred to the XNJS[23] execution framework.

- XNJS internally manages all the jobs instantiated and running on a targeted computing resource. At this stage XNJS performs resource matching with client request by referencing the IDB (resource definition file). Then it furnishes a user request by transforming the incoming JSDL instance to a set of environment variables and commands understandable by a TSI (Target System Interface) instance which is deployed on the target resource management environment and is responsible for real execution in a batch system.

- In this step XNJS transmits the script to the TSI through a secured connection. In a cluster or HPC environment it is deployed on the Login node.

- Once the instructions are received from XNJS, TSI resolves and converts them to the format of the target resource management scripts. Finally the job is submitted. Until the job termination TSI monitors the job state and during that phase it notifies XNJS, which further notifies up to the services layer where OGSA-BES is deployed.

2.2 The VENUS-C Job Management

VENUS-C develops and deploys an industrial-quality service-oriented cloud computing platform based on virtualization technologies, to serve research and industrial user communities by taking advantage of previous experiences and knowledge on grids and supercomputing. The ultimate goal is to eliminate the obstacles to the wider adoption of cloud computing technologies by designing and developing a shared data and computing resource infrastructure that is less challenging to implement and less costly to operate.

One of the requirements of VENUS-C architecture is to define a way to support multiple programming models at the same time. In order to shield the researcher from the intricacies of the concrete implementation of different programming models, each one is enacted behind a job submission service, where researchers can submit jobs to and manage their scientific workload. For this purpose, VENUS-C provides OGSA-BES compliant services, the Programming Model Enactment Services, which take care of the actual enactment of a job in a specific cloud provider.

2.2.1 The COMPSs Enactment Service

The porting of the COMPSs framework to the VENUS-C platform includes the development of several components depicted in Figure 2. A BES compliant enactment service has been developed to allow researchers to execute COMPSs applications on the VENUS-C infrastructure. The researcher uses a client to contact the PMES in order to submit the execution of a COMPSs application. This request is expressed through a JSDL document containing the application name, the input parameters and data references, following the HPC-BP specification. The same client allows the user to interact with the cloud storage to upload a packaged version of his code; such package contains the COMPSs binaries and configuration files needed by the application.

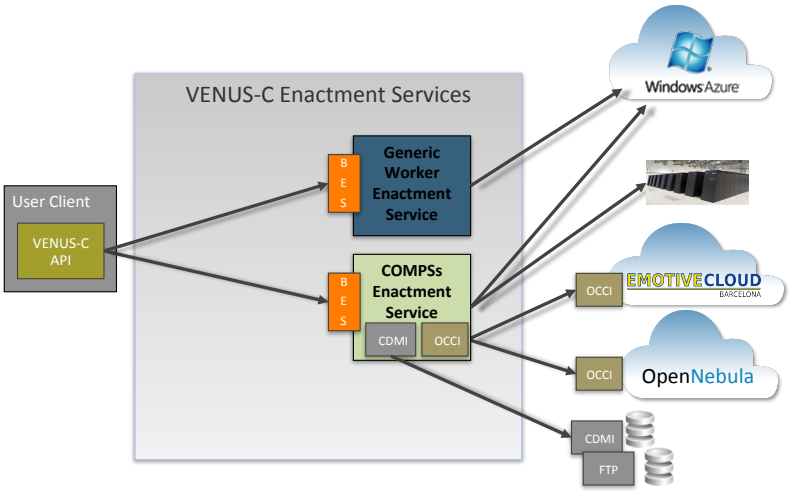

Figure 2. The VENUS-C Job Management

The implementation of the COMPSs BES includes a Job Manager and a Job Dispatcher that actually execute the application and manage its life cycle.

The implementation of the BES-Factory port type for the COMPSs enactment service supplies the following operations:

- **CreateActivity:** This operation is used to submit a job. It takes an ActivityDocument (JSDL document) describing the application to run, and returns an identifier of a WS-Addressing EndpointReference (EPR). This EPR can be used in subsequent operations. The JSDL document is forwarded to the Job Manager that, through the *StartJob* method, creates an enactment service job object assigning it a Universally Unique Identifier (UUID); a SAGA job is created and queued in the Job Dispatcher which is responsible for dealing with execution as is described in the following section.

- **GetActivityStatuses:** This operation accepts a list of EPRs (previously returned from CreateActivity), and returns the state of each COMPSs job. The EPR is parsed in order to get the UUID of the job. The Job object stored in the Job Manager is requested, and its status retrieved through the *GetJobStatus* method. The state model includes a basic set of states being: *Pending, Running (Staging-In, Executing, Staging-Out)* and *Finished* for normal execution stages and *Failed* or *Cancelled* for abnormal execution situations. A controller for expired jobs takes care of *FAILED* or *FINISHED* jobs removing them from the list of managed jobs after a maximum allowed time using two configurable timeouts.

- **GetActivityDocuments:** This operation accepts a list of EPRs and return an ActivityDocument for each EPR requested. The ActivityDocument just wraps a JSDL document containing the job description; it has been implemented for specification compatibility reasons.

- **TerminateActivities:** Takes a list of EPRs of COMPSs jobs to be terminated and returns true or false for each job depending on whether the operation was successful or not. The UUID is extracted from the EPR and the *TerminateJob* method is called to change the status of the Job object stored in the Job Manager to *CANCELLED*; the Job Dispatcher is notified so that the job can be terminated if running or not started if still in the queue.

The Job Manager delegates the whole control of the execution to the Job Dispatcher (see Figure 3), which maintains a user-resizable pool of threads that is responsible for picking jobs from a Job Dispatcher queue filled by the Job Manager. First a virtual machine is requested from the cloud provider in order to deploy the COMPSs runtime that schedules the tasks on the remote nodes. A SAGA SSH persistent connection is established with this machine where the Job Dispatcher deploys the application downloading its package from a cloud storage. This package includes the binary, the scripts for setting the required environment variables, and other configuration files needed both by the COMPSs runtime and by the application itself. Once the runtime machine is deployed the COMPSs application

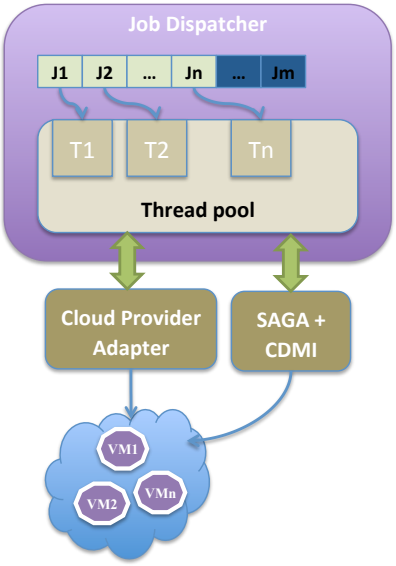

Figure 3. The VENUS-C COMPSs Job Dispatcher

is remotely started using the SAGA GAT adaptor. In its turn, the COMPSs runtime will schedule the tasks on a set of machines created on demand. This phase includes the staging of input files from remote locations as specified in the JSDL document. The format and protocol of these location references depend on the storage provider that hosts the data; in the case of VENUS-C platform the cloud storage is accessed in two ways, either through methods of an SDK tailored to the cloud storage or through a data access proxy service which can wrap multiple cloud storage systems. In the latter case, a CDMI[27] compliant web service is made available to the users and to the programming models.

When the job is completed, the output data is moved back to the storage according to the user JSDL request, then the Master VM is shutdown and the information on the performed activities is available in the Job Manager for a time specified by the *expired jobs controller*.

2.2.2 The Generic Worker

The Generic Worker consists of a worker process (a Windows Service or UNIX daemon process) which runs 24/7 on a virtual machine in the cloud. The pattern enables running multiple machines with generic workers concurrently so that new work items can be distributed and scaled across a set of nodes according to the application requirements. A GW client submits jobs and manages them through the GW PMES.

Figure 4 shows how all the concepts described above fit together. The researchers upload the executable to the storage service (Application Storage) where the GW has access to. Various existing services including HTTP, FTP and cloud storage services can be used for this purpose. Together with the executable, the researcher also provides an application description which contains the metadata information of the application. This metadata information is needed by the GW in order to understand the input and output parameters of the application.

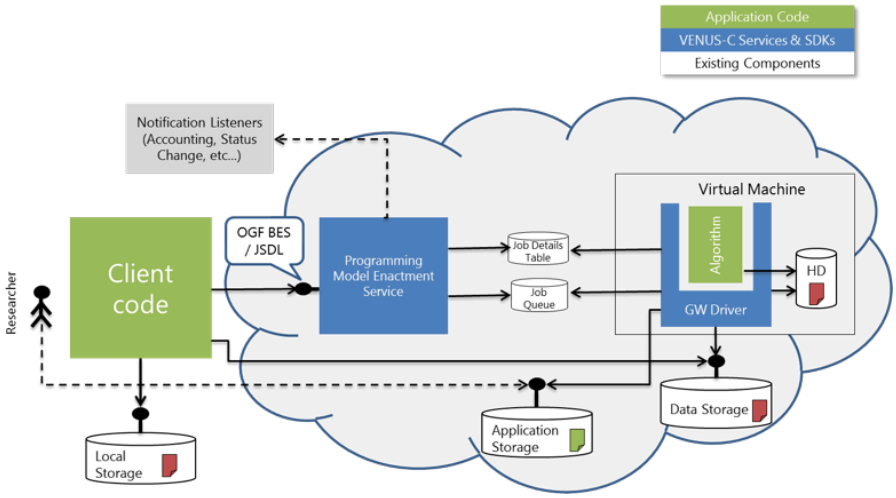

Figure 4. The Generic Worker Architecture

After uploading the application together with its metadata, the researcher uses the client application to upload input data from local storage to the storage service named as data storage in the Figure 4 which again should be reachable by the GW. The researcher submits batch jobs through the PMES. In order to provide security in the enactment service, various security mechanisms including STS and username/ password can be applied. The PMES translates the job submission request into a GW internal job representation, and injects the job into the GW's job queue and details table. One of the generic workers (assuming

the service is deployed on multiple virtual machines) retrieves a potential job from the queue/table system, and uses the data and application storage to determine whether all necessary input data sets, as well as the required algorithms, are available. If the job is executable and all the required data is available, the GW claims responsibility for this job in the queue/table system. If the requested application is not yet installed on the machine, the GW installs the application binaries from the application storage. It retrieves all necessary input data sets from the data storage. Now, all prerequisites for a successful job execution should be in place: the application binaries are installed and the necessary input data is on the local hard disk. The GW then launches the application with the help of the application description.

After completing the job, the GW uploads the results to the data storage. Finally, it notifies the PMES about completion or failure of the job through the job queue and table. Depending on whether the enactment service has been configured to support client-side notifications, the researcher is notified using the scenario- and domain-specific notification mechanism.

2.3 VENUS-C BES Extensions

Name	**Description**
GetJobs	Retrieves the jobs of a specific owner
GetAllJobs	Retrieves list of all jobs in the system
GetRoot	Retrieves the root of the job hierarchy
GetHierarchy	Retrieves list of all jobs which are below the given job in the hierarchy; if the root of the hierarchy is given, all the jobs in the hierarchy will be retrieved
CreateSubscription	Creates a notification subscription for all statuses
CreateSubscriptionForStatuses	Creates a notification subscription for a list of statuses
Unsubscribe	Remove a notification subscription from the system
ListDeployments	Get a list of deployments (pairs of CloudProviderID, InstanceCount)
UpdateDeployment	Modify the number of instances on a specified cloud provider

Table 1. VENUS-C BES Extensions

In VENUS-C, the basic BES specification is fully supported by the GW and COMPSs enactment services along with other extensions required by the VENUS-C end-users for retrieving extended job information, notification and scaling. The interfaces for these services are described in Table 1.

3 Validation

In order to validate the proposed approach for interoperability, a bioinformatics application has been executed on an hybrid testbed composed of cloud resources provided by the VENUS-C platform through the PMES and grid nodes available through a UNICORE endpoint. The UNICORE command line client has been used to interact with both grid and cloud endpoints in order to submit the BLAST workflow depicted in Figure 5 which has been ported to COMPSs in the context of VENUS-C. BLAST[26] is a widely-used bioinformatics tool for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or the nucleotides of DNA sequences with sequence databases, identifying sequences that resemble the query sequence above a certain threshold. The work performed by BLAST is computationally intensive and embarrassingly parallel, which makes it a good candidate to benefit from distributed environments like grids and clouds. Figure 6 depicts the testbed used for the execution that included 96 cores in a private cloud managed by the Emotive Cloud, 20 small Windows Azure virtual instances and grid nodes available in a cluster at Jülich Supercomputing Centre.

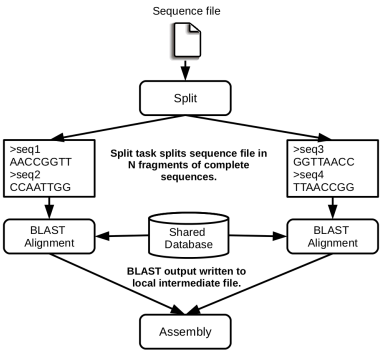

Figure 5. The Blast workflow

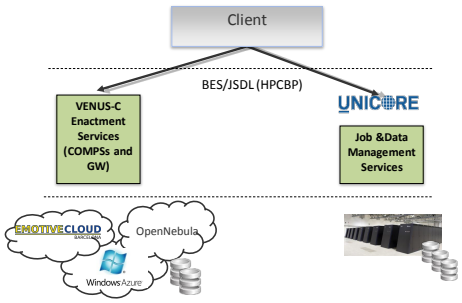

Figure 6. The Testbed

The use case for this experiment consisted of the alignment of the sequences of the metagenome of the Sargasso sea with respect to the Swissprot database. The Sargasso Sea sample was reduced to 1000 sequences and the database occupied about 260MB comprising 536.029 sequences. Half of the computation was executed on the VENUS-C platform and the other half on the UNICORE nodes, with the results merged locally on the user's laptop. The input (database and sequences file) and output (blast output file and several logs) were deployed on the storage by using the stage-in stage-out references in the JSDL job description. This testbed was used to simulate a scenario where a scientist can leverage traditional HPC resources for the computation turning to cloud for peaks of computational loads. This happened without the use of different clients and the need to remind a different syntax for each one of the platforms. It is worth nothing that the aim of the test was to prove the interoperability and not to evaluate the performances of the execution in the hybrid testbed compared to the grid one.

4 Conclusions

This paper presented the work for achieving interoperability between traditional grid and HPC systems and recently developed cloud platforms. The motivation behind this work derives from the need to offer researchers new computing paradigms without the drawback of continuously adapting the existing frameworks.

The UNICORE middleware and the VENUS-C Platform have been considered for the deployment of a testbed for the execution of scientific workloads through the provision of OGSA-BES compliant interfaces for job submission. UNICORE is a well-established middleware for accessing grid resources already adopted in several e-Infrastructures while the VENUS-C Platform, on the other hand, is an interesting use case for interoperability because it is the first implementation of the BES specification for Clouds.

The proposed approach has been successfully verified through the execution of a bioinformatics scenario seamlessly enacted on the available resources using the same client for job submission. The implementation of such testbed demonstrates that the challenge of interoperability can be achieved through the adoption of open standards.

Acknowledgements

This work has been supported by the Spanish Ministry of Science and Innovation (contracts TIN2007-60625, CSD2007-00050 and CAC2007-00052), by Generalitat de Catalunya (contract 2009-SGR-980), and the European Commission (VENUS-C project, Grant Agreement Number: 261565). This work was also made possible using the computing use grant provided by Microsoft in the VENUS-C project.

References

1. D. Kranzmüller, J. Marco de Lucas, and P. Oster, *The European Grid Initiative (EGI): Towards Sustainable Grid Infrastructure*, Springer US, pp. 61–66, 2010.
2. StratusLab Project, `http://www.stratuslab.eu`, accessed May 2012.
3. European Middleware Initiative, `http://www.eu-emi.eu`, accessed May 2012.
4. T. Goodale and S. Jha, *A Simple API for Grid Applications (SAGA)*, Grid Forum Document GFD-RP.90, `http://www.ogf.org/documents/GFD.90.pdf`, 2011.
5. I. Foster, *The Globus Toolkit for Grid Computing, Cluster Computing and the Grid*, IEEE International Symposium on, p. 2, First IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01), 2001.
6. E. Laure, C. Gr, S. Fisher, A. Frohner, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, et al., *Programming the Grid with gLite*, Computational Methods in Science and Technology, v. 12(1), pp. 33–45, 2006.
7. I. Foster et al., *OGSA Basic Execution Service Version 1.0*, Grid Forum Document GFD-RP.108, `http://www.ogf.org/documents/GFD.108.pdf`, 8/8/2007.
8. A. Savva et al., *Job Submission Description Language (JSDL) Specification, Version 1.0*, Grid Forum Document GFD-R.056, `http://www.gridforum.org/documents/GFD.56.pdf`, 7/11/2005.

9. B. Dillaway, M. Humphrey, C. Smith, M. Theimer, G. Wasson, *HPC Basic Profile Version 1.0*, Grid Forum Document GFD-RP.114, http://www.ogf.org/documents/GFD.114.pdf, 2007.
10. I. Goiri, J. Guitart and J. Torres, *Elastic Management of Tasks in Virtualized Environments*, XX Jornadas de Paralelismo (JP 2009), Coruna, Spain, 2009.
11. Open Nebula, http://opennebula.org.
12. Virtual multidisciplinary ENvironments USing Cloud infrastructures Project, http://www.venus-c.eu, accessed May 2012.
13. Y. Simmhan and V.C. Ingen, *Bridging the Gap between Desktop and the Cloud for eScience Applications*, Microsoft Research, U.S., http://research.microsoft.com/pubs/118329/Simmhan2010CloudSciencePlatform.pdf, 2010.
14. D. Lezzi, R. Rafanell, A. Carrion, I. Blanquer, V. Hernandez and R.M. Badia, *Enabling e-Science applications on the Cloud with COMPSs*, Euro-Par 2011: Parallel Processing Workshops, Lecture Notes in Computer Science, vol. 7155, pp. 25–34, Springer, 2011.
15. Extreme Science and Engineering Discovery Environment, http://www.xsede.org.
16. PRACE Annual Scientific Report 2012, http://www.prace-project.eu/PRACE-Scientific-Annual-Report, accessed May 2012.
17. M. Jette and M. Grondona, *SLURM: Simple Linux Utility for Resource Management*, Linux Clusters: the HPC Revolution Conference, (San Jose, USA), 2003.
18. Platform LSF, http://tinyurl.com/c9ln6b8, accessed May 2012.
19. IBM Loadleveler, http://www-03.ibm.com/systems/software/loadleveler/.
20. TORQUE Resource Manager, http://www.adaptivecomputing.com/products/open-source/torque/.
21. T. Banks (Editor), *OASIS Web Services Resource Framework (WSRF) Primer v 1.2*, http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf.
22. W3C Web Services Addressing (WS-Addressing), http://www.w3.org/Submission/ws-addressing/, 10 August 2004.
23. B. Schuller, R. Menday and A. Streit, *A Versatile Execution Management System for Next-Generation UNICORE Grids*, Euro-Par 2006: Parallel Processing, Lecture Notes in Computer Science, vol. 4375, pp. 195–204, Springer Verlag, 2007.
24. I. Foster et al., *Open Grid Services Architecture Version 1.5*, GFD.80, http://www.ogf.org/documents/GFD.80.pdf, 2006.
25. G. Wasson and M. Humphrey, HPC File Staging Profile Version 1.0, GFD.138, http://www.ogf.org/documents/GFD.138.pdf, 2008.
26. S.F. Altschul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman, *Basic Local Alignment Search Tool*, Journal of Molecular Biology, 1990, v. 215(3), pp.403–410, doi:10.1006/jmbi.1990.9999.
27. SNIA CDMI, http://www.snia.org/tech_activities/standards/curr_standards/cdmi/.
28. SwissProt protein database, http://www.ebi.ac.uk/uniprot.

UNICORE 2020 - Strategic Options for the Future

Morris Riedel[1], **Andrew Grimshaw**[2], **and Thomas Lippert**[1]

[1] Jülich Supercomputing Centre,
Forschungszentrum Jülich, 52425 Jülich, Germany
E-mail: {m.riedel, th.lippert}@fz-juelich.de

[2] School of Engineering and Applied Science, University of Virginia,
151 Engineers Way, Charlottesville, Virginia, USA
E-mail: grimshaw@cs.virginia.edu

International e-Infrastructures offer a wide variety of Information and Communications Technology (ICT) services that federate computing, storage, networking and other hardware in order to create an 'innovative toolset' for multidisciplinary research and engineering. UNICORE services are known to be secure, reliable, and fast providing researchers all over the world with powerful software that enables the use of those e-Infrastructures as a 'commodity tool' in daily geographically distributed activities. As key technology provider of the European Grid Infrastructure (EGI), UNICORE is available as part of the Unified Middleware Distribution (UMD) serving the needs of researchers that require mainly High Throughput Computing (HTC). On the other end of the scale, UNICORE offers specifically optimized resources within the Partnership for Advanced Computing in Europe (PRACE) today. Beyond Europe, UNICORE installations are emerging more and more such as within the Extreme Science and Engineering Discovery Environment (XSEDE) US multi-disciplinary e-Infrastructure (aka Cyberinfrastructure) offering both HTC and HPC resources. The grand challenges in science, engineering, and in society that need to be solved towards 2020 and beyond will increasingly require both geographical and intellectual collaboration across multiple disciplines. International e-Infrastructures are considered to be one key toolset to tackle those grand challenges and this contribution will outline several options how UNICORE can remain one 'technology of choice' towards 2020. A strategic roadmap is presented that illustrates the role of UNICORE alongside the European Commission's (EC) vision for Europe in 2020, including the opportunities that arise for UNICORE in the context of the Digital Agenda for Europe. The roadmap also includes how UNICORE can play a role to tackle, or perhaps rather contribute with processing to the avoidance of 'big data waves' arising from a wide variety of e-Infrastructure users emerging from the European Strategy Forum on Research Infrastructures (ESFRIs).

1 Introduction

The title of this publication is inspired by several long-term agendas, roadmaps, and plans that highlight necessary actions towards 2020 and beyond that affect the existing e-Infrastructures in Europe and beyond. These include the new plans around the Horizon 2020 EC framework[29] as well as it's aligned Digital Agenda for Europe[30]. Furthermore, user communities organized themselves in different scientific thematic groups with numerous projects mentioned as part of the ESFRI roadmap[32] using 'decades' for timescales. ESFRI user communities bear the potential to significantly increase the amount of UNICORE users while at the same time broadening its usage beyond computational resources towards storage resources with 'big data'. But in order to identify future options, this paper firstly goes back to our earlier published UNICORE roadmap from 2010[21] in order to take stock of the current situation. Based on this, this paper outlines future options for the UNICORE community to engage in various activities potentially leading to new partners and new projects towards 2020 supporting innovation in science and ICT.

The remainder of this paper is structured as follows. After the introduction, Section 2 provides a status assessment of our UNICORE roadmap provided in 2010. Section 3 then reveals an update of this roadmap outlining strategic options for future UNICORE activities towards 2020. This paper ends with some concluding remarks.

2 Review 2010 UNICORE Roadmap

This section briefly aims to take stock of many UNICORE activities performed alongside the UNICORE roadmap presented in 2010[21]. Because of the focus on the new roadmap towards 2020, this paper only highlights the progress towards major key goals in context of 'lighthouse user communities' that bear the potential for hundreds or thousands of scientific users instead only a handful of researchers. Other UNICORE activities can be obtained from UNICORE Summit proceedings of the past couple of years.

The status assessment of the initial roadmap starts with the role of UNICORE in major Distributed European Infrastructure for Supercomputing Applications (DEISA)[14] applications organized around the DEISA Extreme Computing Initiative (DECI)[26]. DECI virtual communities of the EU Fusion for ITER Applications (EUFORIA) project[22] and the Virtual Physiological Human (VPH)[15] used UNICORE in conjunction with scientific applications on DEISA HPC resources. Closer collaborations with these projects created not only scientific results, but also advancements in software support for UNICORE such as the KEPLER UNICORE actors being part of the KEPLER Serpens Suite[13] and the support of UNICORE by the Application Hosting Environment (AHE)[11] used in VPH.

The European Middleware Initiative (EMI) project[31] enabled the UNICORE technology to be part of the EGI UMD distribution that can be considered as another success in fulfilling elements of the 2010 roadmap. Being part of EMI and EGI with UMD, UNICORE is much more known in Europe and around the world as ever before and several National Grid Initiatives (NGIs) (e.g. NGI-DE, PL-Grid, etc.) already provide access via UNICORE. EMI also leads to many other benefits for UNICORE such as improved documentation, testing and release processes, as well as being better recognized in the European Grid community. EMI activities[23] also enabled UNICORE with the access to data storage technologies via the Storage Resource Manager (SRM) standard[5] in order to obtain data from the Large Hadron Collider (LHC)[35] and other EGI user communities.

UNICORE is used in PRACE that in turn is the first ESFRI project that takes advantage of UNICORE for a multi-disciplinary scientific community perspective. More user community-specific activities of UNICORE partners included their engagement in other ESFRI projects (e.g. CLARIN[36], DARIAH[39], SKA[37] etc.) where the role of UNICORE components will be still analysed during the course of the next couple of years.

Other parts of the fulfilled roadmap elements include the exploration of UNICORE potentials towards e-Business and Green IT. This includes the use of Service Level Agreements (SLA) around UNICORE (e.g. SLA4D-Grid[38]) and license management (e.g. SmartLM[40]) while also the EMI project towards its end of the project is performing many commercial outreach activities. In terms of GreenIT, UNICORE Target System Interface (TSI) work has been performed to support 'Green IT setups' (e.g. Fit4Greeen Project[42]). Finally, UNICORE partners also engaged in exploring the trend towards 'cloud computing' (e.g. EMI cloud and Grid integration research[20]) while the new 2020 roadmap will address this topic in more detail as it becomes more and more important in the next years.

3 UNICORE Roadmap Update towards 2020

Based on the achievements listed in the previous section and taking into account a wide variety of emerging international activities within the e-Infrastructure context, this section aims to outline a strategic roadmap towards 2020. Figure 1 illustrates a graphical representation while we need to acknowledge that not all possible directions (i.e. arrows) are fully described in this paper due to the paper page limit. In particular the 'big data' related directions and options for UNICORE are illustrated with green arrows while the more traditional directions in context of computing are indicated using red arrows in Figure 1.

The state-of-the-art of the roadmap clearly begins with the expertise of the UNICORE community in context of HPC that over the years was extended to HTC, and more recently, also more and more towards the handling and processing of 'big data'. The consequence of this expertise is the involvement and the usage of UNICORE in key projects of the e-Infrastructure community as shown on the left side of Figure 1. Together with gLite and ARC, UNICORE is at the time of writing part of the EMI project[31] that in turn enabled UNICORE to be part of the EGI UMD distribution rolled out on several NGIs. On the other end of the scale, UNICORE is used within PRACE[27], which in turn also contributed in parts to the fact that UNICORE is considered to be used in XSEDE as subsequent sections of this roadmap will reveal. The state-of-the-art also includes that several UNICORE partners are involved in user-oriented construction or pre-production phases of ESFRI research infrastructures (e.g. CLARIN, DARIAH, SKA, etc.).

Figure 1. UNICORE roadmap outlining strategic options towards 2020.

3.1 With Open Standards Adoptions to XSEDE

One important key driver of the UNICORE technology is the adoption of open standard specifications from Standard Development Organizations (SDOs). UNICORE adopts the IETF X.509 standard[1] as well as several OASIS standards such as the Web Services Resource Framework (WSRF)[10], the Security Assertion Markup Language (SAML)[9] or the Extensible Access Control Markup Language (XACML)[8]. Many other relevant adopted OGF standards include the Job Submission Description Language (JSDL)[4], the OGSA Basic Execution Service (BES)[3], or the Grid Laboratory Uniform Environment Version 2 (GLUE2)[6]. These standards have been used with many scientific applications[19,25,24] using interoperable e-Infrastructures. The UNICORE community should continue its standard adoption activities and its engagements within user-oriented OGF groups such as the Grid Interoperation Now (GIN)[12] and Production Grid Infrastructure (PGI)[2].

As Figure 1 illustrates, the OGF and OASIS standard adoptions lead to the usage of UNICORE in Extreme Science and Engineering Discovery Environment (XSEDE)[17]. One of the key principle of the emerging XSEDE architecture is to separate its architecture from implementation that in turn is fundamentally supported by using open standards in the XSEDE architectural design. The architecture and the roll-out plan on the XSEDE architecture follows the guidance by software engineering experts from Carnegie Mellon University. As a consequence of the strong standard adoption of UNICORE and the resulting interoperability with other known technologies (e.g. GENESIS[18], etc.), UNICORE is going to be deployed at major US supercomputing centres in the next years.

3.2 The EMI Legacy: Science Soft

The EMI project releases software that is required by many different user communities and e-Infrastructures such as EGI and PRACE as shown in Figure 1. But the collaboration as part of the EMI project ends in 2013 alongside many other projects within the e-Infrastructure community such as the Initiative for Globus in Europe (IGE)[41]. Although the sustainability of many technology parts of EMI is solved, there are still challenging questions to which extend streamlined releases for EGI UMD, active collaboration, and technical harmonization activities between EMI technologies are funded in the future. While the EMI project still discusses possibilities for further collaboration, it has initiated a complementary initiative that bear the potential to sustain at least parts of its activities and extending them towards new partners of the science domain. As shown in Figure 1, this new initiative is named 'ScienceSoft'[28] and still in the concept phase in order to enable new partners to form and steer 'ScienceSoft' to a degree that it makes sense for them to join its activities. At the time of writing, ScienceSoft is planned to assist scientific communities in finding the software they need and to promote development and use of open source software for scientific research. The key added-value is considered to be a 'one-stop-shop' to match scientific and academic user needs with already existing software products and services aiming at reducing the duplication of software developments where possible.

In order to reach the aforementioned added-value, several approaches have been already discussed at workshops and are planned to be part of ScienceSoft. 'Identifiers' need to be in place to identify software and authors with clear criteria and properties. This includes citation of software a user is using and a citation of software out of publications that typically often are not able to clearly reference which software products in which versions

contributed to the results of the paper. 'Catalogues of Software and Services' will be established in order to find the right applications and tools that a user needs. This includes numerous features such as a search by platform, functionality, or even user ratings. All these mechanisms eventually sum up into a rich 'Marketplace' of software, services, and professional skills of developers, administrators, or even technical managers. This marketplace allows to match requirements against the available offers and enable the sharing of ideas and solutions in an unprecedented manner. 'Collaboration' is thus significantly supported in a way that developers can better work together with end-users improving also the support of scientific software. Quality assurance following scientific processes such as peer-review are supported enabling the rating of software and to obtain user community recommendations. ScienceSoft members may also take part in dedicated technical groups or may form consortiums with new partners in order to engage in EC funding calls.

The UNICORE community should explore to which extend a participation in ScienceSoft makes sense, because the ScienceSoft idea is not completely new (e.g. Academia[43]). UNICORE representatives are part of the ScienceSoft steering committee alongside others from StratusLab, gLite, and EGI in order to explore and steer benefits for UNICORE. SourceForge already enabled tremendous results both in bringing an active development and administrator community together, but also to provide mature and reliable software packages to its users. The UNICORE Forum acts as a complementary umbrella organization for UNICORE partners and goes beyond project-oriented funding cycles. The engagement of UNICORE partners within ScienceSoft might be influenced of how much this activity is supported by the European Commission (EC) in the next years and to which extend user communities consider ScienceSoft really as beneficial.

3.3 Evolution of EGI and NGIs towards an EU Cloud Infrastructure

The previous paragraph already outlined an evolution of EMI towards ScienceSoft where also EGI is taking part and will be also affected by its transition period. This paragraph highlights the expected evolution of the e-Infrastructure EGI itself together with its National Grid Initiatives (NGIs) that also partly affects UNICORE and some of its strategic options for the future in the context of 'cloud computing' as Figure 1 reveals. The majority of users in the past are satisfied with UNICORE as Grid middleware, but the use of clouds promises benefits in the particular field around resource provisioning that may lead to strategic options for UNICORE in the next years. This is in particular very likely since the Digital Agenda of Europe is pushing the adoption of clouds in every part of science and society reaching one of its goal with 'Every EU citizen cloud-enabled by 2020'[30].

Figure 2 is a conceptual illustration of the transition process from a pure middleware-centric e-Infrastructure towards a more dynamic cloud-based e-Infrastructure in the next years[46]. 'Collaboration Platforms' such as data movement or information services are considered as the core of the infrastructure together with the physical resources. On top of the 'Infrastructure Platforms', individually federated 'Community Platforms' enable scientific researchers the customisation to the most possible degree. Several UNICORE components might be considered as part of those platforms, but will be also still one part of the 'Community Distribution' either as part of UMD or standalone. But the usage of UNICORE inside FutureGrid[44] already revealed challenges for the UNICORE in cloud-based setups (e.g. non-static hostnames, configurations, etc.). For a more elaborated description of the cloud-based platform setup of EGI refer to the EGI business model[46].

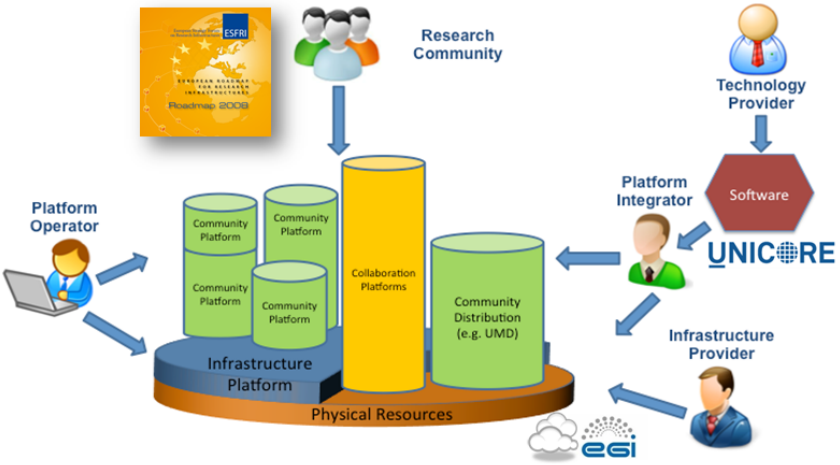

Figure 2. EGI transforms into an EU Cloud Infrastructure (modified from EGI business model[46]).

Apart from joining the strategic activities towards a cloud-based EGI, the UNICORE community needs to ensure the expansion of its user communities as part of a broader set of NGIs than those that rely on UNICORE today. As Figure 1 suggests, many other NGIs (e.g. ES-NGI, Grid Ireland, NGS, or Slowak Grid) have potential interest in UNICORE that can be at least partly explained by the fact that more low-scale HPC systems appear as part of the NGI-based EGI infrastructure.

3.4 Towards International Software Infrastructures

The previous paragraphs revealed how the e-Infrastructure setups in Europe and the US are going to change in the next couple of years. The interesting change of EGI towards a more cloud-based operational model and the transition of EMI towards ScienceSoft to open itself towards contributions and collaborations with a much more broader number of players in the e-Infrastructure community. This section provides recommendations towards 2020 taking into account recommendations that all seem to sum up towards the creation of 'international software infrastructures' as Figure 1 illustrates towards 2020. These are in-line with the ideas of the Open Middleware Infrastructure Institute of Europe (OMII-Europe)[45] in the last years and the currently established software infrastructure projects[47] in the US as well as ScienceSoft introduced in the previous paragraphs.

In order to ensure the creation of software infrastructures that matter, it is important to perform 'application enabling activities' that stand for a close collaboration with scientific communities understanding thus their real requirements towards research-supporting software like UNICORE. The e-Infrastructure Reflection Group (e-IRG)[48] and the European e-Infrastructure Forum (EEF)[50] already provide a good source of requirements (e.g. federated authorization and authentication infrastructure) in order to support the increasing demands of ESFRI and other scientific user communities.

3.5 Roadmap Extension: Processing and Avoiding 'Big Data'

As the previous paragraphs reveal, the next decade clearly present many opportunities, but also several challenges for the UNICORE community. 'Big data requirements and challenges' such as those raised in the High Level Expert Group on Scientific Data Report[51] will have an impact on how science and engineering will be conducted towards 2020. But as Figure 1 suggests, the UNICORE community already expanded in the last years its traditional 'HPC and HTC strength' in computing with a complementary 'big data strength' (e.g. WISNET Grid project[34], EUDAT[33], etc.). This needs to be continued in the next decade, because handling of 'big data' and its analysis is expected to be one of the most significant requirements of user communities arising from the ESFRI projects that partly rely on scientific measurement devices that gather petabytes of data.

As part of the EUDAT project[33], several UNICORE partners collaborate with data centres in Europe and many user communities to create a collaborative data infrastructure (CDI) that federates the expertise of user community centres with those existing in large data and computing centres. An overall vision of many EUDAT activities is the creation of a 'ScienceTube'[16] as shown in Figure 3 that enables the scientific researchers to 'dive' into data sets instead of just accessing them. The vision supports the idea that 'everything is data' thus connecting publications with real scientific data and in turn with measurement data collected from large scientific instruments while access to high capacities of storage are only one click away. Data recommendation systems and data analysis and processing tools can be seamlessly used with scientific data sets thus leading to new insights based on fully multi-disciplinary scientific approaches. The UNICORE consortium is in an excellent position, because of its expertise in computational science, to contribute to the avoidance of 'big data' instead of just handling it. This relates to advanced processing techniques that work against the duplication of data with reproducibility methods or powerful data analysis approaches (e.g. MapReduce, R language, etc.) and sharing data methods that altogether avoid or at least lower 'big data waves'.

The technical concept of ScienceTube in turn is underpinned with many known building blocks where the UNICORE community has proven to have expertise in the past such as access to HPC and HTC resources of existing e-Infrastructures. But specifically more approaches around the access, management, and analysis of scientific data is required to fulfil this vision that also includes solutions for a federated security. UNICORE partners in DEISA, WISNETGrid, and EMI have already started to extend existing approaches towards new security solutions (e.g. Shibboleth, etc.) since new emerging security frameworks will be required in order to cope with the flexibility needed to realize the 'ScienceTube' vision. Standardization as being already successful in the computing area of UNICORE leading to XSEDE, needs to be also important for EUDAT and the coming data activities of the UNICORE community. Examples of emerging standards of interest are those from OASIS OData[49] or the OGF Data Format Description Language (DFDL)[7].

Increasing the experience around data processing and working together with user communities and by performing together 'application enabling', new collaborations may be formed that will lead in the future to new exciting EC or NSF projects that take advantage of UNICORE components (e.g. data-staging, security methods, etc.). As shown in Figure 1, total multi-disciplinary data-intensive science is also expected to be a key part of the Horizon 2020 framework of the EC, including a key role of e-Infrastructures.

Figure 3. The ScienceTube vision relies on a powerful backend that partly takes advantage of UNICORE.

4 Conclusions

The stock taking of the roadmap created in 2010 and the new strategic options that are realistically achievable by the UNICORE community clearly provide evidence that the UNICORE technology is the technology of choice for e-Infrastructures world-wide. The success today is based on clear strategies followed in the past, extraordinary software engineering based on open standards to the most possible degree and innovative end user communities that are willing to engage in new approaches to computational science. But the UNICORE community should not anticipate the changes that we have to make - but drive them (cf. to Explicit Trust Delegation vs. proxy delegation in the past). Therefore, for the continuation of this success it is recommended to take into account the following six high-level actions derived from the 2020 roadmap as conclusions:

Action 1: Continue to adopt open standards to encourage trust by end-users (no vendor-locks) and contribute to open standard processes that enable the separation of e-Infrastructure architectures and their implementations (cf. EGEE and gLite vs. EGI and UMD or TeraGrid and Globus vs. XSEDE and standards). This includes the adoption of software engineering processes such as those from Carnegie Mellon University provided in XSEDE in order to continue to provide good quality software in decades to come.

Action 2: Explore the opportunities for UNICORE of the new movement of Science-Soft in order to prevent that other middlewares take advantage of possible sole visibility for middleware in Europe and in science and engineering.

Action 3: Understand the challenges that arise when running UNICORE in virtualized environments such as those envisaged by EGI and as part of FutureGrid. Not every HPC e-Infrastructure considers clouds, but HTC-oriented e-Infrastructures as well as e-Business environments might take advantage of clouds in the next years.

Action 4: Participate in the international discourse around international software infrastructures to ensure that UNICORE is part of discussions that reach decision makers of 'big science projects' and not only the 'long tale science endeavours'. It will be more and more important in the next years to overcome the boundaries of Europe and US in order to seamlessly work together to support innovative science with e-Infrastructures. Addressing e-IRG and EEF recommendations is important in order to serve the needs of ESFRI projects that will increase the users of UNICORE by an order of magnitude.

Action 5: Understand that handling 'big data' is important while at the same time it is even more important to avoid 'big data creation' in the most possible way. Align activities towards the 'ScienceTube' vision in order to be flexible for more dynamic Web-based gateways that will take advantage of UNICORE computational, storage, and data analytics functionality in the future.

Action 6: Scientific applications are the key to all of these previous actions and it should be ensured in the future that UNICORE partners perform 'application enabling activities' with scientific end user communities to create solutions that truly contribute to tackle the societal challenges towards 2020.

References

1. R. Housley, W. Ford, W. Polk and D. Solo, *Internet X.509 Public Key Infrastructure*, IETF RFC 2459, 1999.
2. M. Riedel and J. Watzl, *OGF Production Grid Infrastructure: Use Case Collection Version 1*, OGF GFD 180, 2011.
3. I. Foster and A. Grimshaw et al., *OGSA Basic Execution Service Version 1.0*, OGF GFD 108, 2007.
4. A. Anjomshoaa et al., *Job Submission Description Language Specification Version 1.0*, OGF GFD 136, 2008.
5. A. Sim et al., *The Storage Resource Manager Interface Specification Version 2.2*, OGF GFD 129, 2008.
6. S. Andreozzi et al., *GLUE Specification Version 2.0*, OGF GFD 147, 2009.
7. A. Powell et al., *Data Format Description Language Version 1.0*, OGF GFD 174, 2011.
8. T. Moses et al., *Extensible Access Control Markup Language Version 2.0 - Core Specification*, OASIS Standard, 2005.
9. S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language - Version 2.0*, OASIS Standard, 2005.
10. T. Banks et al., *Web Service Resource Framework Primer*, OASIS Standard, 2003.
11. S. Zasada et al., *Virtualizing Access to Scientific Applications with the Application Hosting Environment*, Computer Physics Communications **180**, 2513–2525, 2009.
12. M. Riedel et al., *Interoperation of World-Wide Production e-Science Infrastructures*, Concurrency and Computation: Practice and Experience **21**, 961–990, 2009.

13. L. Cabellos et al., *Scientific Workflow Orchestration interoperating HTC and HPC Resources*, Computer Physics Communications **182**, 890–897, 2011.
14. W. Gentzsch et al., *DEISA - Distributed European Infrastructure for Supercomputing Applications*, Journal of Grid Computing **9**, 259–277, 2011.
15. M. Viceconti et al., *The Virtual Physiological Human - a European Initiative for in silico human modeling*, Journal of Physiological Sciences **58**, 441-446, 2008.
16. M. Riedel and P. Wittenburg et al., *Reference Model Design of a Federated Data Infrastructure - Towards the Realization of a ScienceTube Vision*, Journal of Internet Services and Applications: Special Issue on Data Intensive Computing, 2012.
17. M. Riedel and B. Demuth, *UNICORE in XSEDE: Towards a Large-scale Scientific Environment based on Open Standards*, inSide **9**, 52–53, 2011.
18. M. Morgan and S. Grimshaw, *GENESIS II - Standards Based Grid Computing*, Proceedings of CCGRID '07, 611–618, 2007.
19. M. Riedel et al., *Exploring the Potential of Using Multiple e-Science Infrastructures with Emerging Open Standard-based e-Health Research Tools*, Proceedings of CCGRID '10, 341-348, 2010.
20. A. Di Meglio et al., *Grids and Clouds Integration and Interoperability: An Overview*, Proceedings of Science - ISGC, 2011.
21. M. Riedel et al., *The Key Role of the UNICORE Technology in European Distributed Computing Infrastructures, Supporting e-Science Applications in the Decades to Comei*, Proceedings of UNICORE Summit 2010, 83–94, 2010.
22. I. Plasencia et al., *Modelling Mixed Workflows between Grid and HPC in EUFORIA*, Proceedings of 3rd Iberian Grid Infrastructure Conference, 256–265, 2009.
23. C. Loeschen et al., *Grid Data Management for UNICORE*, Proceedings of EGI Community Forum, 2012.
24. M. Riedel et al., *Improving e-Science with Interoperability of the e-Infrastructures EGEE and DEISA* Proceedings of MIPRO, 225–231, 2008.
25. M. Memon et al., *Lessons Learned from Jointly Using HTC- and HPC-driven e-Science Infrastructures in Fusion Science*, Proceedings of ICIET, 2010.
26. S. Vaerttoe et al., *DEISA - Advancing Science in Europe*, ISBN 978-952-5520-32-3, 2008.
27. PRACE press release, `http://www.prace-ri.eu/PRACE-partner-Julich-joins-XSEDE`, accessed October 2010.
28. ScienceSoft Web page, `http://www.sciencesoft.org`, accessed October 2010.
29. European Commission Horizon 2020, `http://ec.europa.eu/research/horizon2020`, accessed October 2010.
30. European Commission Digital Agenda for Europe, `http://ec.europa.eu/information_society/digital-agenda`, accessed October 2010.
31. European Middleware Initiative (EMI) Web page, `http://www.eu-emi.eu`, accessed October 2010.
32. European Strategy Forum on Research Infrastructures (ESFRI) Web page, `http://ec.europa.eu/research/infrastructures`, accessed October 2010.
33. European Data Infrastructure EUDAT, `http://www.eudat.eu`, accessed October 2010.

34. WISNETGrid Project Web page, http://www.wisnetgrid.org, accessed October 2010.
35. LHC Cern Web page, http://lhc.web.cern.ch/lhc, accessed October 2010.
36. CLARIN Project Web page, http://www.clarin.eu, accessed October 2010.
37. SKA Project Web page, http://www.skatelescope.org, accessed October 2010.
38. SLA4D-Grid Web page, http://www.sla4d-grid.de, accessed October 2010.
39. DARIAH Web page, http://www.dariah.eu, accessed October 2010.
40. SMARTLM Web page, http://www.smartlm.eu, accessed October 2010.
41. IGE Project Web page, http://www.ige-project.eu, accessed October 2010.
42. Fit4Green Project Web page, http://www.fit4green.eu, accessed October 2010.
43. Academia Web page, http://www.academia.edu, accessed October 2010.
44. FutureGrid Project Web page, http://www.futuregrid.org, accessed October 2010.
45. OMII-Europe Project Web page, http://www.omii-europe.org, accessed October 2010.
46. EGI Business Model Web page, http://documents.egi.eu/public/ShowDocument?docid=1040, accessed October 2010.
47. NSF Software Infrastructures Web page, http://www.nsf.gov/si2, accessed October 2010.
48. e-Infrastructure Reflection Group Web page, http://www.e-irg.eu, accessed October 2010.
49. OData Web page, http://www.odata.org, accessed October 2010.
50. EEF Web page, http://www.einfrastructure-forum.eu, accessed October 2010.
51. High Level Report on Scientific Data, http://cordis.europa.eu/fp7/ict/e-infrastructure/docs/hlg-sdi-report.pdf, accessed October 2010.

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

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

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 2012, the eighth in its series, took place 30 – 31 May at Dresden University of Technology, Dresden, Germany.

The proceedings at hand include a selection of 14 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 15
ISBN 978-3-89336-829-7

JÜLICH
FORSCHUNGSZENTRUM